



**DIN EN ISO 9001:2000
certified**



**ADDI-DATA GmbH
Dieselstraße 3
D-77833 OTTERSWEIER**



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

Technical description

ADDIALOG APCI-/CPCI-3120

**Analog input and output board
for the PCI/CompactPCI bus**

7th edition 06/2005

Product information

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

Warranty and liability

The user is not permitted to make changes to the product beyond the intended use, or to interfere with the product in any other way.

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

Copyright

This manual, which is intended for the operator and its staff only, is protected by copyright. Duplication of the information contained in the operating instructions and of any other product information, or disclosure of this information for use by third parties, is not permitted, unless this right has been granted by the product licence issued. Non-compliance with this could lead to civil and criminal proceedings.

ADDI-DATA software product licence

Please read this licence carefully before using the standard software. The customer is only granted the right to use this software if he/she agrees with the conditions of this licence.

The software must only be used to set up the ADDI-DATA boards.

Reproduction of the software is forbidden (except for back-up and for exchange of faulty data carriers). Disassembly, decompilation, decryption and reverse engineering of the software are forbidden. This licence and the software may be transferred to a third party if this party has acquired a board by purchase, has agreed to all the conditions in this licence contract and the original owner does not keep any copies of the software.

Trademarks

- ADDI-DATA is a registered trademark of ADDI-DATA GmbH.
- Turbo Pascal, Delphi, Borland C, Borland C++ are registered trademarks of Borland Insight Company.
- Microsoft C, Visual C++, Windows XP, 98, Windows 2000, Windows 95, Windows NT, EmbeddedNT and MS DOS are registered trademarks of Microsoft Corporation.
- LabVIEW, LabWindows/CVI, DasyLab, Diadem are registered trademarks of National Instruments Corp.
- CompactPCI is a registered trademark of PCI Industrial Computer Manufacturers Group.
- VxWorks is a registered trademark of Wind River Systems Inc.

WARNING

The following risks result from improper implementation and from use of the board contrary to the regulations:



- ◆ **Personal injury**
- ◆ **Damage to the MSX-Box, PC and peripherals**
- ◆ **Pollution of the environment**

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

◆ **Read carefully the safety precautions (yellow leaflet).**

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

◆ **Observe the instructions of the manual.**

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

◆ **Used symbols:**



IMPORTANT!

designates hints and other useful information.



WARNING!

It designates a possibly dangerous situation.

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

1	DEFINITION OF APPLICATION	8
1.1	Intended use	8
1.2	Usage restrictions.....	8
1.3	General description of the board	8
2	USER	10
2.1	Qualification	10
2.2	Personal protection.....	10
3	HANDLING OF THE BOARD	11
4	TECHNICAL DATA	12
4.1	Electromagnetic compatibility (EMC)	12
4.2	Physical set-up of the board.....	12
4.3	Options	13
4.4	Versions	13
4.5	Limit values.....	13
4.6	Component schemes	17
5	SETTINGS OF THE BOARD	19
5.1	Settings at delivery.....	19
5.1.1	Jumper location at delivery	19
5.1.2	Jumper settings according to the function used	20
6	INSTALLATION OF THE BOARD	21
6.1	Installation of the APCI-3120 board.....	21
6.1.1	Opening the PC	21
6.1.2	Selecting a free slot	21
6.1.3	Plugging the board into the slot	22
6.1.4	Closing the PC	22
6.2	Installing a CPCI-3120 board	23
7	SOFTWARE	25
7.1	Board registration with ADDIREG	26
7.1.1	Installing a new board	26
7.1.2	MORE information	29
7.1.3	PCI analog input boards with DMA	29
7.1.4	Registering a new board.....	32
7.1.5	Changing the registration of a board	33
7.2	Questions and software downloads on the web.....	34

8	CONNECTING THE PERIPHERAL.....	35
8.1	Connection principle	35
8.2	Connector pin assignment.....	35
8.3	Connection examples.....	36
8.3.1	Analog inputs channels.....	36
8.3.2	Digital input and output channels	37
8.3.3	Connection to the PX 901 screw terminal board.....	38
9	FUNCTIONS OF THE BOARD	39
9.1	Analog output channels	39
9.2	ANALOG input channels	39
9.3	Time-multiplex system.....	40
10	CALIBRATION TOOL	42
10.1	Introduction	42
10.1.1	General description.....	42
10.1.2	Requirements	42
10.2	Installation of the calibration tool.....	43
10.2.1	Remark for Windows 98/ Windows 2000 / Windows XP	43
10.2.2	Board registration.....	43
10.3	Board preparation.....	44
10.3.1	Calibration of the analogue input	44
10.3.2	Analogue input calibration	46
10.4	Using the calibration tool.....	47
10.4.1	Board resources and information	47
10.4.2	Define the test report file and directory	49
10.5	Calibration	50
10.5.1	Analogue input calibration	50
10.5.2	Analogue output calibration	55
10.6	Error messages	57
10.6.1	Possible error messages	57
11	SOFTWARE EXAMPLES	60
11.1	Initialisation.....	60
11.1.1	Initialisation of an APCI-/CPCI-3120 board.....	60
11.1.2	Initialisation of several APCI-/CPCI-3120 boards	62
11.2	Interrupt.....	64
11.2.1	Interrupt routine	64
11.3	Direct conversion of the analog input channels.....	68

11.3.1	Testing an analog input channel	68
11.3.2	Testing several analog input channels.....	70
11.4	Cyclic conversion of the analog input channels.....	72
11.4.1	Cyclic conversion without DMA, external trigger and delay	72
11.4.2	Cyclic conversion with DMA, without external trigger and delay ..	77
11.5	Analog output channels.....	82
11.5.1	Testing one analog output channel.....	82
11.5.2	Testing several analog output channels	84
11.6	Timer.....	86
11.6.1	Testing the timer interrupt	86
11.6.2	Testing the watchdog	91
11.7	Digital input channels.....	93
11.7.1	Reading a digital input channel	93
11.7.2	Reading 4 digital input channels	95
11.8	Digital output channels.....	97
11.8.1	Testing the digital output memory	97

Figures

Fig. 3-1:	Correct handling of the APCI-3120.....	11
Fig. 3-2:	Correct handling of the CPCI-3120	11
Fig. 4-1:	Component scheme of the APCI-3120	17
Fig. 4-2:	Component scheme of the CPCI-3120	18
Fig. 5-1:	Jumper location on the board APCI-3120.....	19
Fig. 5-2:	Jumper location on the board CPCI-3120	19
Fig. 5-3:	Settings at delivery	20
Fig. 6-1:	PCI-5V slot (32-bit)	21
Fig. 6-2:	Opening the blister pack	21
Fig. 6-3:	Inserting the board.....	22
Fig. 6-4:	Fastening the board at the back cover.....	22
Fig. 6-5:	Types of slots for CompactPCI boards	23
Fig. 6-6:	Pushing a CPCI board into a rack.....	23
Fig. 6-7:	Connector keying	24
Fig. 7-1:	ADDIREG registration program (example).....	26
Fig. 7-2:	Selecting a new board	28
Fig. 7-3:	PCI DMA management (Example)	30
Fig. 8-1:	37-pin SUB-D male connector	35
Fig. 8-2:	16-pin ribbon cable connected to to 37-pin SUB-D male connector.....	36
Fig. 8-3:	Analog input channels (SE)	36
Fig. 8-4:	Analog input channels (Diff).....	37
Fig. 8-5:	Digital input channels	37
Fig. 8-6:	Digital output channels.....	37

Fig. 8-7: Connection to the screw terminal board PX901	38
Fig. 10-1: Jumper location	44
Fig. 10-2: Configuration in the single mode	45
Fig. 10-3: Configuration in the differential mode	45
Fig. 10-4: 37-pin SUB D front connector	46
Fig. 10-5: 37-pin SUB-D connector	47
Fig. 10-6: Window: Starting the calibration tool	47
Fig. 10-7: Window: Test report file name	49
Fig. 10-8: Window: Calibration tool version	50
Fig. 10-9: Window: Voltage value -0.00061	50
Fig. 10-10: Window: Measured value is out of range	51
Fig. 10-11: Window: Voltage value 9.995 volt	51
Fig. 10-12: Window: Voltage value -0.00061 volt	52
Fig. 10-13: Window: Voltage value 9.995 volt	52
Fig. 10-14: Window: Voltage value 5.0 volt	53
Fig. 10-15: Window: Analogue input is calibrated	54
Fig. 10-16: Window: Analogue input is not calibrated	54
Fig. 10-17: Window: Analogue output calibration	55
Fig. 10-18: Window: Analogue output potentiometer successfully saved	55
Fig. 10-19: Window: The potentiometer value could not be saved	56

1 DEFINITION OF APPLICATION

1.1 Intended use

The board **APCI-3120** must be inserted in a PC with PCI 5V/32-bit slots, which is used as electrical equipment for measurement, control and laboratory use as definedn in the norm IEC 61010-1.

The board **CPCI-3120** must be inserted in a CompactPCI/PXI computer with Compact PCI 5V/32-bit slots, which is used as electrical equipment for measurement, control and laboratory use as definedn in the norm IEC 61010-1.

1.2 Usage restrictions

The APCI-/CPCI-3120 board must not to be used as safety related part for securing emergency stop functions.

The board must not be used in potentially explosive atmospheres.

1.3 General description of the board

Data exchange between the **APCI-/CPCI-3120** board and the peripheral is to occur through a shielded cable. This cable must be connected to the 37-pin SUB-D male connector of the **APCI-/CPCI-3120** board

The board has up to 16 input channels and 8 output channels for processing analog signals and 4 input channels and 4 output channels for processing digital 24 V signals.

The **PX901** screw terminal board allows the connection of the analog signals with a shielded cable. The use of the board **APCI-/CPCI-3120** in combination with external screw terminal or relay boards is to occur in a closed switch cabinet.

The installation is to be effected competently. **Check the shielding capacity** of the PC housing and of the cable prior to putting the device into operation.

The connection with our standard cable ST010 complies with the following specifications:

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

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.

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.

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

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

2 USER

2.1 Qualification

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

- installation
- use,
- maintenance.

2.2 Personal protection

Consider the country-specific regulations about:

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

3 HANDLING OF THE BOARD

Fig. 3-1: Correct handling of the APCI-3120

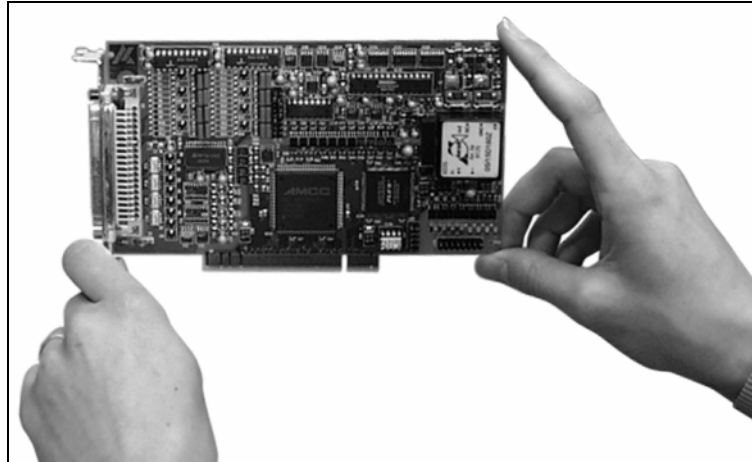
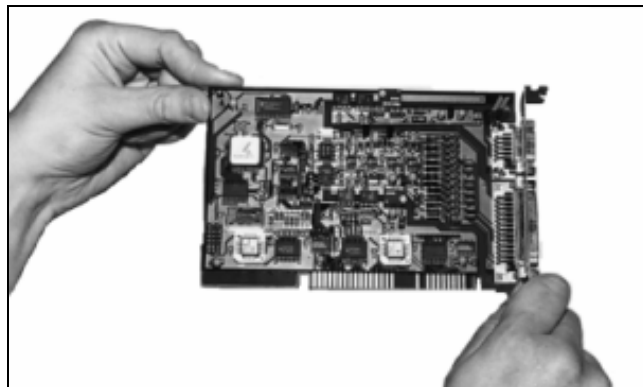


Fig. 3-2: Correct handling of the CPCI-3120



4 TECHNICAL DATA

4.1 Electromagnetic compatibility (EMC)

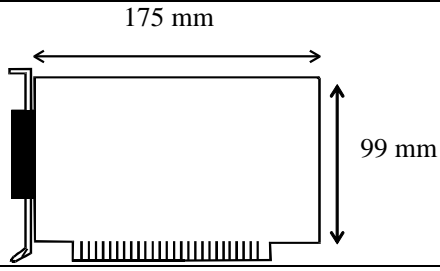
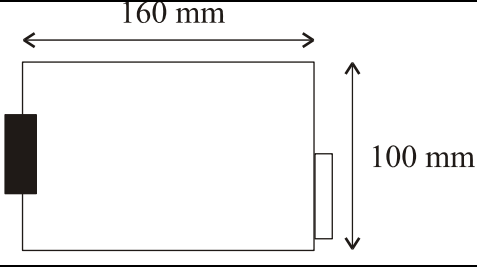
The PC is to comply with the norm IEC61326 for measurement, control and laboratory use and with the specifications for EMC protection.

The board has been subjected to EMC tests in an accredited laboratory. The board complies with the limit values set by the norms IEC61326 as follows:

	True value	Set value
ESD (Discharge by contact/air).....	4/8 kV	4/8 kV
Fields	10 V/m	10 V/m
Burst	4 kV	2 kV
Conducted radio interferences	10 V	10 V

4.2 Physical set-up of the board

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

	APCI-3120	CPCI-3120
Dimensions		
Weight	156 g	200 g
Installation in	PCI-5V (32-bit) or PCI-5V (64-bit) slot	CompactPCI-5V(32-bit) or CompactPCI-5V (64-bit) slot
Connection to the peripheral	37-pin SUB-D male connector	37-pin SUB-D male connector

Connection possibilities to the peripheral:

- through a cable with twisted pairs directly to the analog signal transmitters
- or with our standard cable ST010 to screw terminal board PX 901-AG, PX 901-A.

4.3 Options

- SF, DF** Precision filter for the analog input channels
- PC** Precision current inputs 0-20 mA or 4-20 mA for the analog input channels

Remark: By 4-20 mA the accuracy is altered.

4.4 Versions

The **APCI-/CPCI-3120** board is available in the following versions.

Version	Analog input channels	Analog output channels
xPCI-3120-8-4	8 SE ¹ / 4 Diff. ²	4
xPCI-3120-8-8	8 SE / 4 Diff.	8
xPCI-3120-16-4	16 SE / 8 Diff.	4
xPCI-3120-16-8	16 SE / 8 Diff.	8

4.5 Limit values

Max. altitude: 2000 m
 Operating temperature: 0 to 60°C
 Storage temperature: -25 to 70°C
 Relative humidity: 30% to 99% non condensing

Minimum PC requirements (APCI-3120):

PCI BIOS from Version 1.0

Bus speed: < 33 MHz

Minimum system requirements (CPCI-3120):

- 32-Bit CompactPCI bus (5 Volt)
- bus speed: ≤ 33 MHz
- PCI BIOS, PCI 2.1 specification and CompactPCI 2.1 "compliant"
- 3 U format according to IEEE-1101

Operating system: MS DOS 6.22 or higher
 Windows 3.1, NT 4.0, 9x, 2000, XP

¹ SE for Single-Ended

² Diff. for differential

Energy requirements:

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

	xPCI-3120-8-x	APCI-3120-16-x	CPCI-3120-16-x
+ 5 V from the PC	997 mA	1048 mA	1030 mA

Analog input channels:

Number of analog input channels: 16 SE/ 8 diff. for **xPCI--3120-16x**
8 SE / 4 diff. for **xPCI3120-8x**

Analog resolution: 16-bit, 1 among 65535

Max. sampling rate (1 input channel): 100 kHz

Data transfer : Data to the PC (16-bit only)
via FIFO memory
1) through I/O commands
2) Interrupt at EOC¹ and EOS²
3) DMA transfer at EOC

Start of conversion: 1) per software trigger
2) TIMER 0
3) TIMER 0 and 1
4) external trigger

Monotony: 13-bit

Offset error: after calibration:

APCI- 3120: - Bipolar: \pm 1/2 LSB

- Unipolar: \pm 1/2 LSB

CPCI- 3120: - Bipolar: \pm 1 LSB

- Unipolar: \pm 1 LSB

Drift (0°C to 60°C):

- Bipolar: \pm 2 ppm / °C

- Unipolar: \pm 2 ppm / °C

Gain error: after calibration:

APCI- 3120: - Bipolar: \pm 1/2 LSB

- Unipolar: \pm 1/2 LSB

CPCI- 3120: - Bipolar: \pm 1 LSB

- Unipolar: \pm 1 LSB

Drift (0°C to 60°C):

- Bipolar: \pm 7 ppm / °C

Unipolar: \pm 7 ppm / °C

Analog input ranges: Voltage
Unipolar: 0-10 V
Bipolar: \pm 10 V
Selectable by software

¹ EOC: End of Conversion

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

Analog input channels (continued)

Analog input ranges:	Current
	Unipolar: 0-20 mA
	Selection of the range 0-10 V and of gain x2 is necessary
Overvoltage protection:	20 V when POWER ON
Common mode rejection:	DC up to 10 Hz, 90 dB mini. (Gain = 1)
Band width (-3dB):	Limited to 159 kHz (-3dB) with low-pass filter 1st order; yet the minimum SINAD is still 83 dB at 49 kHz (fin)
Bias currents for each input channel (multiplexer)	± 2 nA max.
Input impedance (PGA):	$10^{12} \Omega // 20$ nF to GND
Integral non-linearity (INL):	± 3 LSB
Differential non-linearity (DNL)	
APCI-3120:	$\pm 2 \frac{1}{2}$ LSB
CPCI-3120:	16-bit, no missing code
Accuracy:	± 1 LSB
Selectable gain:	via PGA gain 1, 2, 5, 10 (selectable by software)
System noise:	Bipolar:
	Gain x1: $\pm 2 \frac{1}{2}$ LSB
	Gain x2: $\pm 2 \frac{1}{2}$ LSB
	Gain x10: ± 6 LSB
	Unipolar:
	Gain x1: $\pm 2 \frac{1}{2}$ LSB
	Gain x2: $\pm 2 \frac{1}{2}$ LSB
	Gain x10: ± 6 LSB
Digital coding:	linear

Analog Input		Binary Code	HEX Code
Bipolar	Unipolar		
- 10V	0V	0000000000000000	0000
0V	5V	1000000000000000	8000
+10V	10V	1111111111111111	FFFF

Optical isolation to the PC: 500 VDC min.

Analog output channels:

Resolution:	14-bit unipolar / 13-bit bipolar
Overvoltage protection:	± 12 V
Number of output channels:	4 to 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 and automatically updated.

Analog output channels (continued):

Settling time at 0,01 % FS,

(FS = Full scale)

with 2 k Ω & 100 pF load \square 30 μ s typ. for a 20 V bounce
at 25°C

\square 50 μ s typ. for a 20 V bounce
above the temperature range

Output voltage ranges: Unipolar: 0-10 V

Bipolar: \pm 10 V

Digital coding: Unipolar: Straight binary coding

Bipolar: Offset binary coding

Output current: \pm 5 mA max.

Capacitive load: 500 pF max.

Short-circuit current: \pm 25 mA

Integral non-linearity (INL): \pm 1 LSB maximum above the
temperature range

Differential non-linearity (DNL): \pm 2 1/2 LSB maximum above the
temperature range

Monotony: 12-bit

Offset error: \pm 2 mV max. Unipolar

\pm 7 mV max. Bipolar

Gain error: \pm 0,05 % of FSR max.

Optical isolation to the PC: 500 VDC min.

Voltage after Reset: 0 V

Watchdog: can be configured by software
through Timer2

times of 100 μ s up to 838.8 s
are possible in steps of 50 μ s

Digital input channels:

Number: 4

Input current at 24 V: 3 mA typ.

Input voltage range: 0-30 V

Max. transfer rate: 5 kHz

Optical isolation: 1000 VAC

Logic „0“ level: 0-5 V

Logic „1“ level 12-30 V

Digital output channels:

Number: 4

Max. switch current: 10 mA typ.

Voltage range: 5-30 V

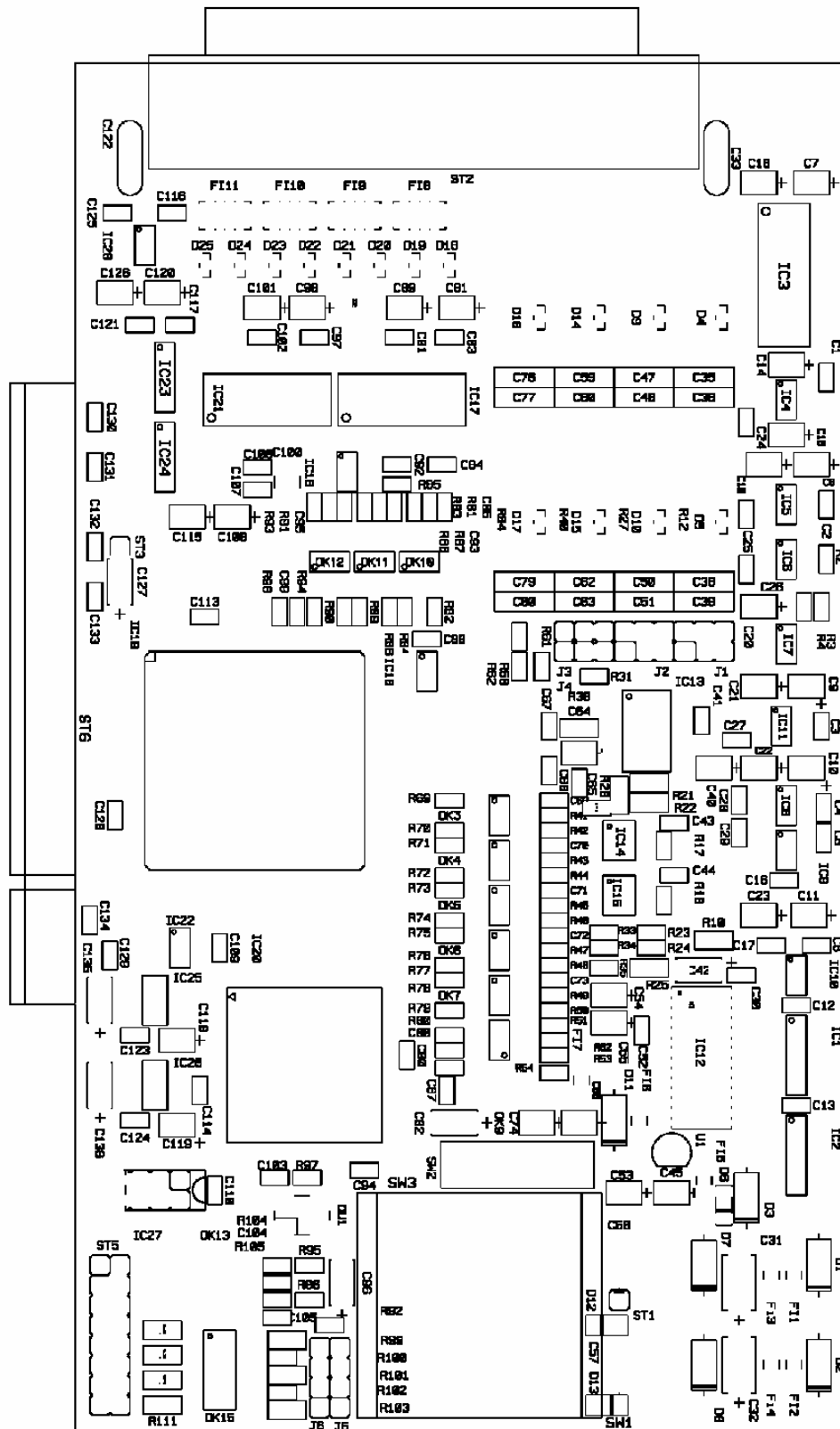
Max. transfer rate: 5 kHz

Optical isolation: 1000 VAC

Type: Open Collector

4.6 Component schemes

Fig. 4-1: Component scheme of the APCI-3120



5 SETTINGS OF THE BOARD



IMPORTANT!

Do observe the safety precautions (yellow leaflet)!

5.1 Settings at delivery

5.1.1 Jumper location at delivery

Fig. 5-1: Jumper location on the board APCI-3120

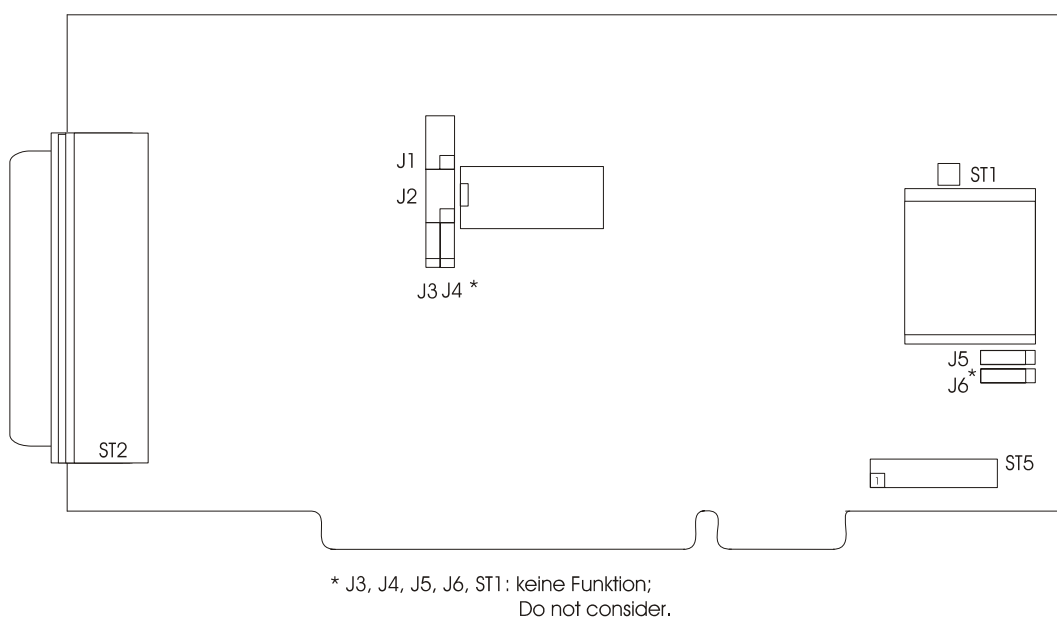


Fig. 5-2: Jumper location on the board CPCI-3120

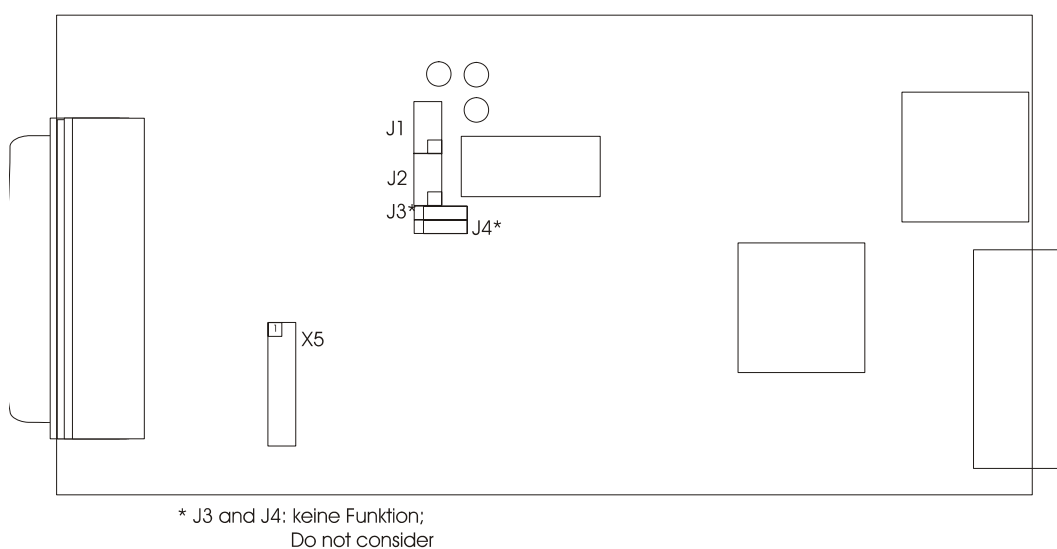
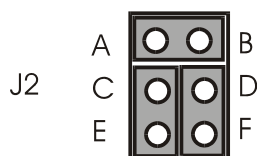
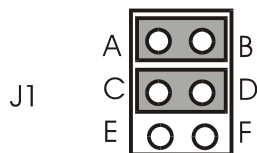


Fig. 5-3: Settings at delivery**APCI-3120****CPCI-3120**

Single-Ended



5.1.2 Jumper settings according to the function used

Table 5-1: Jumper settings according to the function used

Jumper	Position	Function	Delivery settings
J1	A-B, C-D	Single Ended measurement	✓
J1	A-C, D-F	Differential measurement	
J2	A-B, C-E, D-F	Single Ended measurement	✓
J2	B-D, C-E	Differential measurement	

6 INSTALLATION OF THE BOARD



IMPORTANT!

Do observe the safety precautions (yellow leaflet)!

6.1 Installation of the APCI-3120 board

6.1.1 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.

6.1.2 Selecting a free slot

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

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



32 bits

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

Discharge yourself from electrostatic charges.

Take the board out of its protective blister pack.

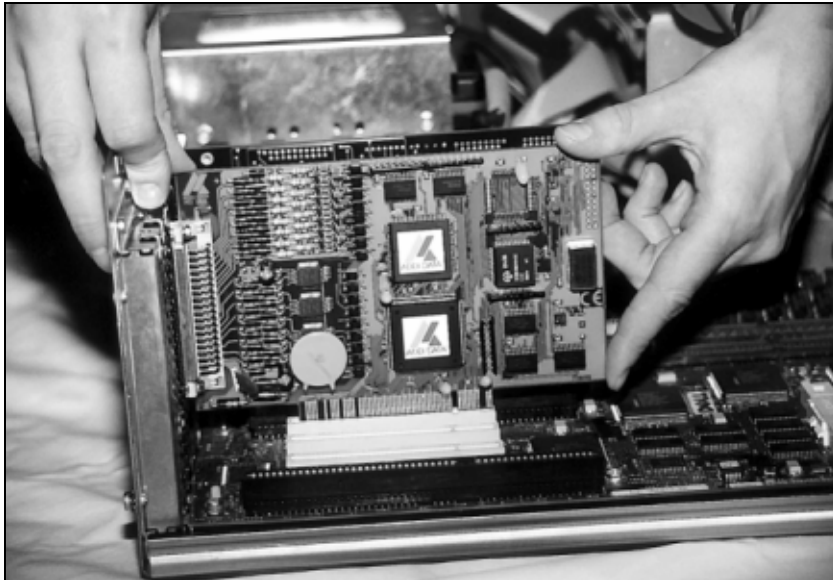
Fig. 6-2: Opening the blister pack



6.1.3 Plugging the board into the slot

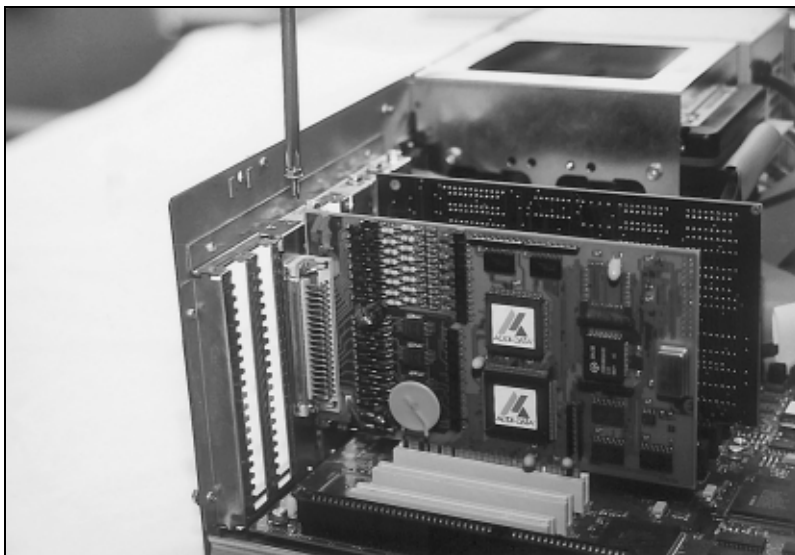
- ◆ Insert the board vertically into the chosen slot.

Fig. 6-3: Inserting the board



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

Fig. 6-4: Fastening the board at the back cover



- ◆ Tighten all the loosen screws.

6.1.4 Closing the PC

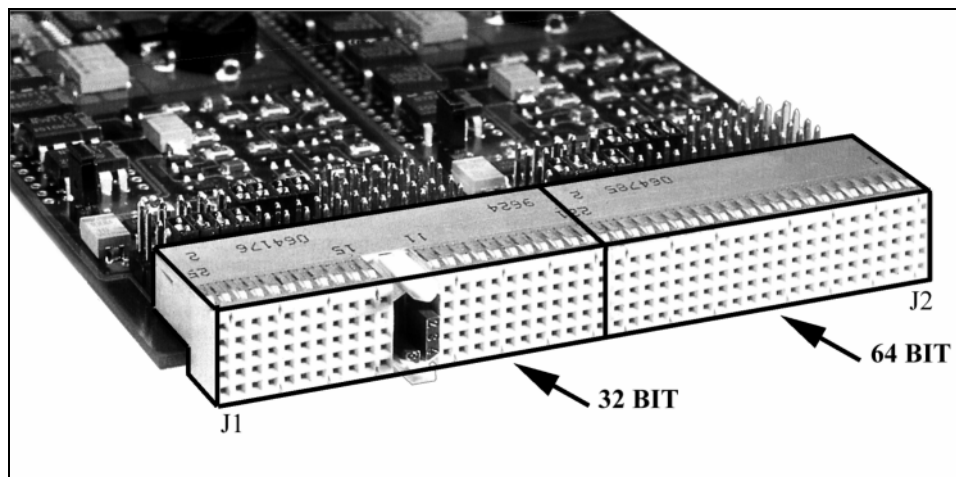
- ◆ Close your PC as described in the manual of the PC manufacturer.

6.2 Installing a CPCI-3120 board

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

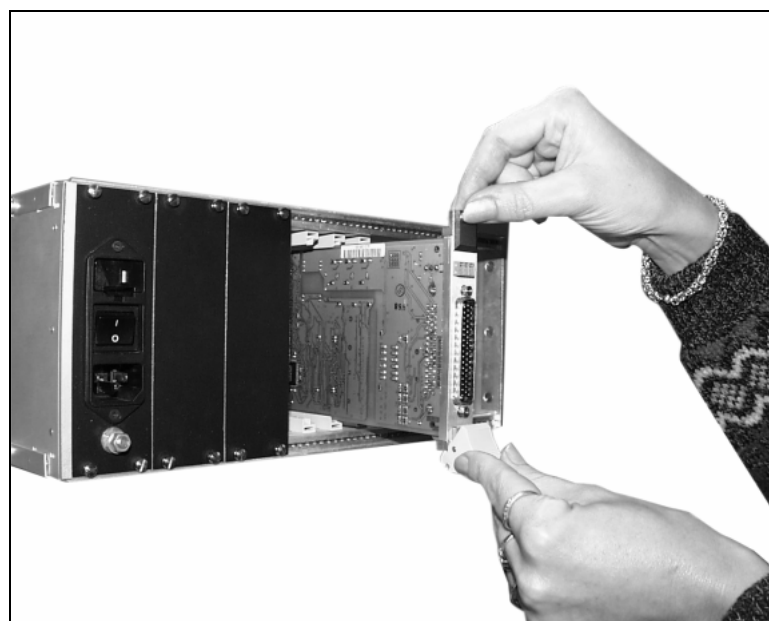
See in the computer manual which types of slots are free.

Fig. 6-5: Types of slots for CompactPCI boards



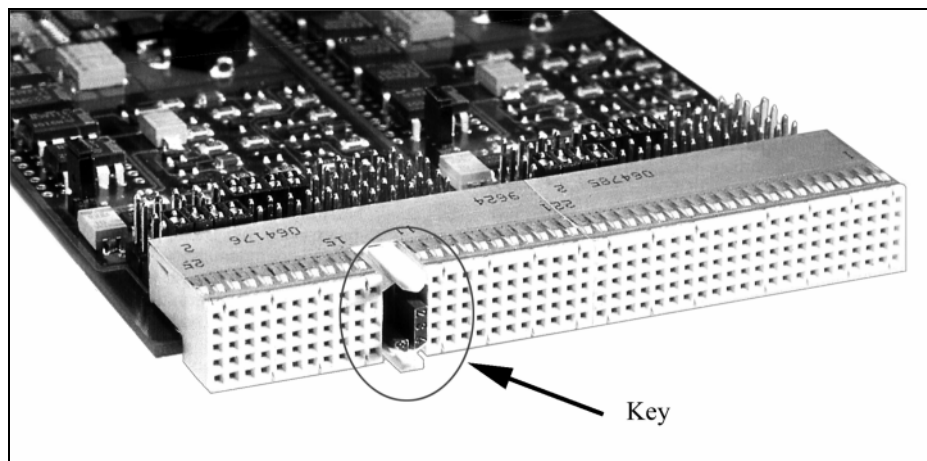
- ◆ Discharge yourself from electrostatic charges
- ◆ Hold the board at its grip (See handling of the board in chapter 3).
- ◆ Insert the board into the guiding rails and push it to the back cover of the rack. In order to fully insert the board, a small resistance has to be overcome.

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



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

Fig. 6-7: Connector keying



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

Note:

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

7 SOFTWARE

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

i

IMPORTANT!

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

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

The board is supplied with a CD-ROM (CD1) containing

- the driver and software samples for Windows NT 4.0 and Windows XP/2000/98,
- the ADDIREG registration program for Windows NT 4.0 and Windows XP/2000/98.

7.1 Board registration with ADDIREG

The ADDIREG registration program is a 32-bit program for Windows NT XP/2000/NT 4.0/ 9x. The user can register all hardware information necessary to operate the ADDI-DATA PC boards.

i

IMPORTANT!

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

7.1.1 Installing a new board

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

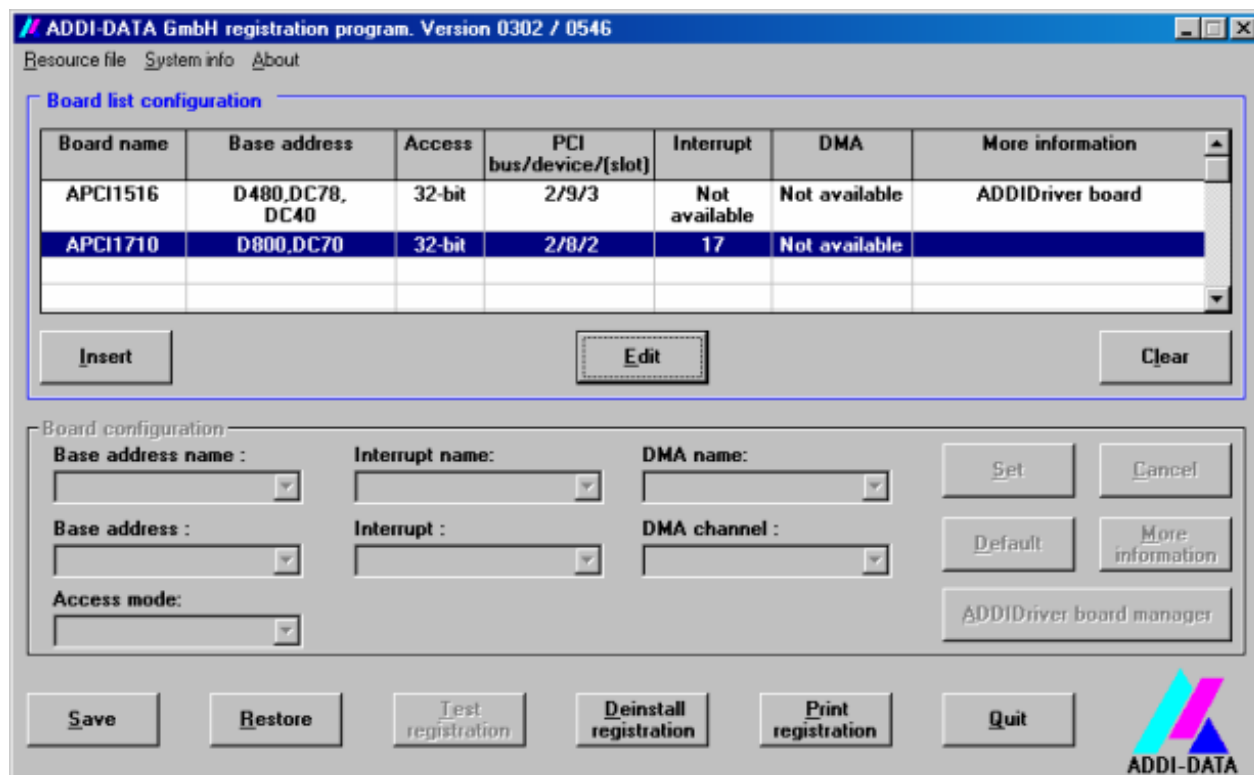


Table:

Board name:

Names of the different registered boards (e.g.: APCI-1710).

Base address:

Selected base address of the board. For PCI boards the base address is allocated through BIOS.

i

WICHTIG!

The base address set in ADDIREG 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 or 32-bit mode.

PCI bus/device/(slot):

Number of the used PCI bus, slot, and device. If the board is no PCI board, the message "NO" is displayed.

Interrupt:

Used interrupt of the board. If the board supports no interrupt, the message "Not available" is displayed. **For PCI boards the interrupt is allocated through BIOS.**

i**WICHTIG!**

The interrupt set in ADDIREG must correspond to the one set through jumper.

ISA DMA (ISA boards only):

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

More information:

Additional information like the identifier string or the installed COM interfaces. It also displays whether the board is programmed with ADDIDRIVER or if a **PCI DMA** memory is allocated to the board.

Text boxes:**Base address name:**

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

Base address:

In this box you can select the base addresses of your PC board. The free base addresses are listed. The used base addresses do not appear in this box.

Interrupt name:

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

Interrupt:

Selection of the interrupt number which the board uses.

DMA name (for ISA boards only):

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

DMA channel (for ISA boards only):

Selection of the used DMA channel.

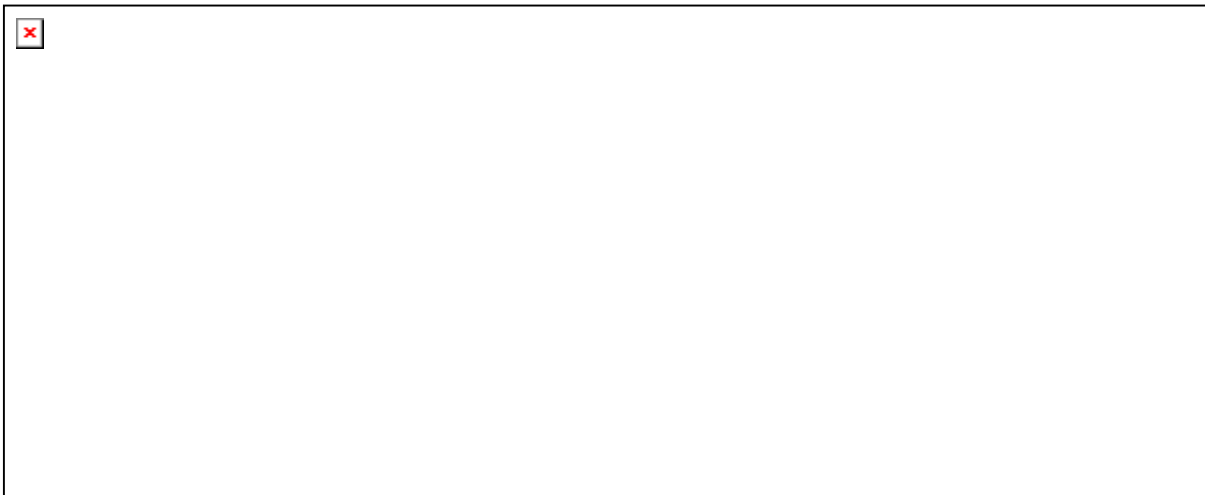
Buttons:**Edit:**

Selection of the highlighted board with the different parameters set in the text boxes.

Insert:

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

Fig. 7-2: Selecting 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 parametered board configuration. The configuration should be set before you save it.

Cancel:

Reactivates the former parameters of the saved configuration.

Default:

Sets the standard parameters of the board.

ADDIDriver Board Manager (only for the boards with ADDIPACK):

Under Edit/ADDIDriver Board Manager you can check or change the current settings of the board set through the ADDEVICE Manager.

ADDevice Manager starts and displays a list of all resources available for the virtual board.

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.

More information (not available for the boards with ADDIPACK)

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

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

**IMPORTANT!**

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

7.1.2 MORE information

You can change the board specific parameters like the identifier string, the COM number, the operating mode of a communication board, etc... If your board does not support these information, you cannot activate this button.

7.1.3 PCI analog input boards with DMA

If you have inserted an APCI-3001 or CPCI-3001 the following dialog box is displayed when clicking on "More information".

Below is the example of 1,000,000 PCI DMA acquisitions (in continuous mode).

For the PCI DMA analog input acquisition, a linear memory buffer of the PC is used. The buffer size depends on the number of acquisitions. For 1 acquisition 2 bytes are needed.

You can define the maximum number of acquisitions used for your application and allocate a large buffer after the PC has started.

If you have selected DMA_USED in the function `i_PCI3001_InitAnalogInputAcquisition` the buffer(s) are used. (See technical description "Standard software")

For a single acquisition, only one buffer is allocated.
 For a continuous acquisition, two buffers are allocated.

Fig. 7-3: PCI DMA management (Example)

System informations

Total real memory	:	200712192
Free memory	:	115949568
Number of available acquisitions	:	57974784
Number of selected acquisitions	:	10000000
Real memory used for PCI DMA	:	40000000

PCI DMA board list

Board name bus/device/(slot)	Number of acquisitions	Acquisition mode	DMA buffer size (bytes)	Status
APCI3120 0/11/2	10000000	Continuous	40000000	Wait PC restart

Single PCI DMA board configuration

Board name : APCI3120 0/11/2

Number of available acquisitions : 28987392

Number of selected acquisitions : Acquisition mode :

System information

Total real memory:

Total real memory of the PC (in bytes).

Free memory:

Returns the PC memory (in bytes) available for PCI DMA acquisition.

Number of available acquisitions:

Returns the number of acquisitions which can be carried out in the single mode.

Number of selected acquisitions:

Returns the number of acquisitions selected by the user.

Real memory used for PCI DMA:

Returns the memory size (in bytes) used for the PCI DMA acquisition.

PCI DMA board list

List of all PCI boards which can use the PCI DMA analog input acquisition.

For each board the user can select the number of acquisitions and the acquisition mode (single/continuous).

Board name:

Indicates the board name, the bus number, the device number and the slot number.

Number of acquisitions:

Number of acquisitions selected by the user.

Acquisition mode:

Acquisition mode selected by the user (single or continuous).

DMA buffer size (in bytes):

Size of the buffer used for this configuration.

Status:

Not used: The number of acquisitions selected by the user is equal to 0

Wait PC restart: Wait until the PC restarts to allocate the memory

Allocation OK: Buffer allocation OK

Allocation error: Buffer allocation error. The driver could not allocate a linear memory buffer for this acquisition.

Buttons

Edit:

Selection of the highlighted board with the different parameters set in the boxes of "Single PCI DMA board configuration". (See below)

Save:

Saves the configuration of all boards.

Quit:

Closes this window.

Single PCI DMA board configuration:

After selecting a board, click on Edit: the selected configuration of the board with PCI DMA is displayed in the "Single PCI DMA board configuration" box.

Board name:

Indicates the board name, the bus number, the device number and the slot number.

Number of available acquisitions:

Indicates the number of acquisitions available **for the selected mode** (acquisition mode) and **for the next board** to be configured.

Number of selected acquisitions:

Number of acquisitions selected by the user ("Not used" means that no buffer is allocated for PCI DMA acquisition).

**IMPORTANT!**

You have to enter an **even number**.

An odd number of acquisitions will not be accepted and automatically replaced by the approaching even number.

Acquisition mode:

Acquisition mode selected the user:

Single: Only one acquisition cycle is used. After this cycle the acquisition is immediately stopped.

Continuous: The acquisition runs until the function `i_PCI3001_StopAnalogInputAcquisition` is called up.

Set:

Sets the user configuration.

Cancel:

Restores the former configuration

7.1.4 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.**

Fig. 7-1 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.

7.1.5 Changing the registration of a board

i

IMPORTANT!

To change the registration of a board, you must have administrator rights.

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

◆ **Call up the ADDIREG program.**

◆ **Select the board to be changed.**

The board parameters (Base address, DMA channel, ..) are listed in the lower fields.

◆ **Click on the parameter(s) you want to set and open the scroll function(s).**

◆ **Select a new value.**

◆ **Activate it with a click. Repeat the operation for each parameter to be modified.**

◆ **Once the wished configuration is set, click on "Set".**

◆ **Save the configuration with "Save".**

You can test if the registration is "OK".

This test controls if the registration is right and if the board is present.

If the test has been successfully completed you can quit the ADDIREG program.

The board is initialised with the set parameters and can now be operated.

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

7.2 Questions and software downloads on the web

Do not hesitate to e-mail us your questions.

per e-mail: info@addi-data.de or
 hotline@addi-data.de

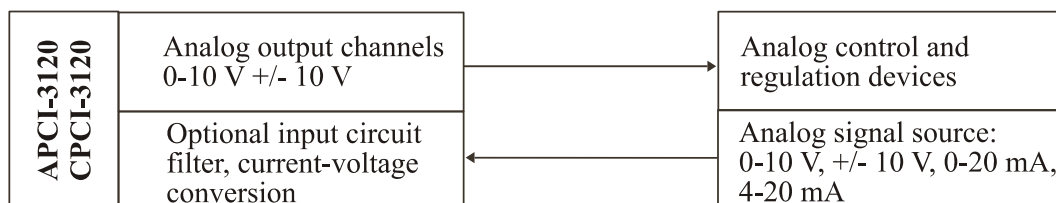
Free downloads of standard software

You can download the latest version of the software for the board **APCI-/CPCI-3120**

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

8 CONNECTING THE PERIPHERAL

8.1 Connection principle



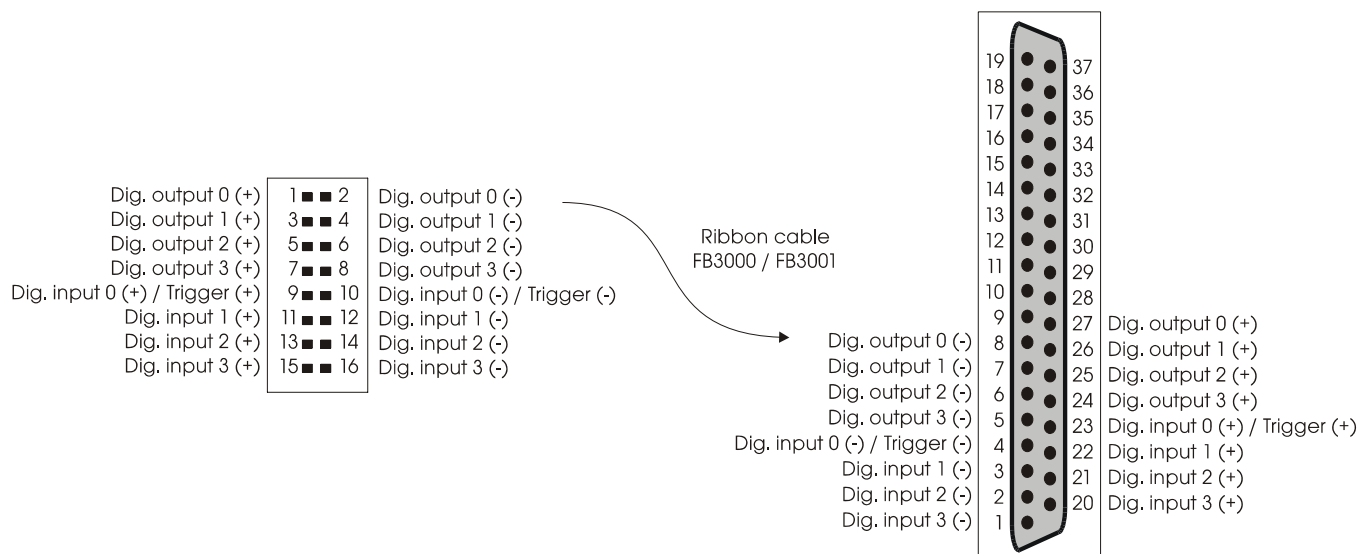
8.2 Connector pin assignment

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

DIFF		SE		SE		DIFF	
(+) An. input 0	(+) An. input 0	20	1	(+) An. input 8	(+) An. input 4		
(+) An. input 1	(+) An. input 1	21	2	(+) An. input 9	(+) An. input 5		
(+) An. input 2	(+) An. input 2	22	3	(+) An. input 10	(+) An. input 6		
(+) An. input 3	(+) An. input 3	23	4	(+) An. input 11	(+) An. input 7		
(-) An. input 3	(+) An. input 7	24	5	(+) An. input 15	(-) An. input 7		
(-) An. input 2	(+) An. input 6	25	6	(+) An. input 14	(-) An. input 6		
(-) An. input 1	(+) An. input 5	26	7	(+) An. input 13	(-) An. input 5		
(-) An. input 0	(+) An. input 4	27	8	(+) An. input 12	(-) An. input 4		
1 {	Analog input GND	28	9	Analog input GND			
	Analog input GND	29	10				
2 {	An. output 0 GND	30	11			An. output 0	
	An. output 1 GND	31	12				
	An. output 2 GND	32	13				
	An. output 3 GND	33	14				
	An. output 4 GND	34	15				
	An. output 5 GND	35	16				
	An. output 6 GND	36	17				
	An. output 7 GND	37	18	An. output 6			
			19				
				An. output 7			

1. The analog input channels have a common ground line.
2. The analog output channels have separated ground lines.

Fig. 8-2: 16-pin ribbon cable connected to 37-pin SUB-D male connector



8.3 Connection examples

8.3.1 Analog inputs channels

Fig. 8-3: Analog input channels (SE)

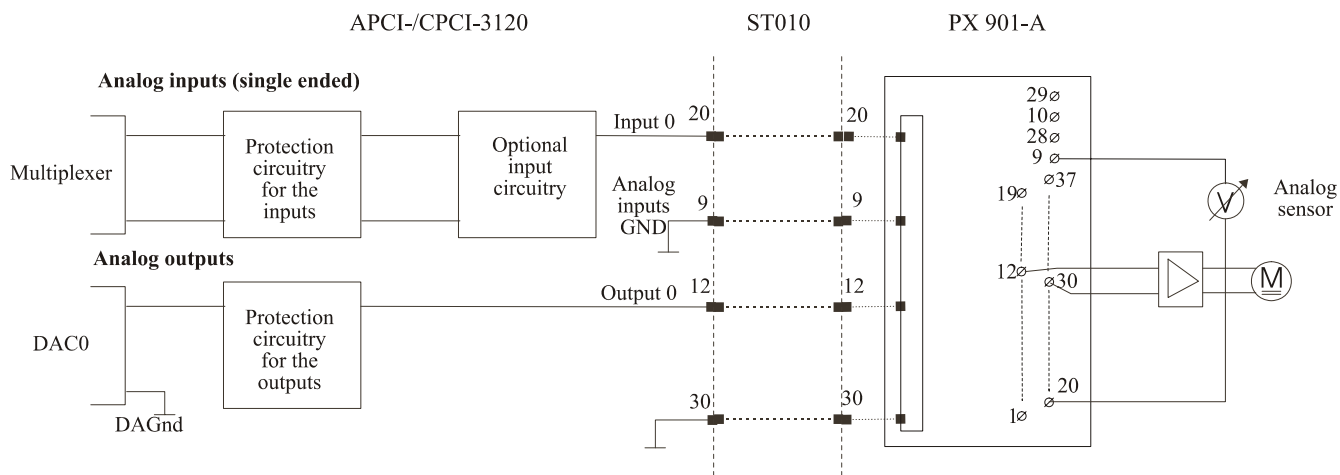
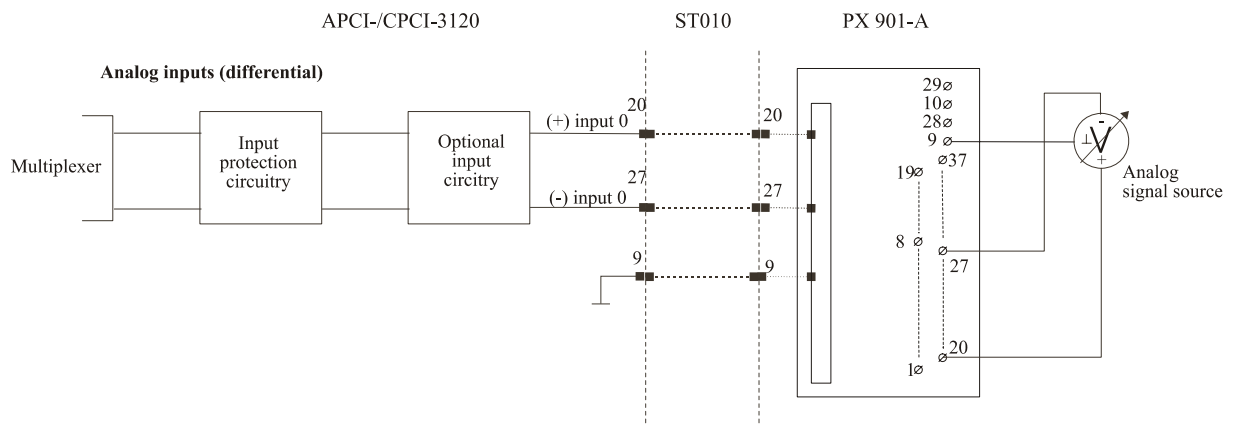


Fig. 8-4: Analog input channels (Diff)



8.3.2 Digital input and output channels

Fig. 8-5: Digital input channels

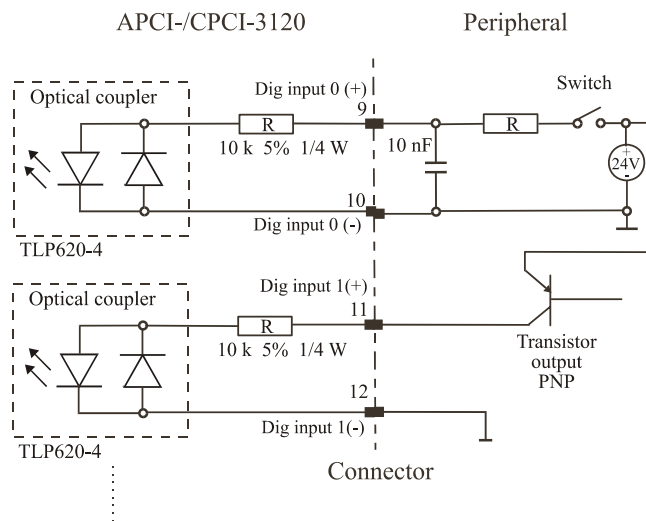
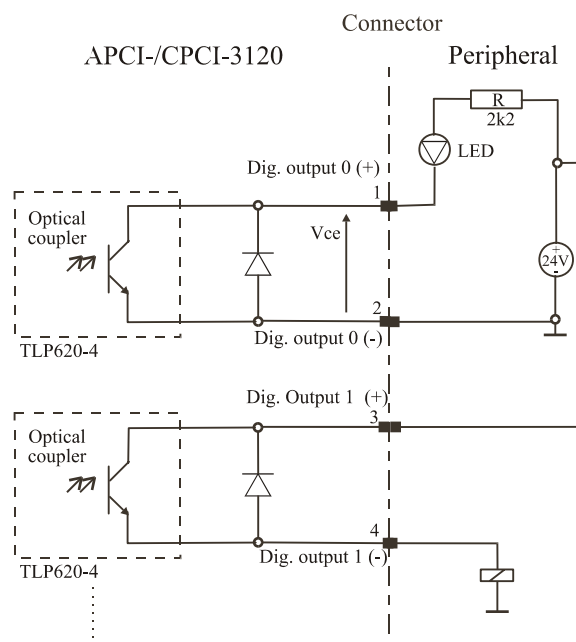
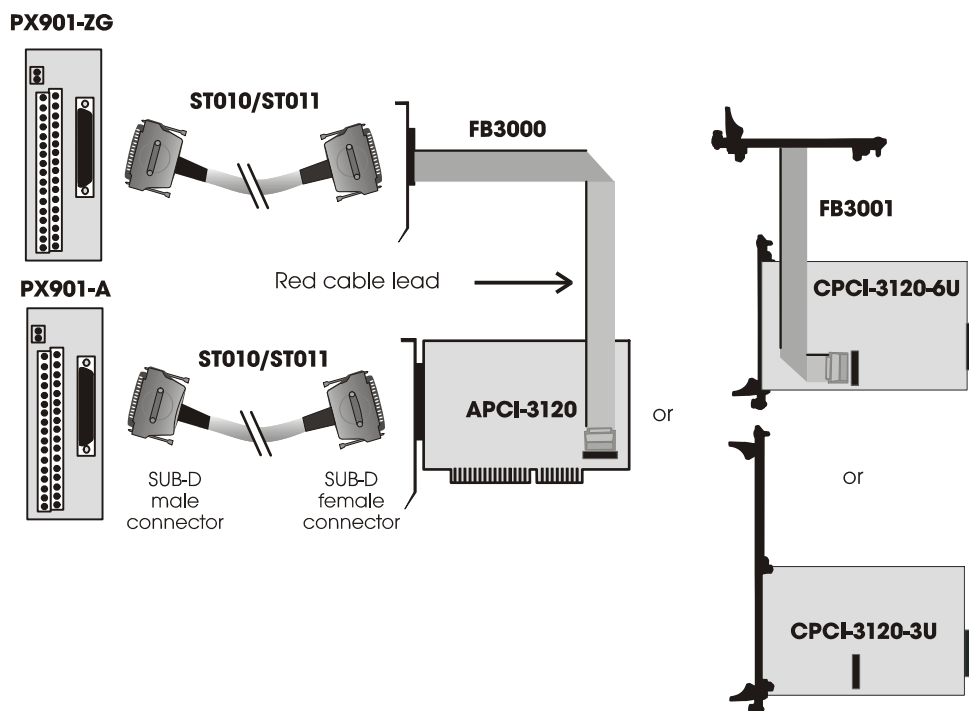


Fig. 8-6: Digital output channels



8.3.3 Connection to the PX 901 screw terminal board

Fig. 8-7: Connection to the screw terminal board PX901



i

IMPORTANT!

Insert the FB3000/FB3001 on the connector with the red cable lead on the side of the pin 1.

9 FUNCTIONS OF THE BOARD

9.1 Analog output channels

There are up to 8 analog output channels on the board.

The analog output channels are set to a 0 V voltage value after Power-ON Reset of the PC.

The analog output channels are updated:

through 16-bit write operations on I/O addresses

A bit of the status register indicates if all the analog output channels are ready to be updated.

A watchdog function is available for the analog output channels (through Timer2)

Timer2 must be programmed as a retriggerable watchdog.

Triggering occurs through a write operation on the analog output channels.

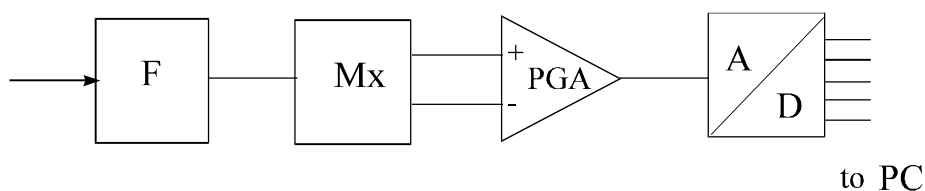
The output channels are not reset as long as writings occur on the analog output channels.

The watchdog circuit can also generate interrupts.

The state of the watchdog is indicated in the *Status Register*.

9.2 ANALOG input channels

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



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

The analog input voltage range (0-10 V; ± 10 V) and the gain

can be configured through software.

It is thus possible

- to have for each channel a different voltage (or current)
- and to use the best resolution of the A/D converter.

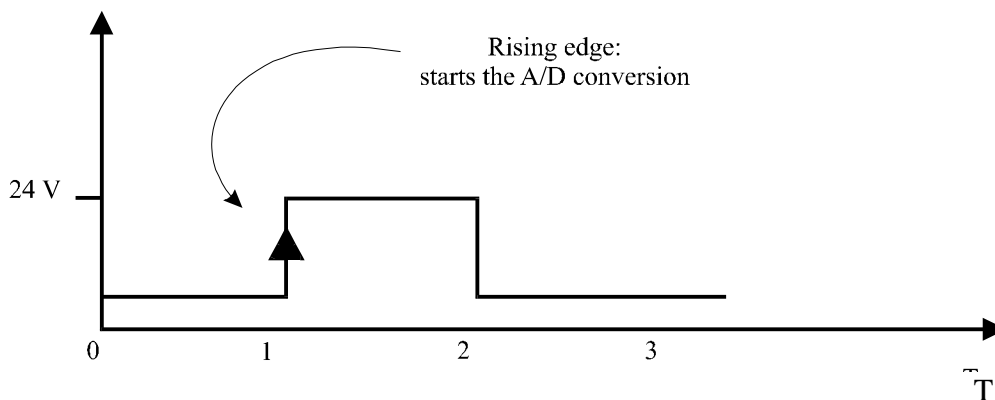
Scan lists (list of the channel features)

These lists can be located in a 16-byte deep RAM.

They allow more flexible data acquisitions. You determine their depth by software.

Possible acquisition of the **scan lists**:

- one channel after the other by software trigger (each channel must be triggered),
- the list of the channels is processed once by software trigger,
- through Timer0: the list can be cyclically handled,
- through Timer0 & 1: the list can be handled while a defined time interval.
- through Timer0, 1 & 2: a definite number of conversions or scans is processed
- The programmed AD conversion is started by a 24 V signal via the input 0 of the 4 digital input channels.



Data exchanges to the PC occur through a 256 word-deep **FIFO**.

- Polling is possible, analysis of the EOC and EOS bit
- Interrupt at EOC, EOS, END OF DMA,
- DMA mode at EOC
- Data is partly loaded (according to the type of conversion) in the FIFO.

9.3 Time-multiplex system

Data acquisition with the **APCI-/CPCI-3120** is based on a time-multiplex system. The board is equipped with a single A/D converter to which the channels are led through an analog multiplexer (software and hardware controlled).

The programmable gain amplifier is highly resistive. It is equipped with a capacitive line from the output channel of the multiplexer to the input channel of the INA, so that each channel is protected by a low-pass filter (RC module).

By converting a channel into one another, the output capacity of the multiplexer is to be converted in the new value.

There is a certain delay between the channel conversion and the start of the A/D converter.

This time delay corresponds to the settling time of an end value. This value depends on the resolution of the acquisition. (e.g.: 0.01 % at 12-bit; 0.0008 % at 14-bit).

The time delay depends on the following factors:

- the switching time of the amplifier, about 3.5 μ s.
- the maximum voltage bounce from a channel to one another.
- the source impedance of the sensory mechanism.
- Option SF = 10 K // 470 nF Fc/-3 dB ca. 30 Hz
- Option DF = 20 K // 470 nF Fc/-3 dB ca. 30 Hz

The delay is supported on the **APCI-/CPCI-3120** board by the 16-bit Timer 0.

You can set this time in steps of 0.5 μ s from 10 μ s to 32767 μ s. In the delivered API functions the delay time is the parameter *ui_ConvertTiming*. (See documentation: Device driver).

With the scan list the next channel can already be switched on during the current conversion (Duration: 10 μ s). It enables to reach the maximum conversion rate of 100 kHz on several channels with low-impedant sensors.

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

When the data acquisition is run by software control (direct conversion) the following table applies:

Rsource	ui_ConvertTiming
<100R	10
< 500R	30..60
< 1K	500..700
< 10K	1000..2000
< 50K	10000...65535

When the data acquisition is run by hardware control (cyclic conversion) the user has to consider the A/D conversion time.

Rsource	ui_ConvertTiming
<100R	10
< 500R	30..80
< 1K	500.. 700
< 10K	1000..2000
< 50K	10000..65535

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

10 CALIBRATION TOOL

10.1 Introduction

10.1.1 General description

10.1.2 Requirements

You require the following components to use the calibration tool:

Table 10-1: Required components

Components included in the standard delivery	Components not included in the standard delivery
Calibration tool software	Calibration device that permits to deliver -0.61millivolt / 5.00 volt and 9.995 volt with a precision from minimal 0.5 millivolt.
Board installation files (for Windows 98/Windows 2000/Windows XP).	Multimeter with 1 mV precision
Cable to connect the APCI-3120 analogue input 0 to the calibration device	
Cable to connect the APCI-3120 analogue output 0 to the digital multimeter	

10.2 Installation of the calibration tool

Run the file called Setup.exe and follow the instructions given on the screen.

The calibration tool is (as default) installed under: *C:\Programme\ADDI-CALIB APCI-3120*

You will find a shortcut to the calibration tool in the “Start” menu of Windows.

10.2.1 Remark for Windows 98/ Windows 2000 / Windows XP

When you insert the APCI-3120 in a computer and start it under Windows 98, Windows 2000 or Windows XP, the hardware manager will detect the board and request installation files. These installations files are located in the inf-folder. Just select the inf -folder directory, it will take automatically the required file to install the **APCI-3120**.

This operation is not required under Windows NT.

10.2.2 Board registration

After installing the required file, you have to register the board with ADDIREG program. If the board is not registered you will not be able to use the calibration tool. To register board follow the instructions in the technical manual of the board.

10.3 Board preparation

10.3.1 Calibration of the analogue input

Calibration device

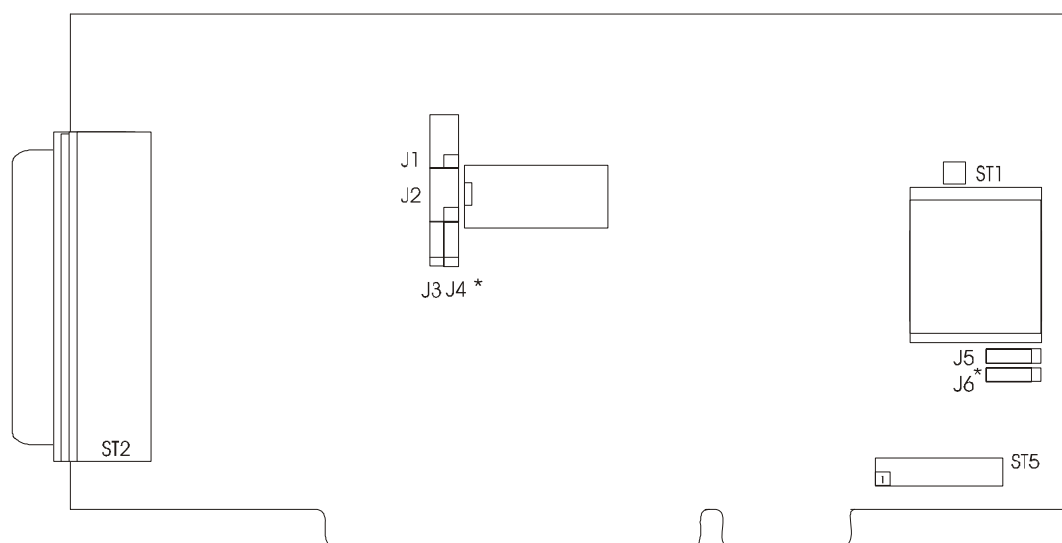
The calibration device must have at least a precision of 0.5 millivolt and must be able to deliver -0.61millivolt / 5.00 volt and 9.995 volt.

Recommended devices:

- Burster DIGISTANT Type 6705
- Burster DIGISTANT Type 4462.
- Fluke 5700
- Fluke 5720

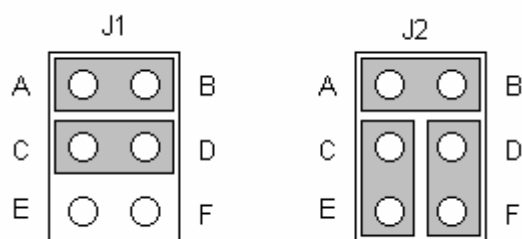
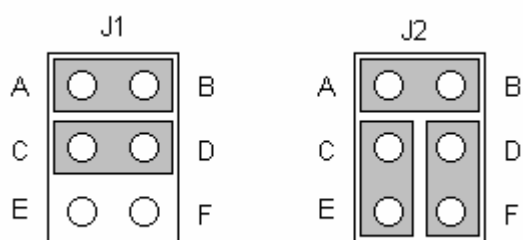
Analogue input in single or differential mode

Fig. 10-1: Jumper location



* J3, J4, J5, J6, ST1: No function

The calibration can be done either in single or differential mode:

Fig. 10-2: Configuration in the single mode**Fig. 10-3: Configuration in the differential mode**

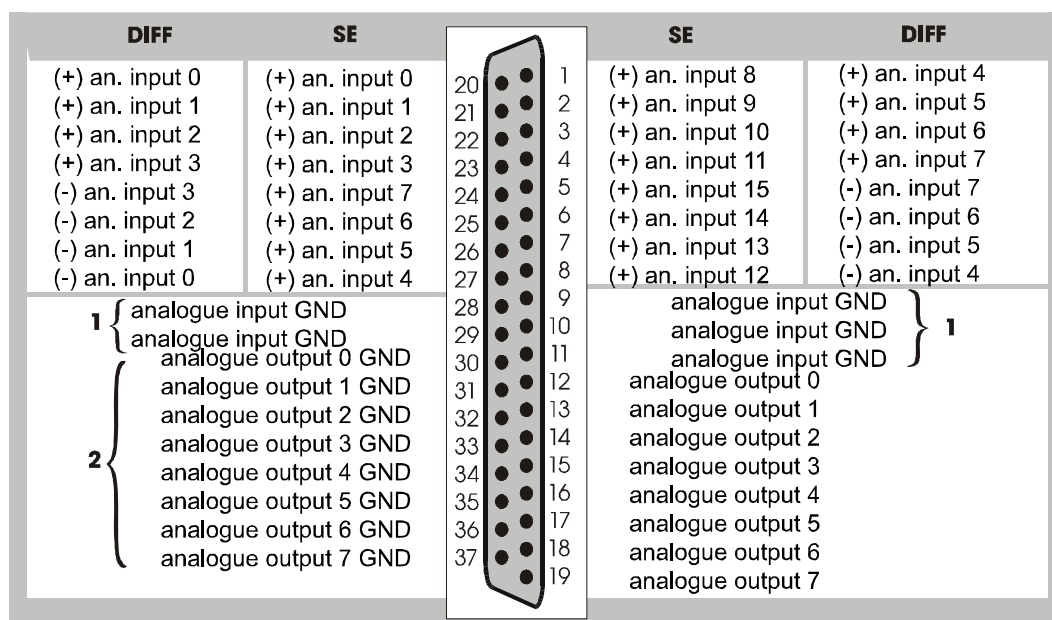
Calibration device and board connection

The analogue input 0 has to be connected to the calibration device.

Table 10-2: Calibration device and board connection

In the single mode	In the differential mode
Plus (+) from the board (pin 20) to calibration device plus (+)	Plus (+) from the board (pin 20) to calibration device plus (+).
GND from the board (pin 9, 10, 11, 28, 29) to calibration device GND	Minus from the board (pin 27) to calibration device GND

Fig. 10-4: 37-pin SUB D front connector



SE = Single mode Diff = Differential mode

10.3.2 Analogue input calibration

Digital multimeter

The digital multimeter must have at least a precision from ± 1 millivolt

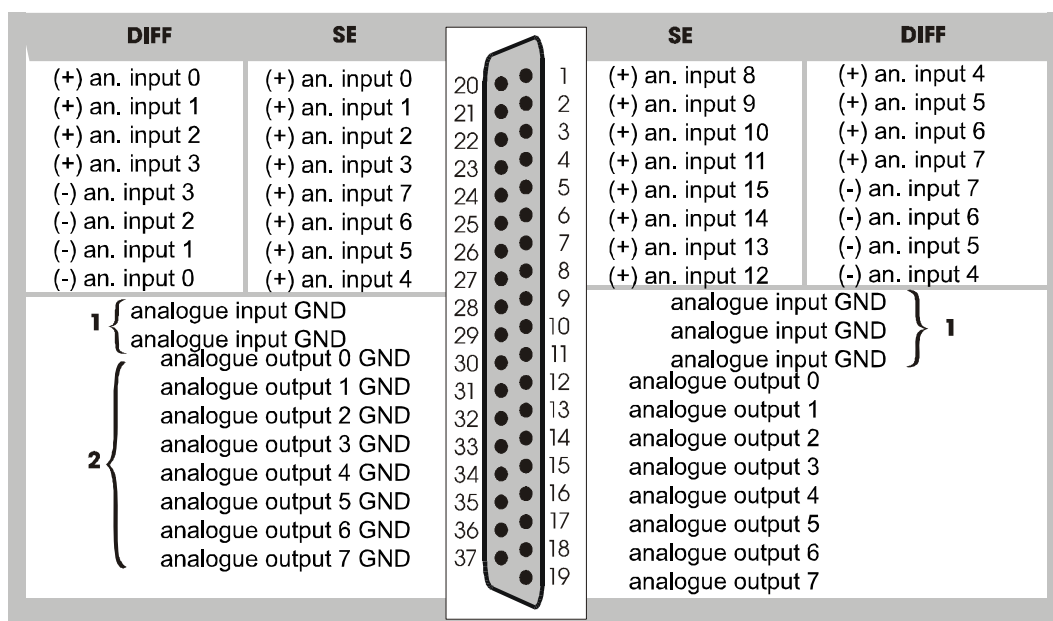
Recommended models:

- Keithley 2000 multimeter
- Fluke 8840A multimeter

Digital multimeter and board connection

The analogue Output 0 from the board has to be connected to the digital multimeter. Pin 12 from the board must be connected to the plus from the multimeter and the pin 30 to the multimeter GND.

Fig. 10-5: 37-pin SUB-D connector

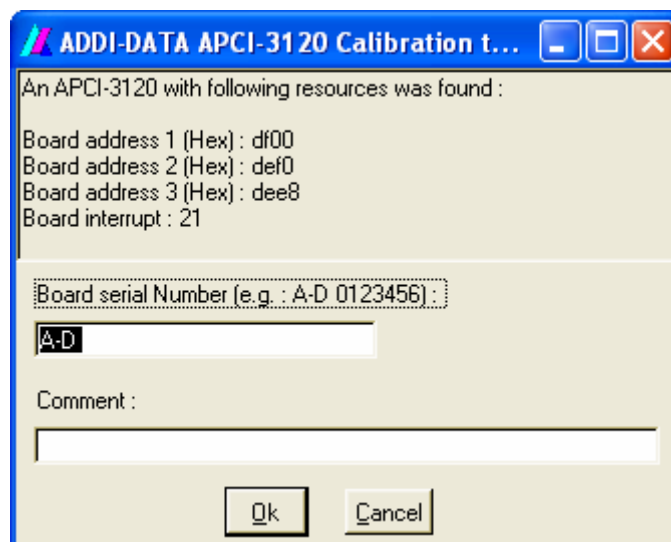


10.4 Using the calibration tool

10.4.1 Board resources and information

Once the board is prepared you can start the calibration tool. The following window will appear:

Fig. 10-6: Window: Starting the calibration tool



If the board is found:

1. Board resources are displayed.
2. Enter the board serial number in order to identify it. It will be used as test report name. If you do not have board serial number you may enter another file name that you have chosen like "Cal_Test_01".
3. You can enter a comment which will be saved in the test report. This comment is not necessary.

◆ Click "Ok" to continue

◆ Click "Cancel" to quit the program

If the board is found but not registered or several boards are inserted in the computer:

A message informs you that a board has been found but this device is not registered in ADDIREG or several boards are inserted in the computer.

◆ Please register the board with ADDIREG and make sure that just one board is inserted in the computer.

i**IMPORTANT!**

The program cannot work if more than one board is inserted in the computer!

You can quit the program as follows:

◆ Click "Cancel" or

◆ Close the window and register the board with ADDIREG.

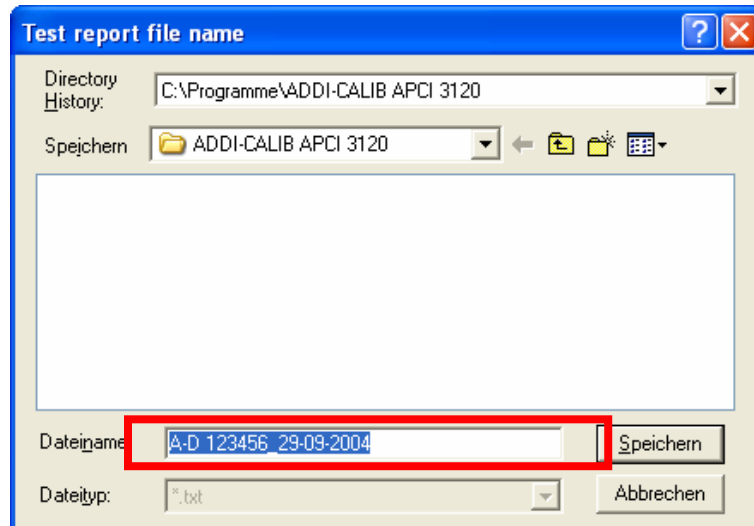
If the board is not found:

A message informs you that the board has not been found. You can quit the program by clicking "Cancel" or by closing the window.

10.4.2 Define the test report file and directory

After clicking “OK” the following dialog window appears:

Fig. 10-7: Window: Test report file name



As you can see, the program proposes you a file name made of the serial number entered before and of the system date. If you not agree with this proposition, you can change the directory and the file name.

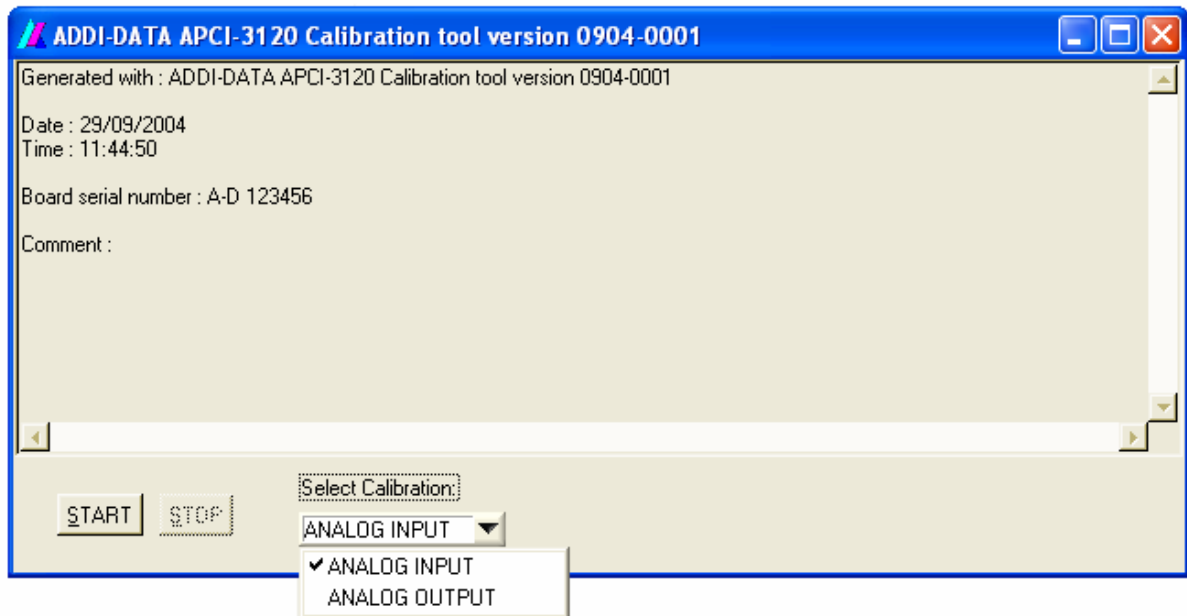
Click on “Save” (“Speichern”) when you have selected the correct directory and file name.

Remark:

- If you do not click on “Save” (“Speichern”), the program will give the generated default name.
- The report file will be generated when you close the program. Before this no report will be generated.

10.5 Calibration

Fig. 10-8: Window: Calibration tool version



1. This text box displays a part of the future contents of the test report.
2. Now you have to select the type of calibration. You have the possibility to calibrate analogue inputs or analogue outputs.

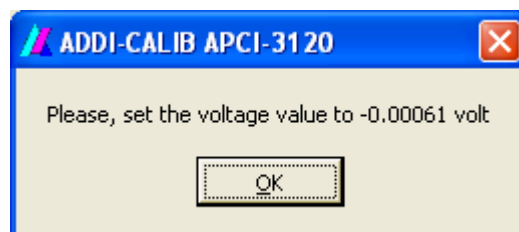
Click on “START” to start the calibration after having selected the calibration type.

10.5.1 Analogue input calibration

Calibration of bipolar offset on potentiometer 0 with -0.00061 volt

You must configure the calibration device with the specified voltage value (in this case -0.00061 volt):

Fig. 10-9: Window: Voltage value -0.00061



Click “OK” when the supply voltage is configured on analogue input 0 with the specified value.

A progress bar will appear and show you the acquisition progression. Under the progress bar the following text is displayed: Acquisition in progress (Attempt: 1/256).

Attempt: 1/256: For each potentiometer, the program will run a maximum of 256 calibration probes. If more than 256 probes are necessary, an error will be generated. In this case the board is not calibrated.

If the calibration is correct, you will see the following line in the text box:

PASS: Calibration of bipolar offset on potentiometer 0 with -0.00061 V

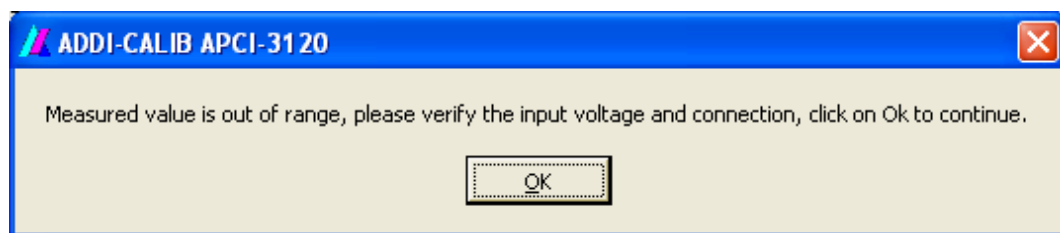
If the calibration is wrong, you will see the following line in the text box:

FAIL: Calibration of bipolar offset on potentiometer 0 with -0.00061 V

Remark:

If the voltage measured by the board is different by ± 0.150 V from the asking voltage, the following message will appear:

Fig. 10-10: Window: Measured value is out of range



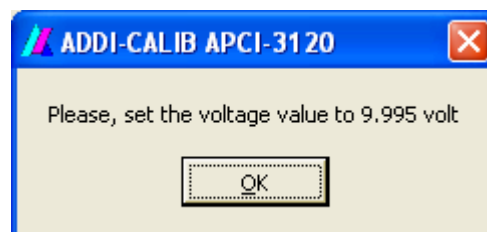
◆ Click on “OK” to verify the connection and voltage configuration

If the problem remains, the calibration will be stopped and an error will be generated.

Calibration of bipolar gain on potentiometer 1 with 9.995 volt

You have to configure the calibration device with the specified voltage value (in this case 9.995 volt):

Fig. 10-11: Window: Voltage value 9.995 volt



Click on “OK” when the supply voltage is configured on analogue input 0 with the specified value.

A progress bar will appear and show you the acquisition progression. Please wait during the calibration process.

If the calibration is correct, you will see the following line in the text box:

PASS: Calibration of bipolar gain on potentiometer 1 with 9.995 volt.

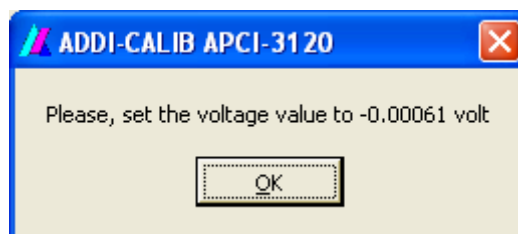
If the calibration is wrong, you will see the following line in the text box:

FAIL: Calibration of bipolar gain on potentiometer 1 with 9.995 volt.

Calibration of bipolar offset on potentiometer 0 with -0.00061 volt

You have to configure the calibration device with the specified voltage value (in this case -0.00061 volt):

Fig. 10-12: Window: Voltage value -0.00061 volt



Once the supply voltage is configured on analogue input 0 with the specified value, click "OK".

A progress bar will appear and show you the acquisition progression. Please wait during calibration.

If the calibration is correct, you will see the following line in the text box:

PASS: Calibration of bipolar offset on potentiometer 0 with -0.00061 volt

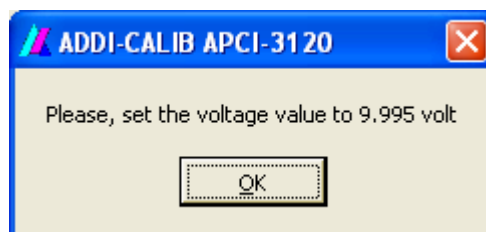
If the calibration is wrong, you will see the following line in the text box:

FAIL: Calibration of bipolar offset on potentiometer 0 with -0.00061 volt

Calibration of bipolar gain on potentiometer 1 with 9.995 volt

You have to configure the calibration device with the specified voltage value (in this case 9.995 volt):

Fig. 10-13: Window: Voltage value 9.995 volt



Once the supply voltage is configured on analogue input 0 with the specified value, click “OK”.

A progress bar will appear and show you the acquisition progression. Please wait during calibration.

If the calibration is correct, you will see the following line in the text box:

PASS: Calibration of bipolar gain on potentiometer 1 with 9.995 volt

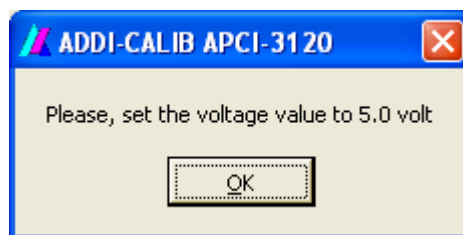
If the calibration is wrong, you will see the following line in the text box:

FAIL: Calibration of bipolar gain on potentiometer 1 with 9.995 volt

Calibration of unipolar offset on potentiometer 2 with 5.00 volt

You have to configure the calibration device with the specified voltage value (in this case 5.0 volt):

Fig. 10-14: Window: Voltage value 5.0 volt



Once the supply voltage is configured on analogue input 0 with the specified value, click “OK”.

A progress bar will appear and show you the acquisition progression. Please wait during calibration.

If the calibration is correct, you will see the following line in the text box:

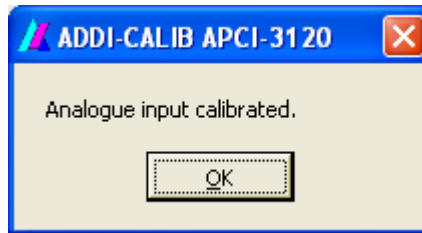
PASS: Calibration of unipolar offset on potentiometer 2 with 5.00 volt

If the calibration is wrong, you will see the following line in the text box:

FAIL: Calibration of unipolar offset on potentiometer 2 with 5.00 volt

Analogue input calibrated

If the analogue input is calibrated, you will see the following message at the end of the sequence:

Fig. 10-15: Window: Analogue input is calibrated

◆ Click on “OK”

All information about the calibration will be displayed in the text box. This information will be saved in the file that you have defined at the beginning when you close the program.

The following text is a report sample.

Table 10-3: Report sample

Generated with: ADDI-DATA APCI-3120 Calibration tool 0904-0001

Date: 29/09/2004

Time: 17:26:33

Board serial number: A-D 123456

Comment: APCI-3120

ANALOGUE INPUT CALIBRATION:

PASS : Calibration of bipolar offset on potentiometer 0 with -0.00061 volt

PASS : Calibration of bipolar gain on potentiometer 1 with 9.995 volt

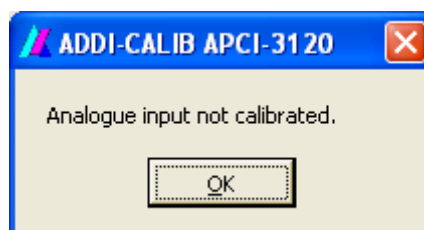
PASS : Calibration of bipolar offset on potentiometer 0 with -0.00061 volt

PASS : Calibration of bipolar gain on potentiometer 1 with 9.995 volt

PASS : Calibration of unipolar offset on potentiometer 2 with 5.0 volt

Analogue input calibrated.

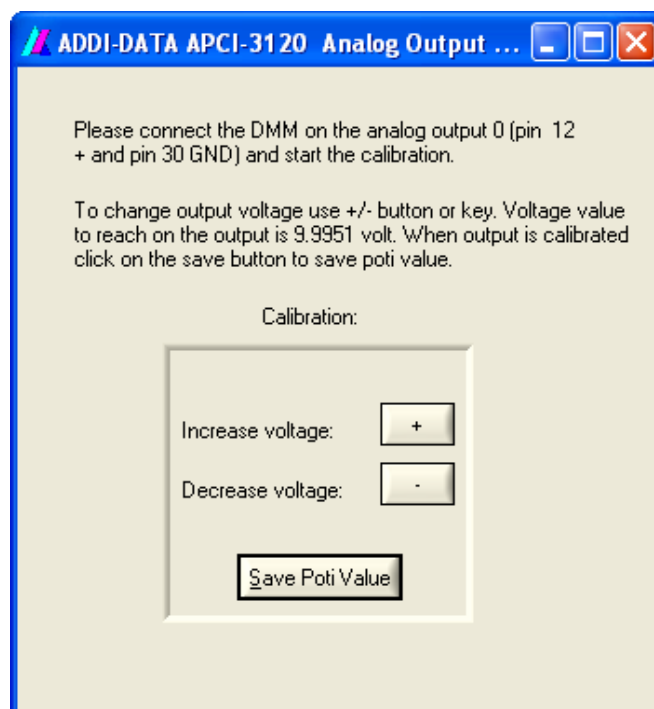
If the board is not calibrated, you will see the following message at the end of the sequence:

Fig. 10-16: Window: Analogue input is not calibrated

“Error” will be displayed in the text box and written in the test report.

10.5.2 Analogue output calibration

Fig. 10-17: Window: Analogue output calibration

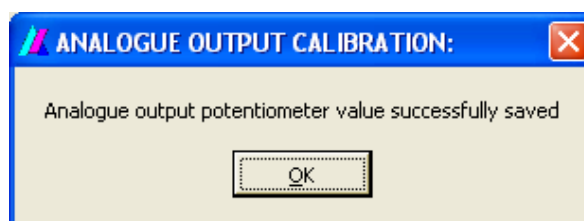


When you have selected the analogue output calibration, the window above will appear.

If you have connected the digital multimeter to the analogue output 0 you must read a voltage between 9.98 and 10.03 volt on the multimeter. If not, make sure that you have connected the plus (+) of the digital multimeter to Pin 12 of the board and the minus (-) of the digital multimeter to Pin 30 of the board.

If you read a correct voltage you can start to calibrate the board. To change the value of the output voltage click on the +/- button or press +/- key. + button/key enables to increase voltage and – button/key enables to decrease voltage. Output voltage must change by step of 0.00015 volt. Change output voltage until you have reached 9.9951 v. You cannot exactly reach 9.9951 v, there can be a failure of +/- 0.00015 volt on the output. After having calibrated the board you can save the potentiometer value on clicking “Save Poti Value”. If the potentiometer value has been saved successfully this message will appear:

Fig. 10-18: Window: Analogue output potentiometer successfully saved



All information about the calibration will be displayed in the text box. This information will be saved in the file that you have defined at the beginning when you close the program.

The following text is a report sample:

Table 10-4: Report sample

Generated with: ADDI-DATA APCI-3120 Calibration tool 0904-0001

Date: 29/09/2004

Time: 17:26:33

Board serial number: A-D 123456

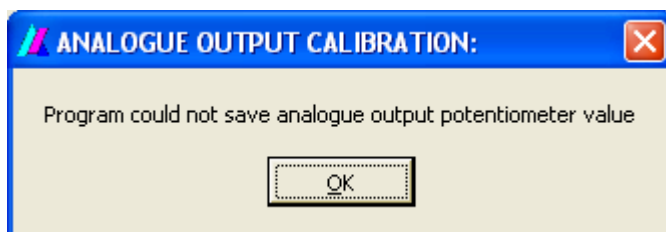
Comment: APCI-3120

ANALOGUE OUTPUT CALIBRATION:

PASS: Analogue output potentiometer value successfully saved.

If the potentiometer value could not be saved, the window below will appear:

Fig. 10-19: Window: The potentiometer value could not be saved



The error will be displayed in the text box and written in the test report.

10.6 Error messages

10.6.1 Possible error messages

In the following possible error messages are mentioned together with their possible reasons:

Calibration tool already started, only one session can be started

Possible reason:

- You tried to open the calibration tool several times. But only one session can run at the same time.

APCI-3120 not found

Click on “cancel” to stop the program when this message appears.
The calibration tool did not find the **APCI-3120**.

Possible reason:

- Board not present in the computer.
- The slot where the board is located is not activated or is defective.

An APCI-3120 was found but device not registered in ADDIREG or several boards are inserted in the computer

The calibration tool found one board but could not access it.

Possible reason:

- Board not registered in ADDIREG.
- There is more than one **APCI-3120** inserted in the computer. Program works with one board in the computer. If you want to use the program you have to remove the other boards.

Fail to close the driver

Possible reason:

- Problems with the board or the driver.

Analogue input not calibrated

Possible reason:

- An error occurred during calibration. See other error message.

Program could not save analogue output potentiometer value

Possible reason:

- An error occurred while saving the potentiometer value. See other error message.

Fail to write the test report: Error type

Possible reason:

- See error type message.

Calibration error: The measured value is out of limits

Before calibrating the board, the program runs a first measurement process with a tolerance of ± 0.150 volt in order to test if there is no connection error.

Possible reasons:

- Wrong input voltage.
- Connection between the analogue input 0 and the power supply is wrong.
- Board defective.

Calibration error: Read analogue input value error

Fail to read analogue value.

Possible reasons:

- Connection problem.
- Board resources are false.
- Board defective.

Calibration error: Too many attempts to calibrate the board

The program tries to calibrate a potentiometer at most 256 times to calibrate it. If more probes are needed an error occurs.

Possible reasons:

- Connection problem.
- Board resources are false.
- Board defective.

FAIL: Calibration of bipolar/unipolar gain/offset on potentiometer 0/1/2 with x volt

The indicated potentiometer cannot be calibrated. See other error messages to find out the reason.

11 SOFTWARE EXAMPLES



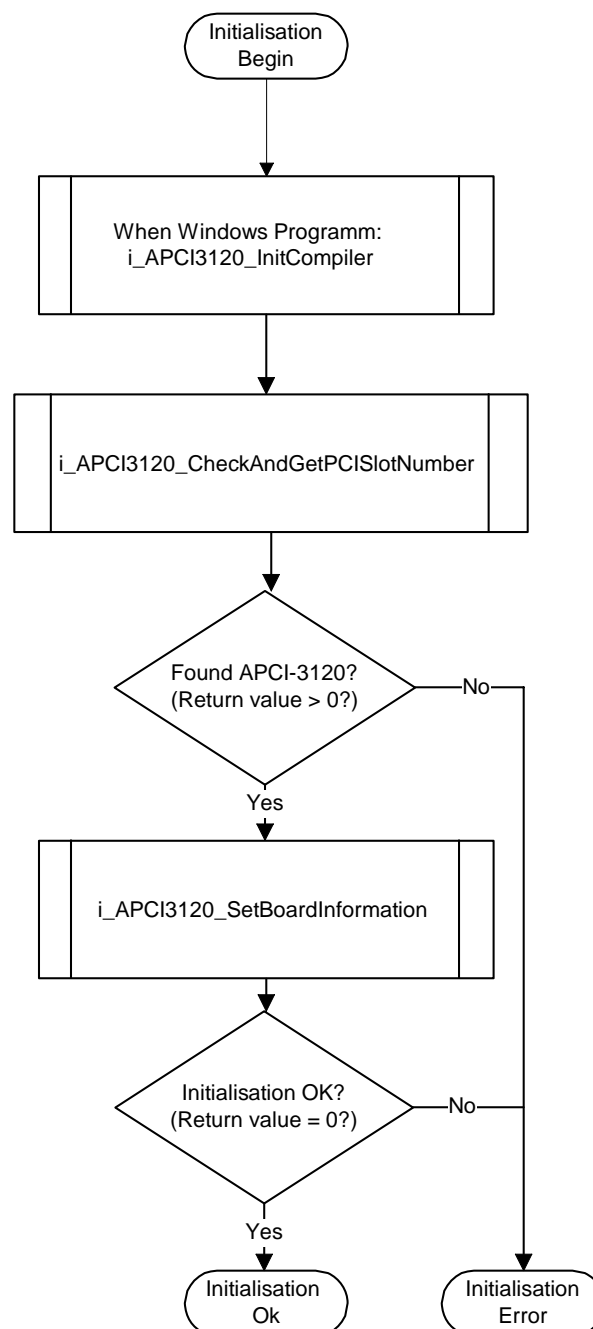
IMPORTANT!

These **examples** are not the functions of a real-time application. They only represent the **possible functions** which can be processed with the board CPCI-3120.

11.1 Initialisation

11.1.1 Initialisation of an APCI-/CPCI-3120 board

a) Flow chart



b) Example in C

```

int Initialisation(unsigned char *pb_BoardHandle)
{
    unsigned char b_SlotNumberArray [8];

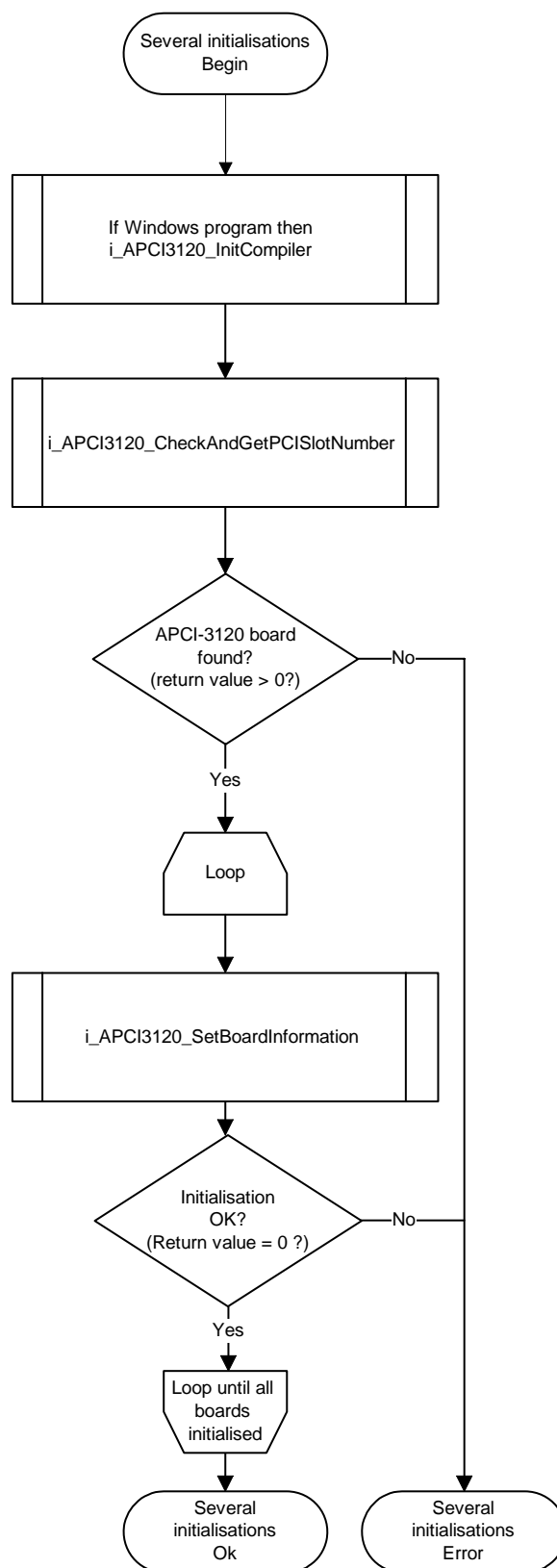
    #ifdef _Windows
        i_APCI3120_InitCompiler (DLL_COMPILER_C);
    #endif

    if(i_PCI3120_CheckAndGetPCISlotNumber (b_SlotNumberArray))
    {
        if(i_APCI3120_SetBoardInformation (b_SlotNumberArray[0],
                                           16,
                                           8,
                                           pb_BoardHandle) == 0)
        {
            return (0);      /* OK */
        }
        else
        {
            return (-1);     /* ERROR */
        }
    }
    else
    {
        return (-1);        /* ERROR */
    }
}

```


11.1.2 Initialisation of several APCI-/CPCI-3120 boards

a) Flow chart



b) Example in C

```
int MoreInitialisation(unsigned char *pb_BoardHandleArray)
{
    int          i_NbrOfBoard;
    int          i_Cpt;
    unsigned char b_SlotNumberArray [8];

    #ifdef _Windows
        i_APCI3120_InitCompiler (DLL_COMPILER_C);
    #endif

    i_NbrOfBoard = i_PCI3120_CheckAndGetPCISlotNumber (b_SlotNumberArray)

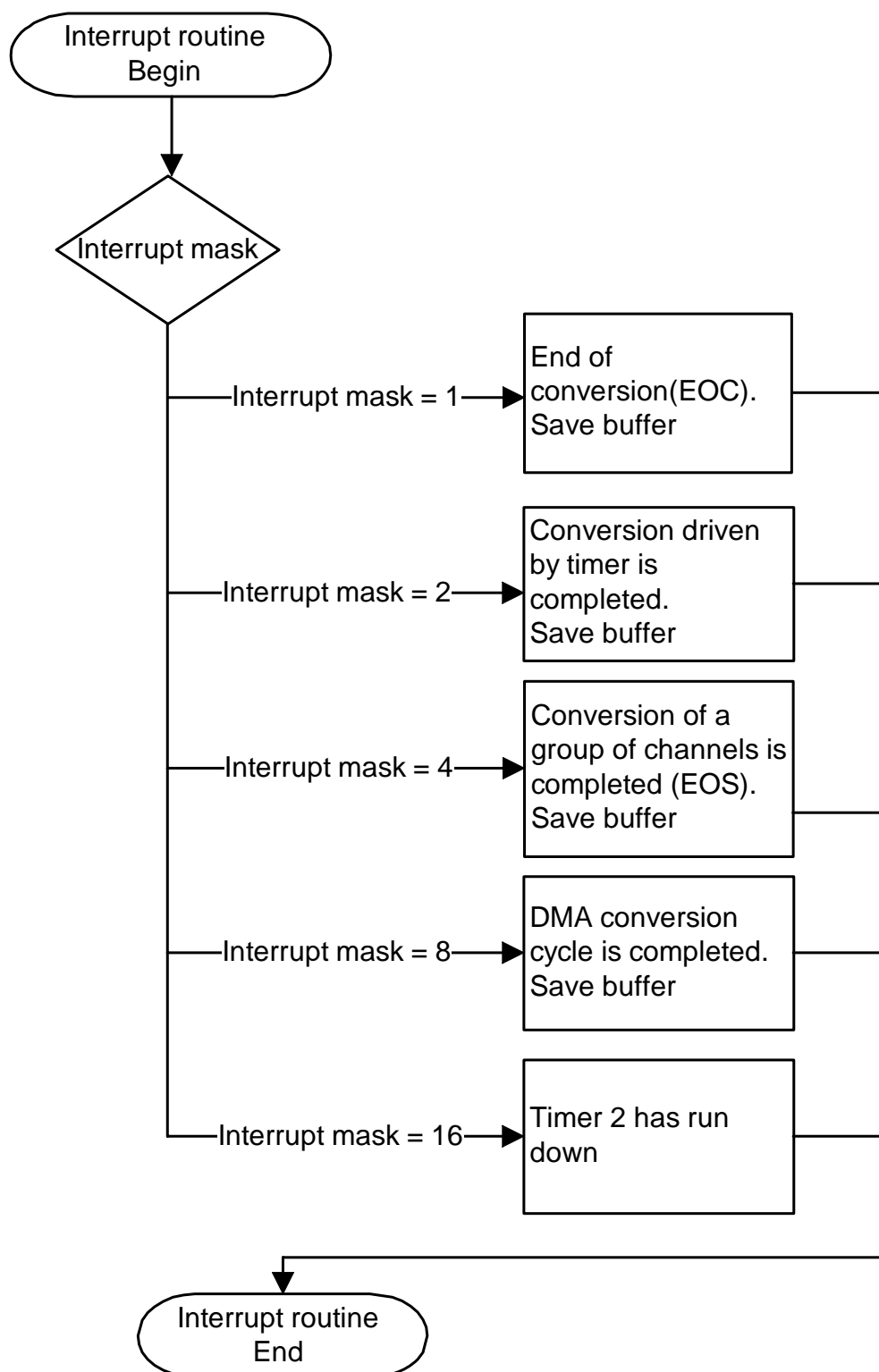
    if(i_NbrOfBoard > 0)
    {
        for (i_Cpt = 0; i_Cpt < i_NbrOfBoard; i_Cpt ++)
        {
            if (i_APCI3120_SetBoardInformation (b_SlotNumberArray[i_Cpt],
                                                16,
                                                8,
                                                &pb_BoardHandleArray [i_Cpt]) != 0)
            {
                break;
            }
        }

        if (i_Cpt == i_NbrOfBoard)
        {
            return (i_Cpt); /* Return number of board found */
        }
        else
        {
            return (-1);    /* ERROR */
        }
    }
    else
    {
        return (-1);       /* ERROR */
    }
}
```


11.2 Interrupt

11.2.1 Interrupt routine

a) Flow chart



b) Example in C for DOS and Windows 3.1x

```

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

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

    switch(b_InterruptMask)
    {
        case 1:
            /* EOC interrupt */
            ui_SaveArray[0] = pui_ValueArray[1];
            break;

        case 2:
            /* EOS interrupt Acquisition */
            for (ui_Cpt=0;ui_Cpt<pui_ValueArray [0];ui_Cpt++)
                ui_SaveArray [ui_Cpt] = pui_ValueArray [1+ui_Cpt];
            break;

        case 4:
            /* EOS interrupt Read More*/
            for (ui_Cpt=0;ui_Cpt<pui_ValueArray [0];ui_Cpt++)
                ui_SaveArray [ui_Cpt] = pui_ValueArray [1+ui_Cpt];
            break;

        case 8:
            /* DMA completed */
            for (ui_Cpt=0;ui_Cpt<16;ui_Cpt++)
                ui_SaveArray [ui_Cpt] = pui_ValueArray [ui_Cpt];
            break;

        case 16:
            /* Timer 2 has run down */
            ui_TimerIntCpt = ui_TimerIntCpt + 1;
            break;

        default :
            break;
    }
    b_ReceiveInterrupt = 1;
}

```


c) Example in C for Windows NT and Windows 95/98 (asynchronous mode)

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

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

    pusi_Index = pui_ValueArray;
    switch(b_InterruptMask)
    {
        case 1:
            /* EOC interrupt */
            ui_SaveArray[0] = pui_ValueArray[1];
            break;

        case 2:
            /* EOS interrupt Acquisition */
            for (ul_Cpt=0;ul_Cpt<pui_ValueArray [0];ul_Cpt++)
                ui_SaveArray [ul_Cpt] = pui_ValueArray [1+ul_Cpt];
            break;

        case 4:
            /* EOS interrupt Read More*/
            for (ul_Cpt=0;ul_Cpt<pui_ValueArray [0];ul_Cpt++)
                ui_SaveArray [ul_Cpt] = pui_ValueArray [1+ul_Cpt];
            break;

        case 8:
            /* DMA completed */
            for (ul_Cpt=0;ul_Cpt<ul_NbrAcquisitionDMA;ul_Cpt++)
                ui_SaveArray [ul_Cpt] = pusi_Index[ul_Cpt];
            break;

        case 16:
            /* Timer 2 has run down */
            ui_TimerIntCpt = ui_TimerIntCpt + 1;
            break;

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


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

```

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

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

    str_UserStruct *ps_UserStruct = (str_UserStruct *) pv_UserSharedMemory;
    pusi_Index = pui_ValueArray;

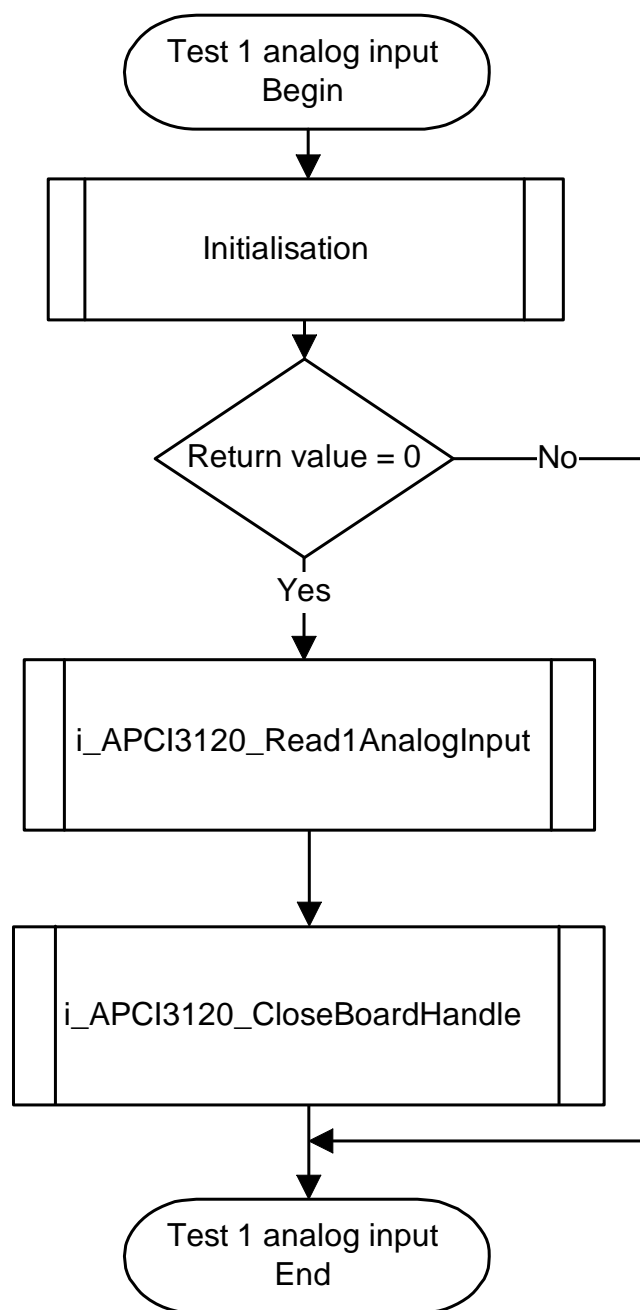
    i_APCI3120_KRNL_Set1DigitalOutputOn(0x390,1);
    if ((b_InterruptMask&1) == 1) /* EOC interrupt */
    {
        ps_UserStruct->ui_SaveArray[0] = pui_ValueArray[1];
    }
    if ((b_InterruptMask&2) == 2) /* EOS interrupt Acquisition */
    {
        for (ui_Cpt= 1;ui_Cpt<= pui_ValueArray [0];ui_Cpt++)
            ps_UserStruct->ui_SaveArray [ui_Cpt] = pui_ValueArray [ui_Cpt];
    }
    if ((b_InterruptMask&4) == 4) /* EOS interrupt Read More*/
    {
        for (ui_Cpt=0;ui_Cpt<pui_ValueArray [0];ui_Cpt++)
            ps_UserStruct->ui_SaveArray [ui_Cpt] = pui_ValueArray [1+ui_Cpt];
    }
    if ((b_InterruptMask&8) == 8) /* DMA completed */
    {
        for (ui_Cpt=0;ui_Cpt<16;ui_Cpt++)
            ps_UserStruct->ui_SaveArray [ui_Cpt] = pusi_Index[ui_Cpt];
    }
    if ((b_InterruptMask&16) == 16)
    {
        /* Timer 2 has run down */
        ps_UserStruct->ui_TimerIntCpt = ps_UserStruct->ui_TimerIntCpt + 1;
    }
    i_APCI3120_KRNL_Set1DigitalOutputOn(0x390,2);
    ps_UserStruct->b_ReceiveInterrupt =ps_UserStruct->b_ReceiveInterrupt + 1;
}

```


11.3 Direct conversion of the analog input channels

11.3.1 Testing an analog input channel

a) Flow chart



b) Pin assignment

The analog input channels are set in single mode. See chapter 5.1 „Jumper settings“ page 11.

Pin 21: Read the analog input 1

Set the voltage between 0 and 10 V.

Pin 28: GND: Set to 0 V

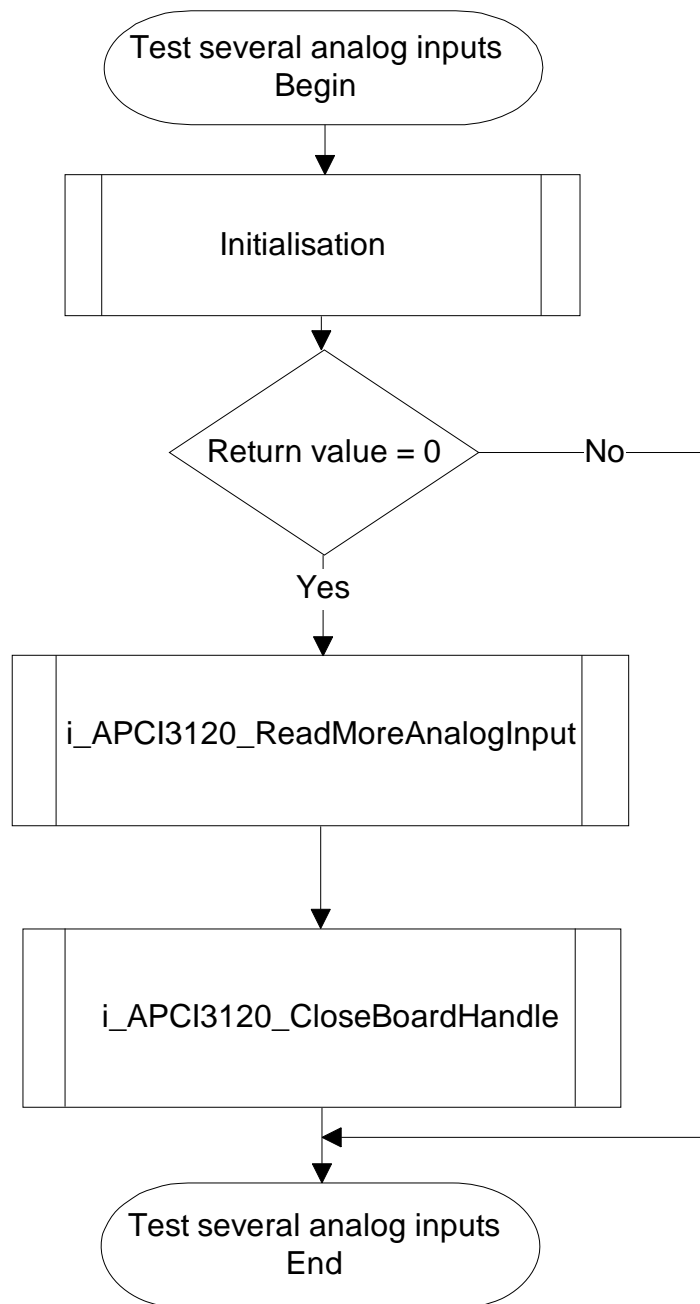
c) Example in C

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

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_APCI3120_Read1AnalogInput (b_BoardHandle,
                                         APCI3120_CHANNEL_1,
                                         APCI3120_1_GAIN,
                                         APCI3120_UNIPOLAR,
                                         10,
                                         APCI3120_DISABLE,
                                         &ui_ReadValue) == 0)
        {
            printf ("ui_ReadValue = %u", ui_ReadValue);
        }
        else
        {
            printf ("Read value error");
        }
        i_APCI3120_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```


11.3.2 Testing several analog input channels

a) Flow chart



b) Pin assignment

The analog input channels are set in single mode. See chapter 5.1 „Jumper settings“ page 11.

Pin 28: GND: Set to 0 V
an. input 0 to 10 V.

Pin 20: Set the

Pin 21: Set the an. input 1 to 10 V.

Pin 22: Set the an. input 2 to 10 V.

Pin 23: Set the an. input 3 to 10 V

Pin 27: Set the an. input 4 to 10 V.

Pin 26: Set the an. input 5 to 10 V

Pin 25: Set the an. input 1 to 10 V.

Pin 24: Set the an. input 7 to 10 V

The value to be read is 65535 for each input.

c) Example in C

```

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

    b_Channel[0] = APCI3120_CHANNEL_0;
    b_Channel[1] = APCI3120_CHANNEL_1;
    b_Channel[2] = APCI3120_CHANNEL_2;
    b_Channel[3] = APCI3120_CHANNEL_3;
    b_Channel[4] = APCI3120_CHANNEL_4;
    b_Channel[5] = APCI3120_CHANNEL_5;
    b_Channel[6] = APCI3120_CHANNEL_6;
    b_Channel[7] = APCI3120_CHANNEL_7;

    b_Gain[0] = APCI3120_0_GAIN;
    b_Gain[1] = APCI3120_1_GAIN;
    b_Gain[2] = APCI3120_1_GAIN;
    b_Gain[3] = APCI3120_1_GAIN;
    b_Gain[4] = APCI3120_1_GAIN;
    b_Gain[5] = APCI3120_1_GAIN;
    b_Gain[6] = APCI3120_1_GAIN;
    b_Gain[7] = APCI3120_1_GAIN;

    b_Polar[0] = APCI3120_UNIPOLAR;
    b_Polar[1] = APCI3120_UNIPOLAR;
    b_Polar[2] = APCI3120_UNIPOLAR;
    b_Polar[3] = APCI3120_UNIPOLAR;
    b_Polar[4] = APCI3120_UNIPOLAR;
    b_Polar[5] = APCI3120_UNIPOLAR;
    b_Polar[6] = APCI3120_UNIPOLAR;
    b_Polar[7] = APCI3120_UNIPOLAR;

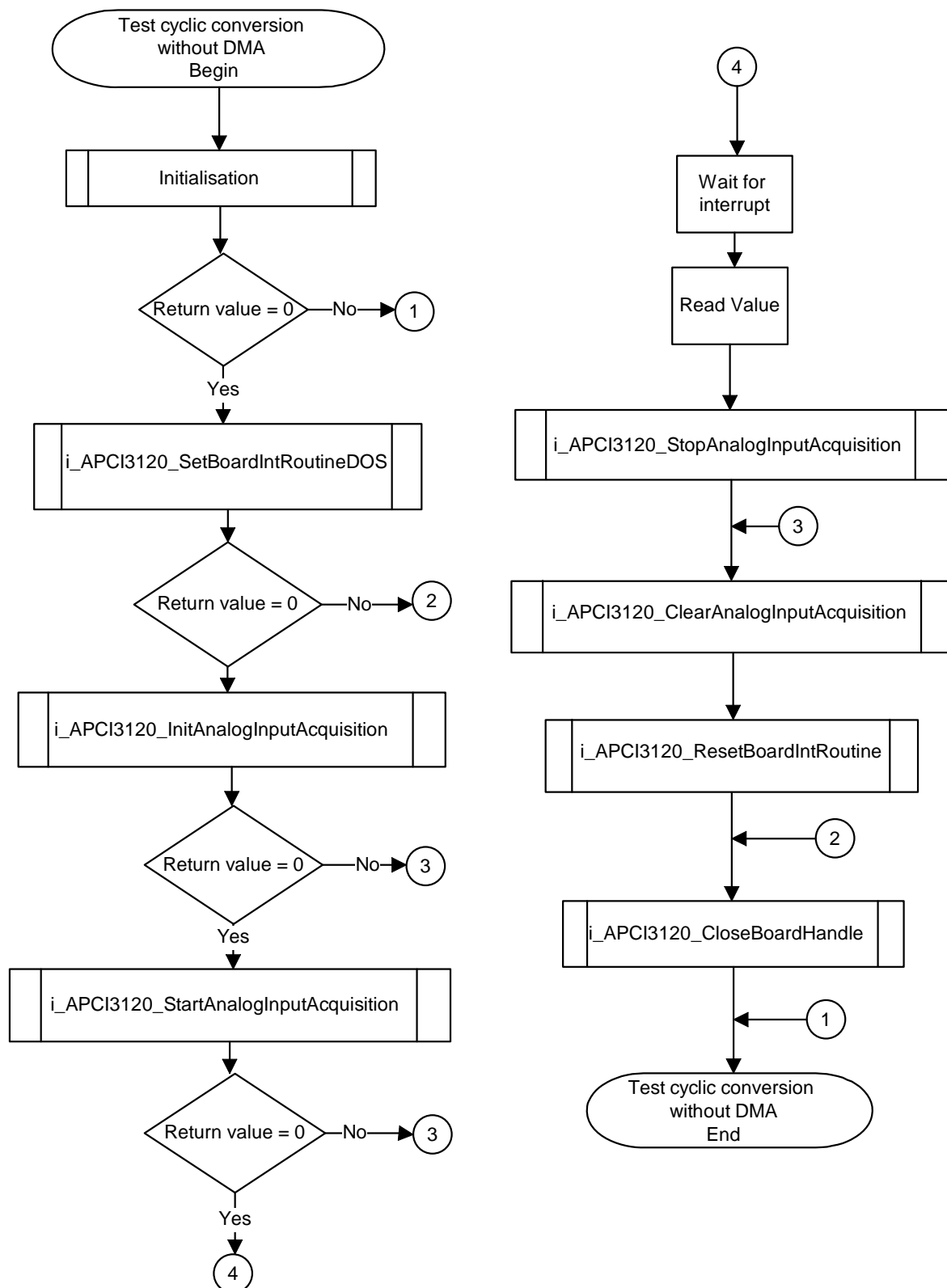
    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_APCI3120_ReadMoreAnalogInput (b_BoardHandle, 8, b_Channel, b_Gain,
                                            b_Polar, 10, APCI3120_DISABLE,
                                            ui_ReadValueArray) == 0)
        {
            printf ("ui_ReadValue = %u %u %u %u %u %u %u %u",
                    ui_ReadValue [0], ui_ReadValue [1], ui_ReadValue [2], ui_ReadValue [3],
                    ui_ReadValue [4], ui_ReadValue [5], ui_ReadValue [6], ui_ReadValue [7]);
        }
        else
        {
            printf ("Read value error");
        }
        i_APCI3120_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}

```


11.4 Cyclic conversion of the analog input channels

11.4.1 Cyclic conversion without DMA, external trigger and delay

a) Flow chart



b) Pin assignment

The analog input channels are in single mode .(See Jumper installation).

Pin 28: GND: Set to 0 V.

Pin 20: Set input 0 to 10 V

Pin 21: Set input 1 to 10 V

Pin 22: Set input 2 to 10 V

Pin 23: Set input 3 to 10 V .

The value to be read is 65535 for each input channel.

c) Example in C for DOS

```
void main(void)
{
    int i_Cpt;
    unsigned char b_Gain [4];
    unsigned char b_Polar [4];
    unsigned char b_Channel [4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if (i_APCI3120_SetBoardIntRoutineDos(b_BoardHandle, v_InterruptRoutine) == 0)
        {
            b_Channel[0] = APCI3120_CHANNEL_0;b_Gain[0] = APCI3120_1_GAIN;b_Polar[0] = APCI3120_UNIPOLAR;
            b_Channel[1] = APCI3120_CHANNEL_1;b_Gain[1] = APCI3120_1_GAIN;b_Polar[1] = APCI3120_UNIPOLAR;
            b_Channel[2] = APCI3120_CHANNEL_2;b_Gain[2] = APCI3120_1_GAIN;b_Polar[2] = APCI3120_UNIPOLAR;
            b_Channel[3] = APCI3120_CHANNEL_3;b_Gain[3] = APCI3120_1_GAIN;b_Polar[3] = APCI3120_UNIPOLAR;
            if (i_APCI3120_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
            APCI3120_SIMPLE_MODUS,APCI3120_DISABLE,1500,0,4,APCI3120_DMA_NOT_USED,APCI3120_SINGLE)==0)
            {
                b_ReceiveInterrupt = 0;
                if (i_APCI3120_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_APCI3120_StopAnalogInputAcquisition(b_BoardHandle);
                }
                else
                {
                    printf("\n Start acquisition error");
                    i_APCI3120_ClearAnalogInputAcquisition(b_BoardHandle);
                }
            }
            else
            {
                printf("\n Acquisition initialisation error");
                i_APCI3120_ResetBoardIntRoutine(b_BoardHandle);
            }
        }
        else
        {
            printf("\n Interrupt routine initialisation error");
            i_APCI3120_CloseBoardHandle(b_BoardHandle);
        }
    }
    else
    {
        printf("\n Initialisation error");
    }
}
```


d) Example in C for Windows 3.1x

```

Void main(void)
{
    int i_Cpt;
    unsigned char b_Gain [4];
    unsigned char b_Polar [4];
    unsigned char b_Channel [4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if (i_APCI3120_SetBoardIntRoutineWin16(b_BoardHandle, v_InterruptRoutine) == 0)
        {
            b_Channel[0] = APCI3120_CHANNEL_0;b_Gain[0] = APCI3120_1_GAIN;b_Polar[0] = APCI3120_UNIPOLAR;
            b_Channel[1] = APCI3120_CHANNEL_1;b_Gain[1] = APCI3120_1_GAIN;b_Polar[1] = APCI3120_UNIPOLAR;
            b_Channel[2] = APCI3120_CHANNEL_2;b_Gain[2] = APCI3120_1_GAIN;b_Polar[2] = APCI3120_UNIPOLAR;
            b_Channel[3] = APCI3120_CHANNEL_3;b_Gain[3] = APCI3120_1_GAIN;b_Polar[3] = APCI3120_UNIPOLAR;
            if (i_APCI3120_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
APCI3120_SIMPLE_MODUS,APCI3120_DISABLE,1500,0,4,APCI3120_DMA_NOT_USED,APCI3120_SINGLE)==0)
            {
                b_ReceiveInterrupt = 0;
                if (i_APCI3120_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_APCI3120_StopAnalogInputAcquisition(b_BoardHandle);
                }
                else
                {
                    printf("\n Start acquisition error");
                    i_APCI3120_ClearAnalogInputAcquisition(b_BoardHandle);
                }
            }
            else
            {
                printf("\n Acquisition initialisation error");
                i_APCI3120_ResetBoardIntRoutine(b_BoardHandle);
            }
        }
        else
        {
            printf("\n Interrupt routine initialisation error");
            i_APCI3120_CloseBoardHandle(b_BoardHandle);
        }
    }
    else
    {
        printf("\n Initialisation error");
    }
}

```


e) Example in C for windows NT/95/98 (Asynchronous mode)

```

void main(void)
{
    int i_Cpt;
    unsigned char b_Gain [4];
    unsigned char b_Polar [4];
    unsigned char b_Channel [4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if (i_APCI3120_SetBoardIntRoutineWin32(b_BoardHandle,0,NULL,v_InterruptRoutine) == 0)
        {
            b_Channel[0] = APCI3120_CHANNEL_0;b_Gain[0] = APCI3120_1_GAIN;b_Polar[0] = APCI3120_UNIPOLAR;
            b_Channel[1] = APCI3120_CHANNEL_1;b_Gain[1] = APCI3120_1_GAIN;b_Polar[1] = APCI3120_UNIPOLAR;
            b_Channel[2] = APCI3120_CHANNEL_2;b_Gain[2] = APCI3120_1_GAIN;b_Polar[2] = APCI3120_UNIPOLAR;
            b_Channel[3] = APCI3120_CHANNEL_3;b_Gain[3] = APCI3120_1_GAIN;b_Polar[3] = APCI3120_UNIPOLAR;
            if (i_APCI3120_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
APCI3120_SIMPLE_MODUS,APCI3120_DISABLE,1500,0,4,APCI3120_DMA_NOT_USED,APCI3120_SINGLE)==0)
            {
                b_ReceiveInterrupt = 0;
                if (i_APCI3120_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_APCI3120_StopAnalogInputAcquisition(b_BoardHandle);
                }
                else
                {
                    printf("\n Start acquisition error");
                    i_APCI3120_ClearAnalogInputAcquisition(b_BoardHandle);
                }
            }
            else
            {
                printf("\n Acquisition initialisation error");
                i_APCI3120_ResetBoardIntRoutine(b_BoardHandle);
            }
        }
        else
        {
            printf("\n Interrupt routine initialisation error");
            i_APCI3120_CloseBoardHandle(b_BoardHandle);
        }
    }
    else
    {
        printf("\n Initialisation error");
    }
}

```


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

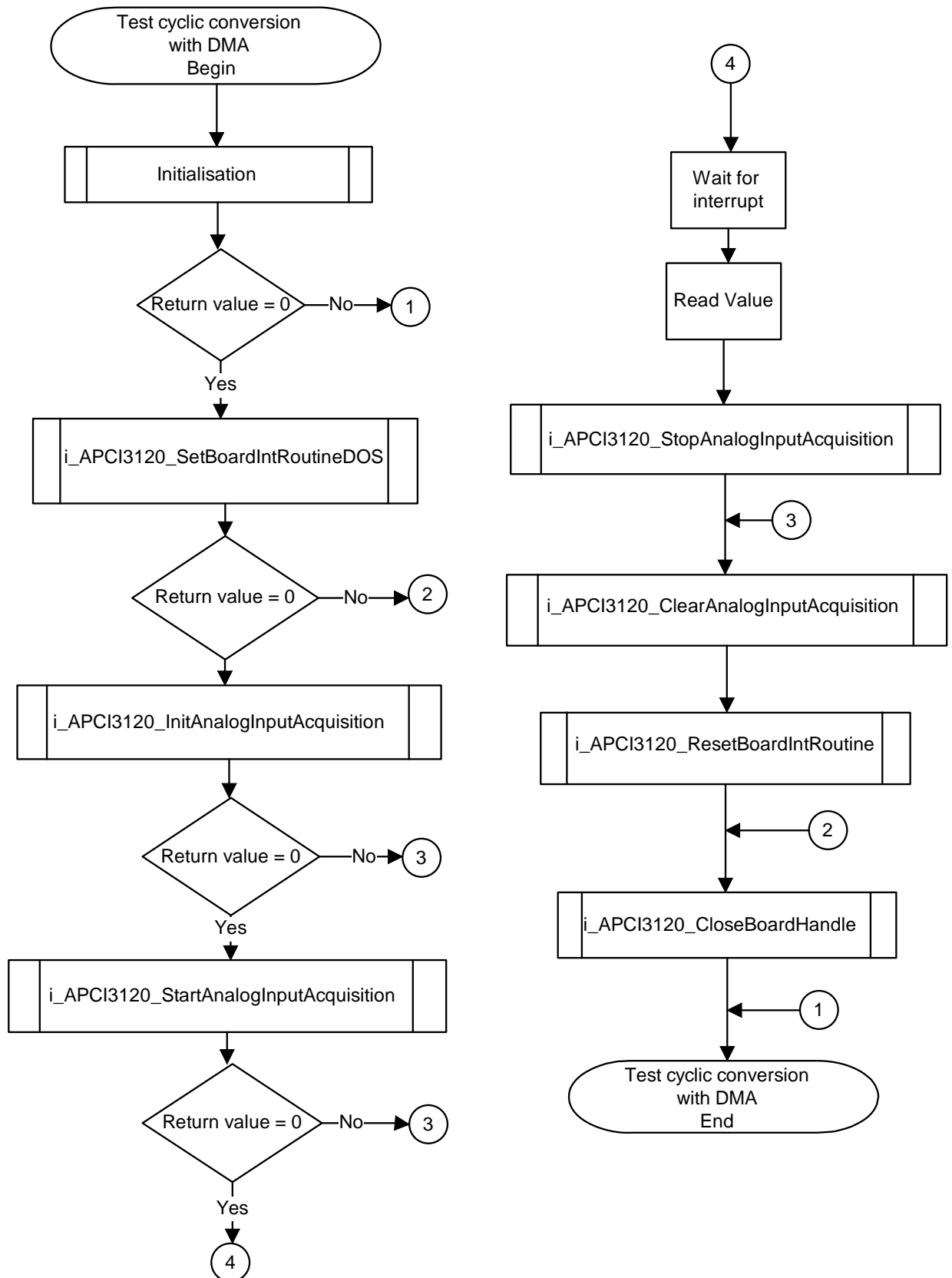
```

Void main(void)
{
    int i_Cpt;
    unsigned char b_Gain [4]; unsigned char b_Polar [4]; unsigned char b_Channel [4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if (i_APCI3120_SetBoardIntRoutineWin32(b_BoardHandle, sizeof(str_UserStruct),
                                                (void **))
        &ps_GlobalUserStruct, v_InterruptRoutine) == 0)
        {
            b_Channel[0] = APCI3120_CHANNEL_0; b_Gain[0] = APCI3120_1_GAIN; b_Polar[0] = APCI3120_UNIPOLAR;
            b_Channel[1] = APCI3120_CHANNEL_1; b_Gain[1] = APCI3120_1_GAIN; b_Polar[1] = APCI3120_UNIPOLAR;
            b_Channel[2] = APCI3120_CHANNEL_2; b_Gain[2] = APCI3120_1_GAIN; b_Polar[2] = APCI3120_UNIPOLAR;
            b_Channel[3] = APCI3120_CHANNEL_3; b_Gain[3] = APCI3120_1_GAIN; b_Polar[3] = APCI3120_UNIPOLAR;
            if (i_APCI3120_InitAnalogInputAcquisition (b_BoardHandle, 4, b_Channel, b_Gain, b_Polar,
                                                        APCI3120_SIMPLE_MODUS, APCI3120_DISABLE, 1500,
                                                        0, 4, APCI3120_DMA_NOT_USED, APCI3120_SINGLE) == 0)
            {
                ps_GlobalUserStruct -> b_ReceiveInterrupt = 0;
                if (i_APCI3120_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_APCI3120_StopAnalogInputAcquisition(b_BoardHandle);
                }
                else
                {
                    printf("\n Start acquisition error");
                    i_APCI3120_ClearAnalogInputAcquisition(b_BoardHandle);
                }
            }
            else
            {
                printf("\n Acquisition initialisation error");
                i_APCI3120_ResetBoardIntRoutine(b_BoardHandle);
            }
        }
        else
        {
            printf("\n Interrupt routine initialisation error");
            i_APCI3120_CloseBoardHandle(b_BoardHandle);
        }
    }
    else
    {
        printf("\n Initialisation error");
    }
}

```


11.4.2 Cyclic conversion with DMA, without external trigger and delay

a) Flow chart



b) Pin assignment

The analog input channels are in single mode .(See Jumper installation).

Pin 28: GND: Set to 0 V.

Pin 20: Set input 0 to 10 V

Pin 21: Set input 1 to 10 V

Pin 22: Set input 2 to 10 V

Pin 23: Set input 3 to 10 V .

The value to be read is 65535 for each input channel.

c) Example in C for DOS

```
void main(void)
{
    int i_Cpt;
    unsigned char b_Gain [4];
    unsigned char b_Polar [4];
    unsigned char b_Channel [4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if (i_APCI3120_SetBoardIntRoutineDos(b_BoardHandle, v_InterruptRoutine) == 0)
        {
            b_Channel[0] = APCI3120_CHANNEL_0;b_Gain[0] = APCI3120_1_GAIN;b_Polar[0] = APCI3120_UNIPOLAR;
            b_Channel[1] = APCI3120_CHANNEL_1;b_Gain[1] = APCI3120_1_GAIN;b_Polar[1] = APCI3120_UNIPOLAR;
            b_Channel[2] = APCI3120_CHANNEL_2;b_Gain[2] = APCI3120_1_GAIN;b_Polar[2] = APCI3120_UNIPOLAR;
            b_Channel[3] = APCI3120_CHANNEL_3;b_Gain[3] = APCI3120_1_GAIN;b_Polar[3] = APCI3120_UNIPOLAR;
            if (i_APCI3120_InitAnalogInputAcquisition
                (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                 APCI3120_SIMPLE_MODUS,APCI3120_DISABLE,1500,0,16,APCI3120_DMA_USED,APCI3120_SINGLE) == 0)
            {
                b_ReceiveInterrupt = 0;
                if (i_APCI3120_StartAnalogInputAcquisition (b_BoardHandle) == 0)
                {
                    while (b_ReceiveInterrupt == 0);
                    b_ReceiveInterrupt = 0;
                    printf("\n Acquisition 1 %u %u %u %u %u %u %u %u \n %u %u %u %u %u %u %u",
                        ui_SaveArray[0],ui_SaveArray[1],ui_SaveArray[2],ui_SaveArray[3],
                        ui_SaveArray[4],ui_SaveArray[5],ui_SaveArray[6],ui_SaveArray[7],
                        ui_SaveArray[8],ui_SaveArray[9],ui_SaveArray[10],ui_SaveArray[11],
                        ui_SaveArray[12],ui_SaveArray[13],ui_SaveArray[14],ui_SaveArray[15]);
                    i_APCI3120_StopAnalogInputAcquisition(b_BoardHandle);
                    i_APCI3120_ClearAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
            }
            else
                printf("\n Acquisition initialisation error");
            i_APCI3120_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_APCI3120_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}
```


d) Example in C for Windows 3.1x

```

void main(void)
{
    int i_Cpt;
    unsigned char b_Gain [4];
    unsigned char b_Polar [4];
    unsigned char b_Channel [4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if (i_APCI3120_SetBoardIntRoutineWin16(b_BoardHandle, v_InterruptRoutine) == 0)
        {
            b_Channel[0] = APCI3120_CHANNEL_0;b_Gain[0] = APCI3120_1_GAIN;b_Polar[0] = APCI3120_UNIPOLAR;
            b_Channel[1] = APCI3120_CHANNEL_1;b_Gain[1] = APCI3120_1_GAIN;b_Polar[1] = APCI3120_UNIPOLAR;
            b_Channel[2] = APCI3120_CHANNEL_2;b_Gain[2] = APCI3120_1_GAIN;b_Polar[2] = APCI3120_UNIPOLAR;
            b_Channel[3] = APCI3120_CHANNEL_3;b_Gain[3] = APCI3120_1_GAIN;b_Polar[3] = APCI3120_UNIPOLAR;
            if (i_APCI3120_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
            APCI3120_SIMPLE_MODUS,APCI3120_DISABLE,1500,0,16,APCI3120_DMA_USED,APCI3120_SINGLE) == 0)
            {
                b_ReceiveInterrupt = 0;
                if (i_APCI3120_StartAnalogInputAcquisition (b_BoardHandle) == 0)
                {
                    while (b_ReceiveInterrupt == 0);
                    b_ReceiveInterrupt = 0;
                    printf("\n Acquisition 1 %u %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_APCI3120_StopAnalogInputAcquisition(b_BoardHandle);
                    i_APCI3120_ClearAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
            }
            else
                printf("\n Acquisition initialisation error");
            i_APCI3120_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_APCI3120_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```


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

```

void main(void)
{
    int i_Cpt;
    unsigned char b_Gain [4];
    unsigned char b_Polar [4];
    unsigned char b_Channel [4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if (i_APCI3120_SetBoardIntRoutineWin32(b_BoardHandle,0,NULL,v_InterruptRoutine) == 0)
        {
            b_Channel[0] = APCI3120_CHANNEL_0;b_Gain[0] = APCI3120_1_GAIN;b_Polar[0] = APCI3120_UNIPOLAR;
            b_Channel[1] = APCI3120_CHANNEL_1;b_Gain[1] = APCI3120_1_GAIN;b_Polar[1] = APCI3120_UNIPOLAR;
            b_Channel[2] = APCI3120_CHANNEL_2;b_Gain[2] = APCI3120_1_GAIN;b_Polar[2] = APCI3120_UNIPOLAR;
            b_Channel[3] = APCI3120_CHANNEL_3;b_Gain[3] = APCI3120_1_GAIN;b_Polar[3] = APCI3120_UNIPOLAR;
            if (i_APCI3120_InitAnalogInputAcquisition (b_BoardHandle,4,b_Channel,b_Gain,b_Polar,
                APCI3120_SIMPLE_MODUS,APCI3120_DISABLE,1500,0,16,APCI3120_DMA_USED,APCI3120_SINGLE) ==
0)
            {
                b_ReceiveInterrupt = 0;
                if (i_APCI3120_StartAnalogInputAcquisition (b_BoardHandle) == 0)
                {
                    while (b_ReceiveInterrupt == 0);
                    b_ReceiveInterrupt = 0;
                    printf("\n Acquisition 1 %u %u %u %u %u %u %u %u \n %u %u %u %u %u %u %u",
                        ui_SaveArray[0],ui_SaveArray[1],ui_SaveArray[2],ui_SaveArray[3],
                        ui_SaveArray[4],ui_SaveArray[5],ui_SaveArray[6],ui_SaveArray[7],
                        ui_SaveArray[8],ui_SaveArray[9],ui_SaveArray[10],ui_SaveArray[11],
ui_SaveArray[12],ui_SaveArray[13],ui_SaveArray[14],ui_SaveArray[15]);
                    i_APCI3120_StopAnalogInputAcquisition(b_BoardHandle);
                    i_APCI3120_ClearAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
            }
            else
                printf("\n Acquisition initialisation error");
            i_APCI3120_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_APCI3120_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```


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

```

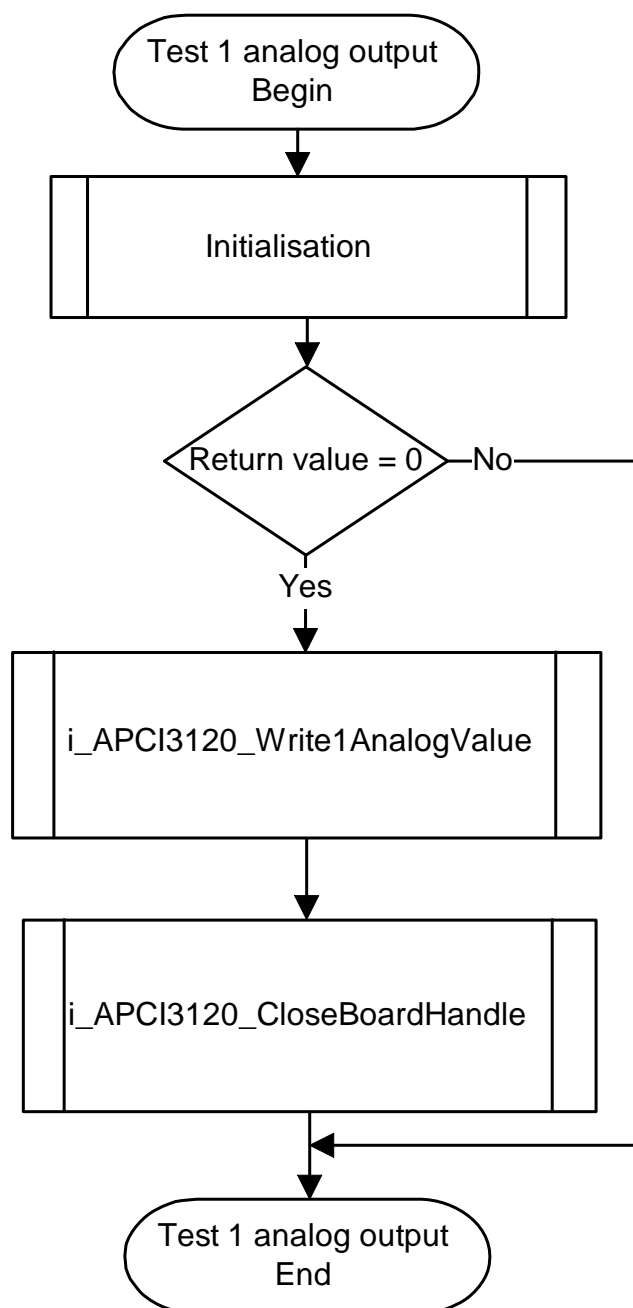
void main(void)
{
int i_Cpt; unsigned char b_Gain [4]; unsigned char b_Polar [4]; unsigned char b_Channel [4];
    unsigned char b_BoardHandle;
    if (Initialisation(&b_BoardHandle) == 0)
    {
        if (i_APCI3120_SetBoardIntRoutineWin32(b_BoardHandle, sizeof(str_UserStruct),
            (void **) &GlobalUserStruct, v_InterruptRoutine) == 0)
        {
            b_Channel[0] = APCI3120_CHANNEL_0; b_Gain[0] = APCI3120_1_GAIN; b_Polar[0] = APCI3120_UNIPOLAR;
            b_Channel[1] = APCI3120_CHANNEL_1; b_Gain[1] = APCI3120_1_GAIN; b_Polar[1] = APCI3120_UNIPOLAR;
            b_Channel[2] = APCI3120_CHANNEL_2; b_Gain[2] = APCI3120_1_GAIN; b_Polar[2] = APCI3120_UNIPOLAR;
            b_Channel[3] = APCI3120_CHANNEL_3; b_Gain[3] = APCI3120_1_GAIN; b_Polar[3] = APCI3120_UNIPOLAR;
            if (i_APCI3120_InitAnalogInputAcquisition (b_BoardHandle, 4, b_Channel, b_Gain, b_Polar,
                APCI3120_SIMPLE_MODUS, APCI3120_DISABLE, 1500, 0, 16, APCI3120_DMA_USED, APCI3120_SINGLE) == 0)
            {
                ps_GlobalUserStruct -> b_ReceiveInterrupt = 0;
                if (i_APCI3120_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 %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_APCI3120_StopAnalogInputAcquisition(b_BoardHandle);
                    i_APCI3120_ClearAnalogInputAcquisition(b_BoardHandle);
                }
                else
                    printf("\n Start acquisition error");
            }
            else
                printf("\n Acquisition initialisation error");
            i_APCI3120_ResetBoardIntRoutine(b_BoardHandle);
        }
        else
            printf("\n Interrupt routine initialisation error");
        i_APCI3120_CloseBoardHandle(b_BoardHandle);
    }
    else
        printf("\n Initialisation error");
}

```


11.5 Analog output channels

11.5.1 Testing one analog output channel

a) Flow chart



b) Pin assignment

Write 1 analog output channel .

Connect a voltmeter : - on pin 30 (analog output 0 GND) + on pin 12 (analog output 0) .

The voltage to be measured is 10V.

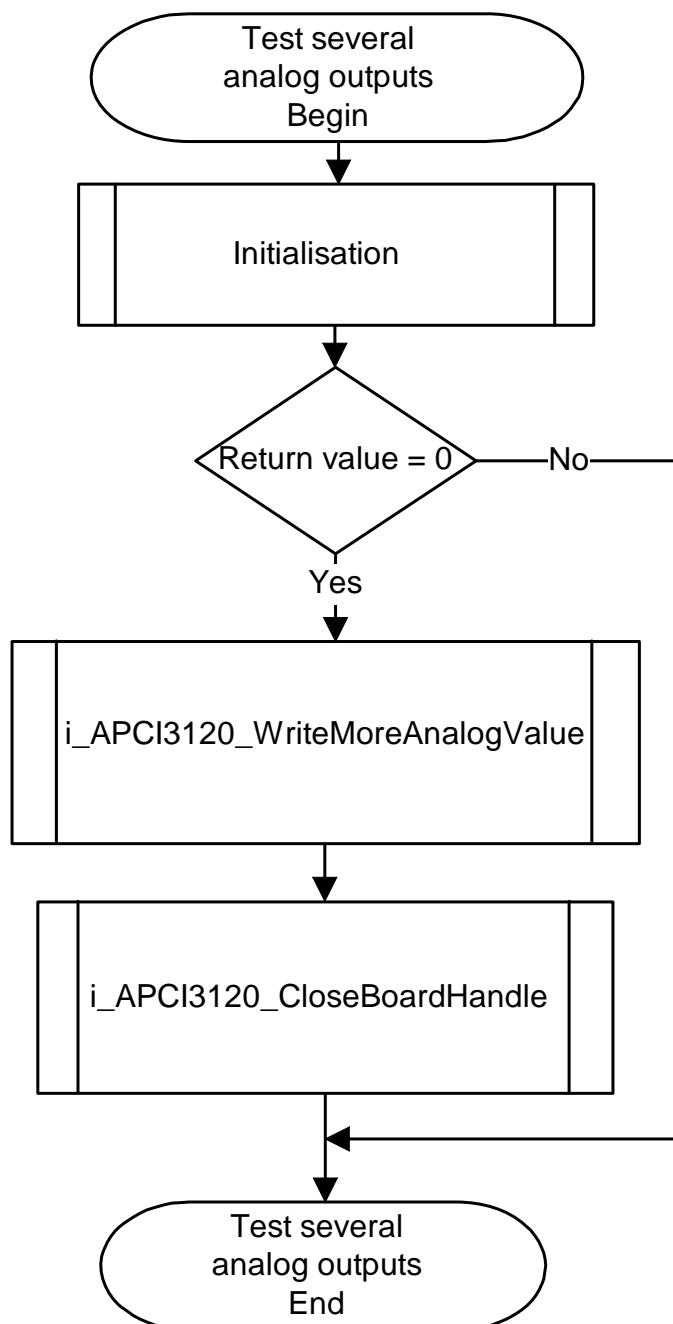
c) Example in C

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

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_APCI3120_WriteAnalogValue (b_BoardHandle,
                                         0,
                                         APCI3120_UNIPOLAR,
                                         8192) == 0)
        {
            printf ("Write test OK");
        }
        else
        {
            printf ("Write value error");
        }
        i_APCI3120_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```


11.5.2 Testing several analog output channels

a) Flow chart



b) Pin assignment

Write several analog output channels.

Connect a voltmeter :

Output 0: - on pin 30 (analog output 0 GND)
 + on pin 12 (analog output 0) .

The voltage to be measured is 0V.

Output 1: - on pin 31 (analog output 1 GND)
 + on pin 13 (analog output 1) .

The voltage to be measured is 5V.

- Output 2: - on pin 32 (analog output 2 GND)
 + on pin 14 (analog output 2) .
 The voltage to be measured is 10V.
- Output 3: - on pin 33 (analog output 3 GND)
 + on pin 15 (analog output 3) .
 The voltage to be measured is 0V.
- Output 4: - on pin 34 (analog output 4 GND)
 + on pin 16 (analog output 4) .
 The voltage to be measured is 5V.
- Output 5: - on pin 35 (analog output 5 GND)
 + on pin 17 (analog output 5) .
 The voltage to be measured is 10V.
- Output 6: - on pin 36 (analog output 6 GND)
 + on pin 18 (analog output 6) .
 The voltage to be measured is 0V.
- Output 7: - on pin 37 (analog output 7 GND)
 + on pin 19 (analog output 7) .
 The voltage to be measured is 5V.

c) Example in C

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

    b_Channel[0] = 0; b_Polarity[0] = APCI3120_UNIPOLAR; ui_WriteValueArray [0] = 0;
    b_Channel[1] = 1; b_Polarity[1] = APCI3120_UNIPOLAR; ui_WriteValueArray [1] = 4095;
    b_Channel[2] = 2; b_Polarity[2] = APCI3120_UNIPOLAR; ui_WriteValueArray [2] = 8192;
    b_Channel[3] = 3; b_Polarity[3] = APCI3120_UNIPOLAR; ui_WriteValueArray [3] = 0;
    b_Channel[4] = 4; b_Polarity[4] = APCI3120_UNIPOLAR; ui_WriteValueArray [4] = 4095;
    b_Channel[5] = 5; b_Polarity[5] = APCI3120_UNIPOLAR; ui_WriteValueArray [5] = 8192;
    b_Channel[6] = 6; b_Polarity[6] = APCI3120_UNIPOLAR; ui_WriteValueArray [6] = 0;
    b_Channel[7] = 7; b_Polarity[7] = APCI3120_UNIPOLAR; ui_WriteValueArray [7] = 4095;

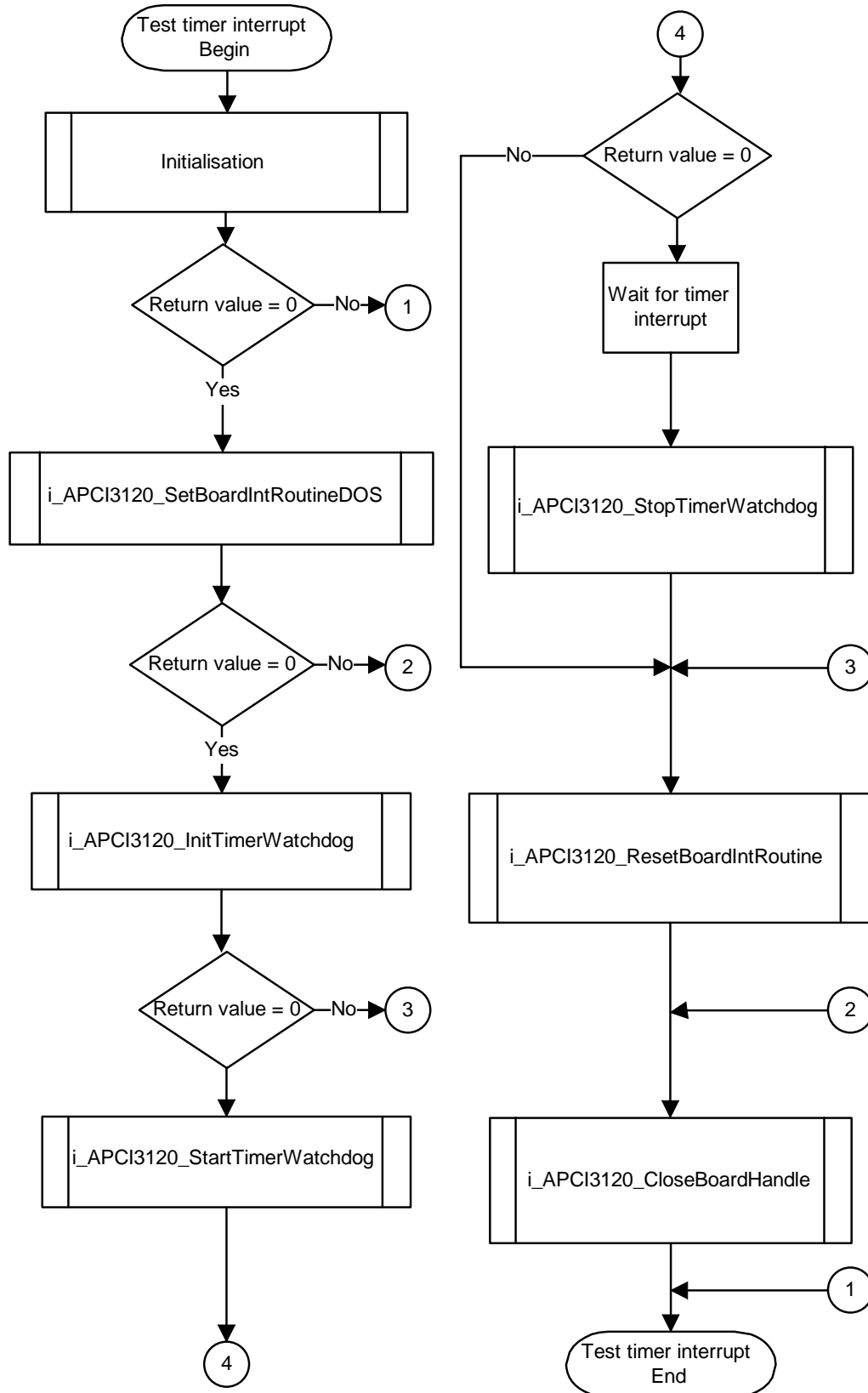
    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_APCI3120_WriteMoreAnalogValue (b_BoardHandle, 1, 8, b_Polarity,
                                             ui_WriteValueArray) == 0)
        {
            printf ("Write test OK");
        }
        else
        {
            printf ("Write value error");
        }

        i_APCI3120_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```


11.6 Timer

11.6.1 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_APCI3120_SetBoardIntRoutineDOS (b_BoardHandle, v_InterruptRoutine) == 0)
        {
            ui_TimerIntCpt = 0;
            if (i_APCI3120_InitTimerWatchdog (b_BoardHandle, APCI3120_TIMER,
                                              1000, APCI3120_ENABLE) == 0)
            {
                if (i_APCI3120_StartTimerWatchdog (b_BoardHandle) == 0)
                {
                    while (ui_TimerIntCpt == 0);
                    printf ("Receive timer interrupt");
                    i_APCI3120_StopTimerWatchdog (b_BoardHandle);
                }
                else
                    printf ("Start timer error");
            }
            else
                printf ("Init timer error");
            i_APCI3120_ResetBoardIntRoutine (b_BoardHandle);
        }
        else
            printf ("Interrupt routine initialisation error");
        i_APCI3120_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_APCI3120_SetBoardIntRoutineWin16 (b_BoardHandle, v_InterruptRoutine) == 0)
        {
            ui_TimerIntCpt = 0;
            if (i_APCI3120_InitTimerWatchdog (b_BoardHandle, APCI3120_TIMER,
                                             1000, APCI3120_ENABLE) == 0)
            {
                if (i_APCI3120_StartTimerWatchdog (b_BoardHandle) == 0)
                {
                    while (ui_TimerIntCpt == 0);
                    printf ("Receive timer interrupt");
                    i_APCI3120_StopTimerWatchdog (b_BoardHandle);
                }
                else
                {
                    printf ("Start timer error");
                }
            }
            else
            {
                printf ("Init timer error");
            }
            i_APCI3120_ResetBoardIntRoutine (b_BoardHandle);
        }
        else
        {
            printf ("Interrupt routine initialisation error");
        }
        i_APCI3120_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```


d) Example in C for Windows NT/95/98 (asynchronous mode)

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

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_APCI3120_SetBoardIntRoutineWin32 (b_BoardHandle, APCI3120_ASYNCHRONOUS_MODE,
                                                0, NULL, v_InterruptRoutine) == 0)
        {
            ui_TimerIntCpt = 0;
            if (i_APCI3120_InitTimerWatchdog (b_BoardHandle, APCI3120_TIMER,
                                              1000, APCI3120_ENABLE) == 0)
            {
                if (i_APCI3120_StartTimerWatchdog (b_BoardHandle) == 0)
                {
                    while (ui_TimerIntCpt == 0);
                    printf ("Receive timer interrupt");
                    i_APCI3120_StopTimerWatchdog (b_BoardHandle);
                }
                else
                    printf ("Start timer error");
            }
            else
                printf ("Init timer error");
            i_APCI3120_ResetBoardIntRoutine (b_BoardHandle);
        }
        else
            printf ("Interrupt routine initialisation error");
        i_APCI3120_CloseBoardHandle (b_BoardHandle);
    }
    else
        printf ("Initialisation error");
}
```

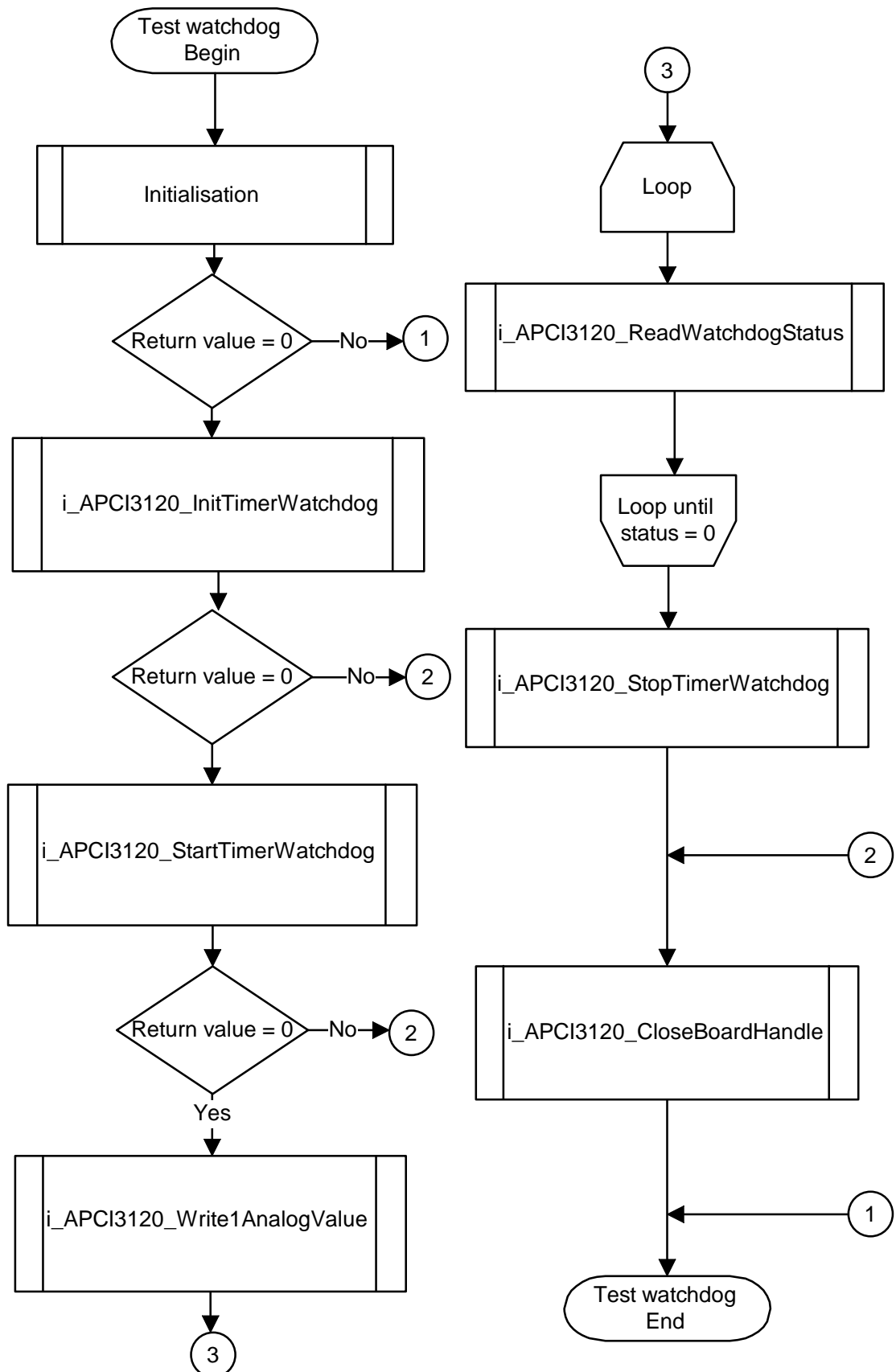

e) Example in C for Windows NT/95/98 (synchronous mode)

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

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_APCI3120_SetBoardIntRoutineWin32 (b_BoardHandle, APCI3120_SYNCHRONOUS_MODE,
            sizeof(str_UserStruct), (void **) &ps_GlobalUserStruct, v_InterruptRoutine) == 0)
        {
            ps_GlobalUserStruct->ui_TimerIntCpt = 0;
            if (i_APCI3120_InitTimerWatchdog (b_BoardHandle, APCI3120_TIMER,
                1000, APCI3120_ENABLE) == 0)
            {
                if (i_APCI3120_StartTimerWatchdog (b_BoardHandle) == 0)
                {
                    while (ps_GlobalUserStruct->ui_TimerIntCpt == 0);
                    printf ("Receive timer interrupt");
                    i_APCI3120_StopTimerWatchdog (b_BoardHandle);
                }
                else
                {
                    printf ("Start timer error");
                }
            }
            else
            {
                printf ("Init timer error");
            }
            i_APCI3120_ResetBoardIntRoutine (b_BoardHandle);
        }
        else
        {
            printf ("Interrupt routine initialisation error");
        }
        i_APCI3120_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```


11.6.2 Testing the watchdog

a) Flow chart



b) Pin assignment

Test the watchdog without using the interrupt.

Set a voltmeter between pin 31 (-) and pin 13 (+)

While testing, the output channel is set to 10 V and reset per watchdog.

c) Example in C

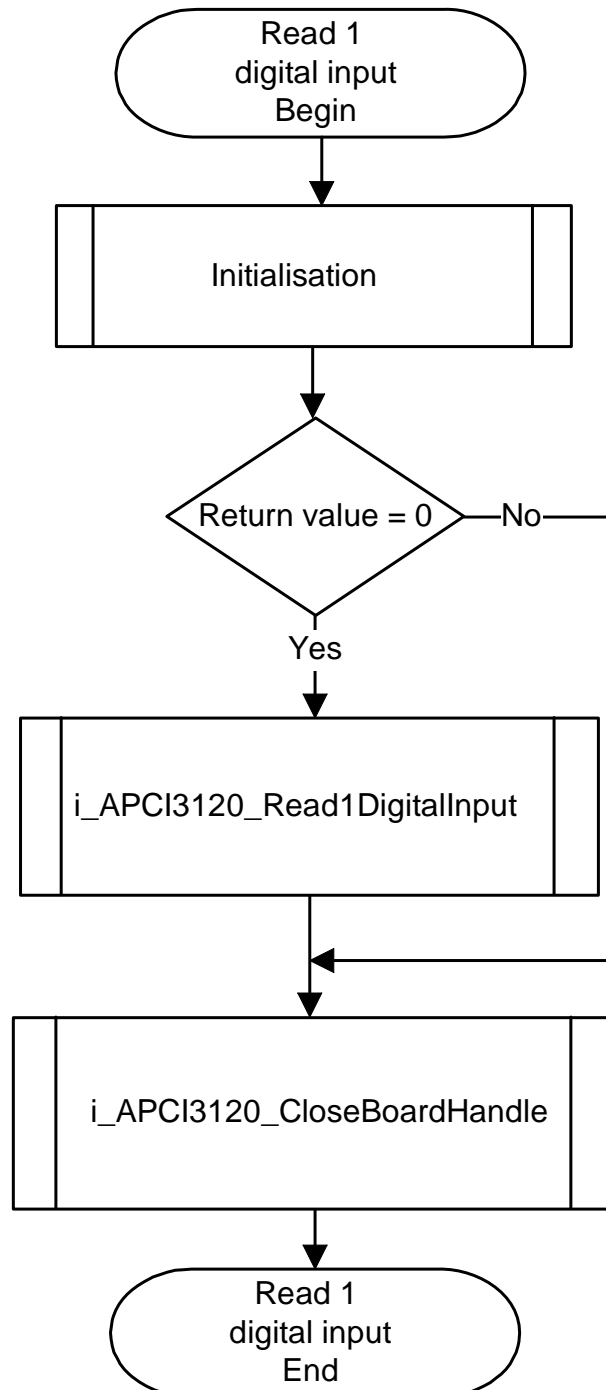
```
void main (void)
{
    unsigned char b_BoardHandle;
    unsigned char b_WatchdogStatus;

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_APCI3120_InitTimerWatchdog (b_BoardHandle, APCI3120_WATCHDOG,
                                           1000, APCI3120_DISABLE) == 0)
        {
            if (i_APCI3120_StartTimerWatchdog (b_BoardHandle) == 0)
            {
                i_APCI3120_WriteAnalogValue (b_BoardHandle, 1, APCI3120_UNIPOLAR, 8192);
                do
                {
                    i_APCI3120_ReadWatchdogStatus (b_BoardHandle, &b_WatchdogStatus);
                }
                while (b_WatchdogStatus == 0);
                printf ("Receive timer interrupt");
                i_APCI3120_StopTimerWatchdog (b_BoardHandle);
            }
            else
            {
                printf ("Start watchdog error");
            }
        }
        else
        {
            printf ("Init watchdog error");
        }
        i_APCI3120_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```


11.7 Digital input channels

11.7.1 Reading a digital input channel

a) Flow chart



b) Pin assignment

Set the digital input 1 to 24 V.

Pin assignment on the 37-pin SUB-D male connector:

- on pin 4

+ on pin 23

The test result is 1.

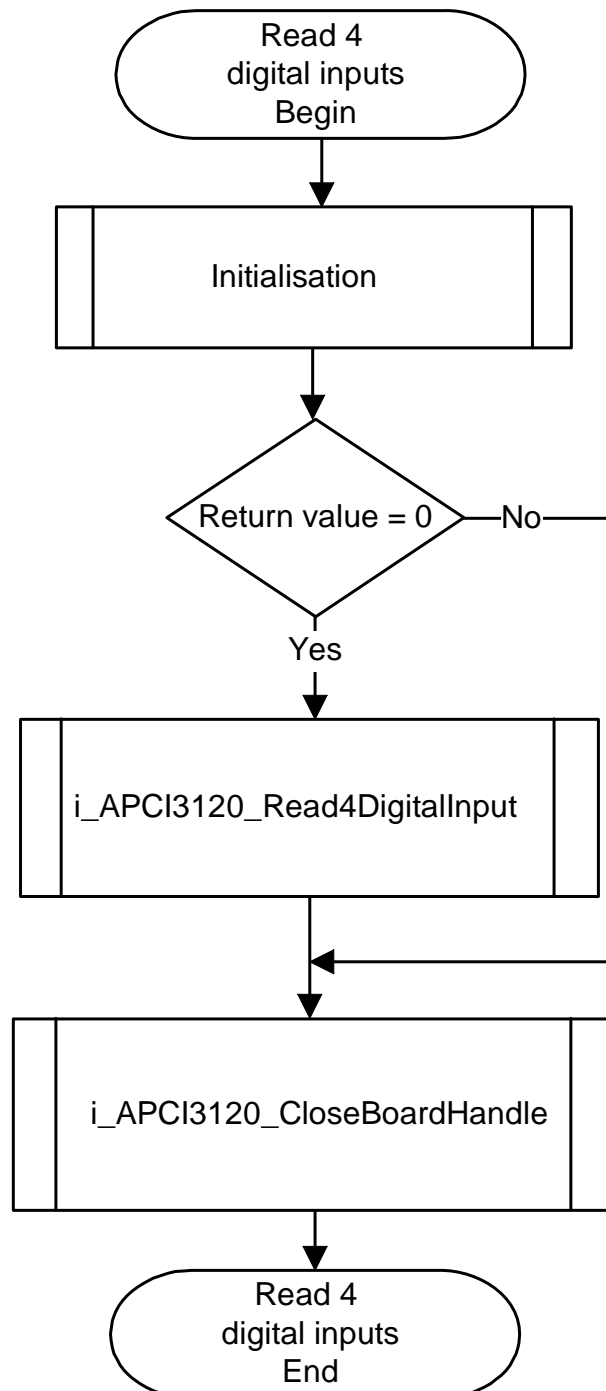
c) Example in C

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

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_APCI3120_Read1DigitalInput (b_BoardHandle,
                                          1
                                          &ui_ReadValue) == 0)
        {
            printf ("ui_ReadValue = %u", ui_ReadValue);
        }
        else
        {
            printf ("Read value error");
        }
        i_APCI3120_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```


11.7.2 Reading 4 digital input channels

a) Flow chart



b) Pin assignment

Set all digital input channels to 24 V.

Pin assignment on the 37-pin SUB-D male connector:

- on pin 4, 3, 2, 1

+ on pin 23, 22, 21, 20

The test result is 15.

c) Example in C

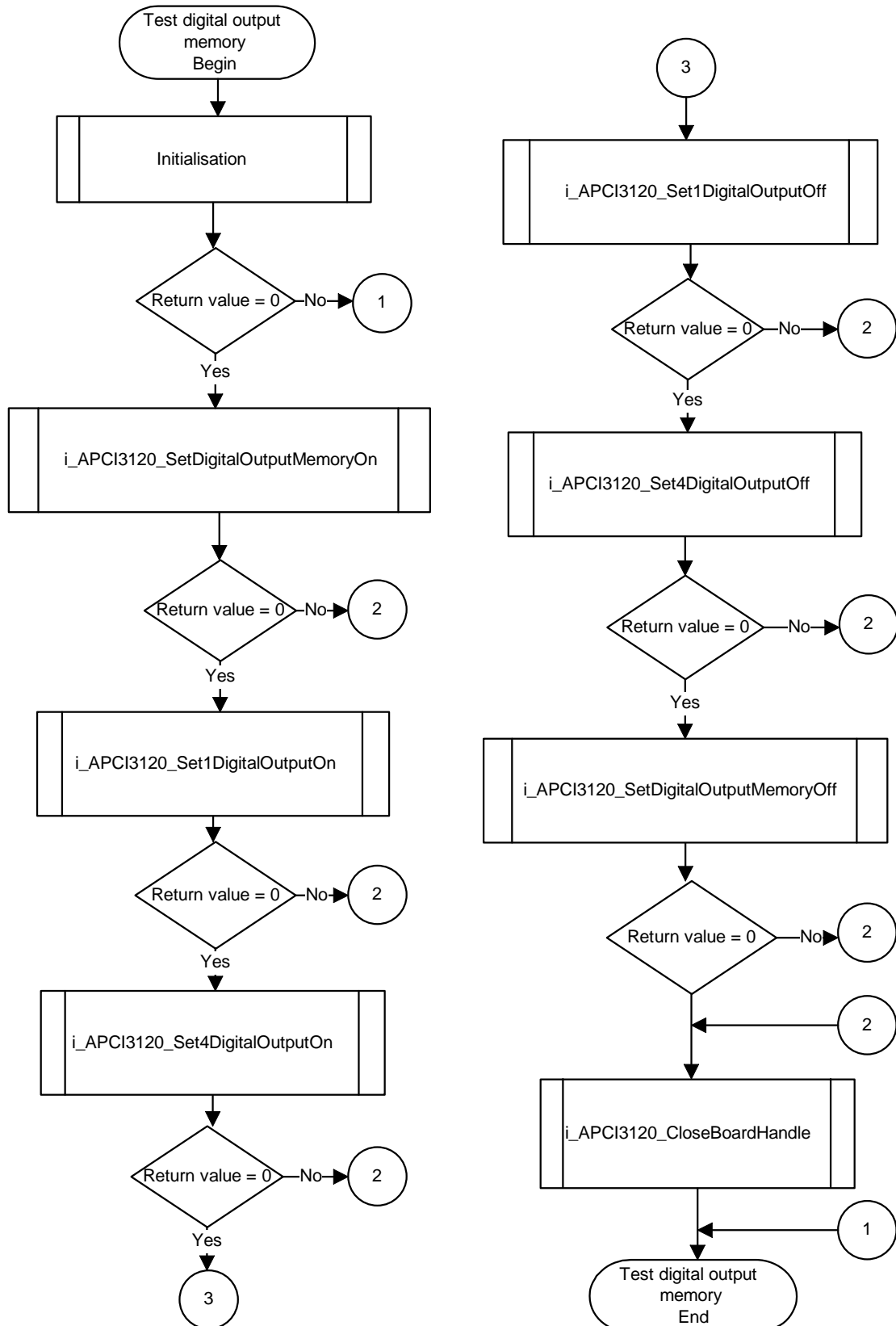
```
void main (void)
{
    unsigned char b_BoardHandle;
    unsigned int  ui_ReadValue;

    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_APCI3120_Read4DigitalInput      (b_BoardHandle,
                                                &ui_ReadValue)  == 0)
        {
            printf ("ui_ReadValue = %u", ui_ReadValue);
        }
        else
        {
            printf ("Read value error");
        }
        i_APCI3120_CloseBoardHandle (b_BoardHandle);
    }
    else
    {
        printf ("Initialisation error");
    }
}
```


11.8 Digital output channels

11.8.1 Testing the digital output memory

a) Flow chart



b) Pin assignment

Set a voltmeter between pin 4 (-) and pin 23 (+):

After the message „Output 1 is set“, measure the voltage. It must be 24V.

After the message „All outputs are set“, measure all output channels. It must be 24V.

After the message „All outputs are reset“, measure the voltage. It must be 0V.

c) Example in C

```
void main (void)
{
    unsigned char b_BoardHandle;
    if (Initialisation (&b_BoardHandle) == 0)
    {
        if (i_APCI3120_SetOutputMemory On (b_BoardHandle) == 0)
        {
            printf ("Digital output memory is activated");
            if (i_APCI3120_Set1DigitalOutputOn (b_BoardHandle, 1) == 0)
            {
                printf(" Output 1 is set ");
                getch();
                if (i_APCI3120_Set4DigitalOutputOn (b_Boardhandle, 14) ==0)
                {
                    printf ("All Output are set ");
                    getch();
                    if (i_APCI3120_Set1DigitalOutputOff (b_Boardhandle, 1 ) == 0)
                    {
                        printf ("Output 1 is reset");
                        getch();
                        if (i_APCI3120_Set4DigitalOutputOff (b_Boardhandle, 14) == 0)
                        {
                            printf ("Output 2,3,4 are reset");
                            if (i_APCI3120_SetOutputMemoryOff (b_Boardhandle) == 0)
                                printf ("Digital Output Memory deactivated");
                            else printf ("Digital Output Memory off error");
                        }
                        else printf ("Reset 4 digital Output error");
                    }
                    else printf ("Reset 1 digital output error ");
                }
                else printf ("Set 4 digital output error");
            }
            else printf ("Set 1 digital output error");
        }
        else printf ("Set Digital Output Memory On error");
        i_APCI3120_CloseBoardHandle (b_BoardHandle);
    }
    else printf ("Initialisation error");
}
```