DIN EN ISO 9001:2000
certified

ADDI-DATA GmbH
Dieselstraße 3
D-77833 OTTERSWEIER

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

Technical description

APCI-3600

Noise and vibration measurement board, optically isolated

Edition: 01.06 - 08/2006

Product information

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

Warranty and liability

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

Copyright

ADDI-DATA software product licence

Please read this licence carefully before using the standard software. The customer is only granted the right to use this software if he/she agrees with the conditions of this licence.
The software must only be used to set up the ADDI-DATA boards.
Reproduction of the software is forbidden (except for back-up and for exchange of faulty data carriers). Disassembly, decompilation, decryption and reverse engineering of the software are forbidden. This licence and the software may be transferred to a third party if this party has acquired a board by purchase, has agreed to all the conditions in this licence contract and the original owner does not keep any copies of the software.

Trademarks

WARNING

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

- **Personal injury**
- **Damage to the board, 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.

Figures

Tables

1 DEFINITION OF APPLICATION

1.1 Intended use

The board **APCI-3600** must be inserted in a PC with PCI 5V or 3.3 V (32/64-bit) slots, which is used as electrical equipment for measurement, control and laboratory pursuant to the norm IEC 61010-1.

1.2 Usage restrictions

The **APCI-3600** board must not to be used as safety related part for securing emergency stop functions.

The **APCI-3600** board must not be used in potentially explosive atmospheres.

1.3 General description of the board

The acquisition of analog with the board **APCI-3600** occurs via co-axial cables that must be connected to the SMB co-axial male connector of the **APCI-3600**, whereas the analog outputs must be connected to a SUB-D male connector.

The connection with our standard cable **ST3600**complies with the following specifications:
- metallized plastic hoods
- shielded cable
- cable shield folded back and firmly screwed to the connector housing.

The board shows the following characteristics:
APCI-3600:
- 8 (4x2) analog inputs (chapter 8.3)
- 2 analog outputs (chapter 8.4)
- 4 chronometer inputs (chapter 8.5)
- 8 digital inputs (chapter 8.6)
- 8 digital outputs (chapter 8.7)
- 2 external clocks (chapter 8.8)
- On Board SDRAM (chapter 8.9)

APCI-3600-L:
- 8 (4x2) analog inputs (chapter 8.3)
- On Board SDRAM (chapter 8.9)

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

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

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.

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

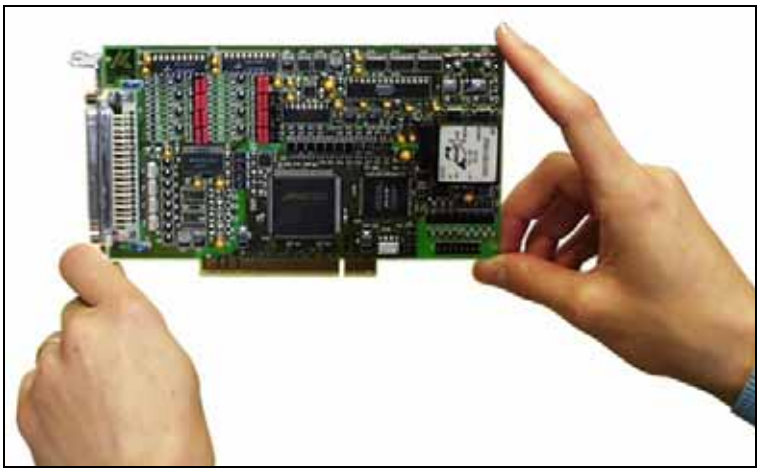
Only persons trained in electronics are entitled to perform the following works:
- installation
- use
- maintenance

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



4 TECHNICAL DATA

4.1 Electromagnetic compatibility (EMC)

- Preliminary -

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

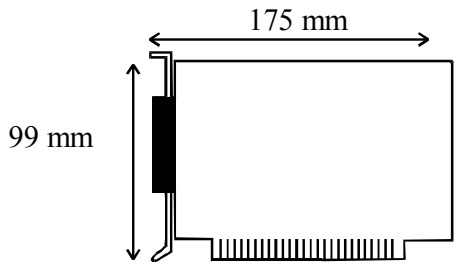
The board has been subjected to EMC tests in an accredited laboratory. The board complies with the limit values set by the norms IEC61326 as follows:

	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 6-layer printed circuit card.

Dimensions:



Weight: ..... approx. 170 g
Installation in: ..... 32/64 bit PCI slot 5 V

Connection to the peripheral:
Front connector: ..... 8 x SMB co-axial male connector
Additional connector:.....3 additional front plane openings
- 37-pin SUB-D male connector
- 2 x 15-pin SUB-D male connector
- 6-pin male connector

Accessories[1]:
ST3601:.....Connection cable RG 178, 2 m
SMB co-axial female connector on
BNC male connector

1 Not included in the standard delivery.

Ribbon cable:

Ribbon cable **FB3600-D**: ..... For the connection of the digital I/O:
From 26-pin connector on 37-pin SUB-D male connector

Ribbon cable **FB3600-AC**:..... 1) For the connection of the analog outputs:
From 14-pin connector on 15-pin SUB-D male connector
2) For the connection of the chronometer:
From 14-pin SUB-D connector on 15-pin SUB-D male connector

4.3 Versions

The board is available in 2 versions:

APCI-3600: .....
- 8 analog inputs
- 4 current sources for the connection of ICP sensors
- 2 analog outputs
- 4 chronometer inputs
- 8 digital inputs, 24 V
- 8 digital outputs, 24 V
- 128 MBytes SDRAM

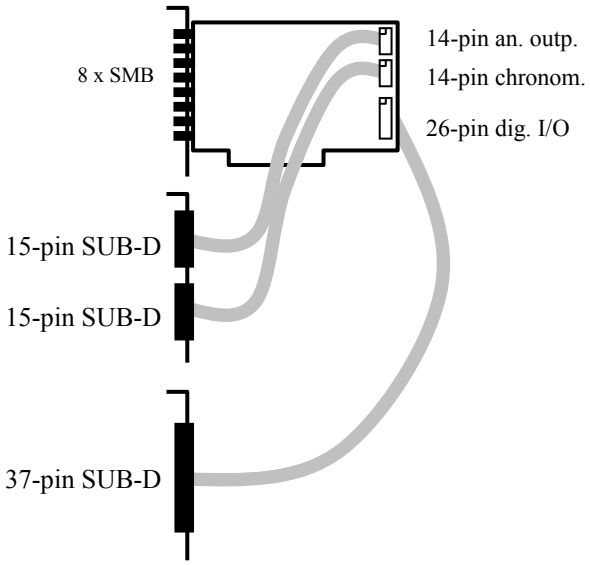
APCI-3600-L: .....
- 8 analog inputs
- 4 current sources for the connection of ICP sensors
- 128 Mbytes SDRAM

4.4 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:
PCI BIOS from Version 1.0
Bus speed: ..... < 33 MHz
Operating system: ..... Windows 2000, XP (further on request)

Fig. 4-1: Required slots



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

	APCI-3600	APCI-3600-L
+ 5 V from PC	Not available	Not available

4.4.1 Analog inputs

Number: 8
Input type: ..... Single-ended or differential (selectable through software)
Resolution: 24-bit
A/D converter: ..... Delta-Sigma, 5th order, Multibit-Delta-Sigma modulator
Gain: x1, x10
Input ranges: ..... ± 10 V single-ended (gain x 1)
± 1 V single-ended (gain x 10)
± 5 V differential (gain x 1)
± 0.5 V differential (gain x 10)

Sampling frequency f_s :..... 2 kHz ≤ f_s ≤ 200 kHz
Selectable through software (see table below)

Table 4-1: Selectable frequencies

2 kHz ≤ f_s ≤ 50 kHz	50 kHz ≤ f_s ≤ 100 kHz	100 kHz ≤ f_s ≤ 200 kHz
50000 Hz	100000 Hz	200000 Hz
40000 Hz	80000 Hz	160000 Hz
33333 Hz	66667 Hz	133333 Hz
25000 Hz	50000 Hz	100000 Hz
20000 Hz		
16667 Hz		
12500 Hz		
10000 Hz		
8000 Hz		
5000 Hz		
4000 Hz		
3333 Hz		
2500 Hz		
2000 Hz		

Oversampling: ..... 64 x f_s (for sampling frequency f_s)
Frequency precision: ..... ± 50 ppm
FIFO depth: 128 DWORD, for the right and left channel of the same ADC
Data transfer: ..... DMA, I/O, IRQ

Throughput range ripple (rel. to 1 kHz), max., DC coupled:
2 kHz ≤ f_S ≤ 50 kHz..... -0.1dB, DC to 0.47 x f_s
50 kHz ≤ f_S ≤ 100 kHz..... -0.1dB, DC to 0.45 x f_s
100 kHz ≤ f_S ≤ 200 kHz..... -0.1dB, DC to 0.24 x f_s

-3 dB bandwidth:
2 kHz ≤ f_S ≤ 50 kHz..... 0.5 x f_s
50 kHz ≤ f_S ≤ 100 kHz..... 0.5 x f_s
100 kHz ≤ f_S ≤ 200 kHz..... 0.358 x f_s

Input coupling: ..... AC, DC, GND,
Selectable through software
AC –3 dB limit frequency: ..... 1.6 Hz

Overvoltage protection
R0-, L0-, R1-, L1-, L/R2+-, L/R3+-
Max. direct current: ..... ± 12 V, ± 200 mA
Max. peak current
(Impulse at 1ms, 10% duty cycle): ..... ± 12 V, ± 300 mA
R0+, L0+, R1+, L1+
Max. direct current: ..... ± 36 V, ± 30 mA
Max. peak current
(Impulse at 1ms, 10% duty cycle): ..... ± 36 V, ± 70 mA

ESD protection: > 2 kV, ESD protection through method 3015.17

Dynamic characteristics
2 kHz ≤ f_S ≤ 50 kHz
Alias free bandwidth (Passband)..... DC (0Hz) to 0.47 xf_s, min. to max.
Stopband..... 0.58 x f_s min
Stopband attenuation (Alias rejection): ..... -95 dB min
Filter delay by ADC
(Total group delay): ..... 12/f_s s typ.

50 kHz ≤ f_S ≤ 100 kHz
Alias free bandwidth (Passband)..... DC (0Hz) to 0.45 xf_s, min. to max.
Stopband..... 0.68 x f_s min
Stopband attenuation (Alias rejection): ..... -92 dB min
Filter delay by ADC
(Total group delay): ..... 9/f_s s typ.

100 kHz ≤ f_S ≤ 200 kHz
Alias free bandwidth (Passband)..... DC (0Hz) to 0.24 xf_s, min. to max.
Stopband..... 0.78 x f_s min
Stopband attenuation (Alias rejection): ..... -97 dB min
Filter delay by ADC
(Total group delay): ..... 5/f_s s typ.

Dynamic range
SNR:
2 kHz ≤ f_S ≤ 50 kHz: .....< -105 dB (short input, gain x1)
< -100 dB (short input, gain x10)
< -80 dB (open input, gain x1)
< -60 dB (open input, gain x10)

50 kHz ≤ f_S ≤ 100 kHz: ..... < -105 dB (short input, gain x1)
< -100 dB (short input, gain x10)
< -80 dB (open input, gain x1)
< -60 dB (open input, gain x10)
100 kHz ≤ f_S ≤ 200 kHz ..... < -75 dB (short input, gain x1)
< -75 dB (short input, gain x10)
< -75 dB (open input, gain x1)
< -60 dB (open input, gain x10)

Crosstalk (oversampling) (between channel R0 and L0, R1 and L1, R2 and L2, R3 and L3 at gain x1):
Short input at f_a = 100 Hz
2 kHz ≤ f_S ≤ 50 kHz: ..... < -105 dB
50 kHz ≤ f_S ≤ 100 kHz: ..... < -100 dB
100 kHz ≤ f_S ≤ 200 kHz ..... < -75 dB

Short input at f_a = 1 kHz:
2 kHz ≤ f_S ≤ 50 kHz: ..... < -100 dB
50 kHz ≤ f_S ≤ 100 kHz: ..... < -100 dB
100 kHz ≤ f_S ≤ 200 kHz ..... < -75 dB

1 kΩ Last at f_a = 100 Hz:
2 kHz ≤ f_S ≤ 50 kHz: ..... < -100 dB
50 kHz ≤ f_S ≤ 100 kHz: ..... < -100 dB
100 kHz ≤ f_S ≤ 200 kHz ..... < -75 dB

1 kΩ Last at f_a = 1 kHz:
2 kHz ≤ f_S ≤ 50 kHz: ..... < -100 dB
50 kHz ≤ f_S ≤ 100 kHz: ..... < -100 dB
100 kHz ≤ f_S ≤ 200 kHz ..... < -75 dB

Phase error (between channel R0 and L0, R1 and L1, R2 and L2, R3 and L3):
At f_s=200 kHz: ..... 0.3° (max.)
0.2° at f_a = 10 kHz sine signal
0.02° at f_a = 1 kHz sine signal

Amplitude error: ..... ± 0.02 dB, max. at f_s 1 kHz
Sine signal at gain x1 and x10

Offset error ..... ± 200 µV, max. at f_s = 2 kHz

4.4.2 Analog outputs

Number of outputs: 2
Resolution: 16-bit

Precision: 13-bit
DAC type: R-2R
Output range: ..... ± 10 V

Settling time:
10 V Step, R_L = 2 kΩ, C_L = 1500 pF
± 0,1%: ..... 5 µs typ.
± 0,01%: ..... 5.6 µs typ.
Overvoltage protection: ..... ± 12 V, 100 mA max. direct current
Short circuit current: ..... ± 45 mA typ.
Output voltage after reset: ..... 0 V
FIFO depth: ..... 256 Word
Data transfer: ..... DMA, IO, IRQ

4.4.3 Digital inputs

Number of inputs: 8
Filter/protection: ..... Low-pass/Transorb diode
Optical isolation: ..... 1000 V
Nominal voltage: ..... 24 V external
Input voltage: ..... 0 to 30 V
Input current: ..... 7 mA at 24 VDC, typ.

Logic input level:
UH (max): ..... 30 V
UH (min): ..... 19 V
UL (max): ..... 14 V
UL (min): ..... 0 V
Input frequency (max.): ..... 5 kHz at 24 V
Trigger input: ..... digital input 0

4.4.4 Digital outputs

Number of outputs: 8
Optical isolation: ..... 1000 V
Output type: ..... Open collector
Nominal voltage: ..... 24 V
Distribution voltage: ..... 5-30 V
Output current for each output: ..... 50 mA max.
Current: .....300 mA limited through PTC
Switching-on time: .....0.25 µs typical
Switching-off time: .....0.25 µs typical

4.4.5 Current sources

Number: 4
Type: ..... Constant current source for the supply of the ICP[1] sensors
Current: ..... 4 mA typ.
Standard: ..... 24 V max.

4.4.6 Chronometer

Number: ..... 4 chronometers, 2 gate on chronometer 0 and 1
Input type: RS485
Max. speed: ..... 1 MHz max.
Counter depth: 32-bit
Divisor: ..... From 2^0 to 2^{15} per counter input
FIFO depth: ..... 256 DWORD
Data transfer: ..... DMA, IO, IRQ
Differential Threshold Voltage: ..... -200 mV min
-50 mV max
Input resistance: ..... 120 Ω differential
ESD protection: ..... +/-15 kV Human Body Model

4.4.7 Master trigger, external clock

Type: ..... differential, ANSI TIA/EIA-644-1995 standard
Input voltage:
High level: ..... 2 V min.
Low level: ..... 0.8 V max.
Range of the differential input voltage, V_{ID} ..... 0.1 V min.
0.6 V max.
Common mode input voltage: ..... $|V_{ID}|/2$ min.
$2.4 - |V_{ID}|/2$ max.

Receiver (Slave board)
Differential input voltage
Positive switching threshold: ..... 50 mV min.
Negative switching threshold: ..... -50 mV max.

Input current for each input channel: ..... -11 µA typ., $V_I = 0$ V
-3 µA typ., $V_I = 2.4$ V
Input capacity: ..... 5 pF typ.

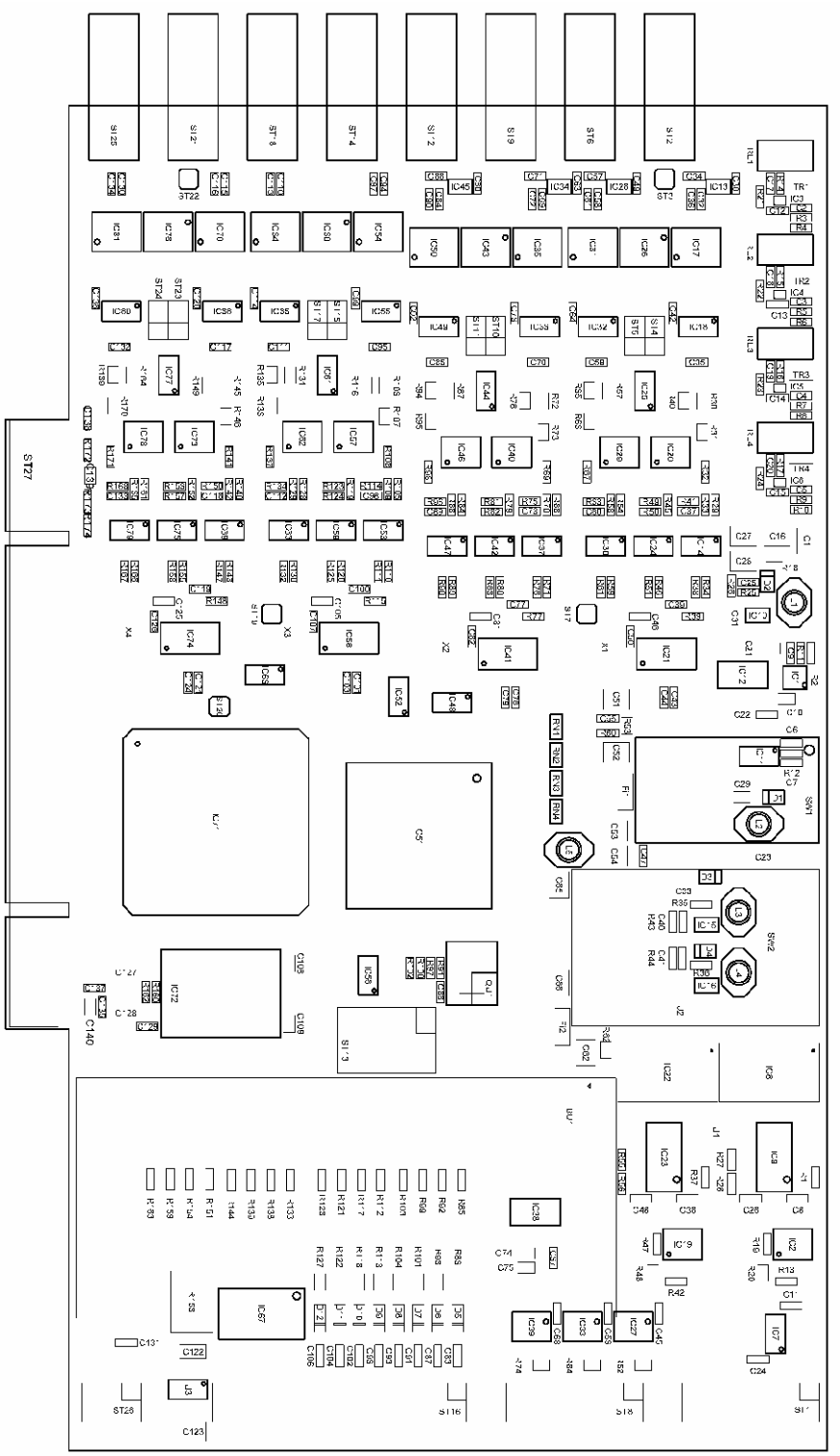
[1] ICP Sensor: (integrated circuit piezoelectric)

Driver (Master board)

Differential input voltage: ..... 340 mV typ.
Change of the differential input voltage
between logical levels ..... -50 mVmin.
50 mV max.
Short circuit current: ..... 10 mA max.
Output current (high impedance): ..... +/-1 µA max.

4.5 Component scheme

Fig. 4-2: Component scheme



5 INSTALLATION OF THE BOARD

IMPORTANT!
Do observe the safety precautions (yellow leaflet)!

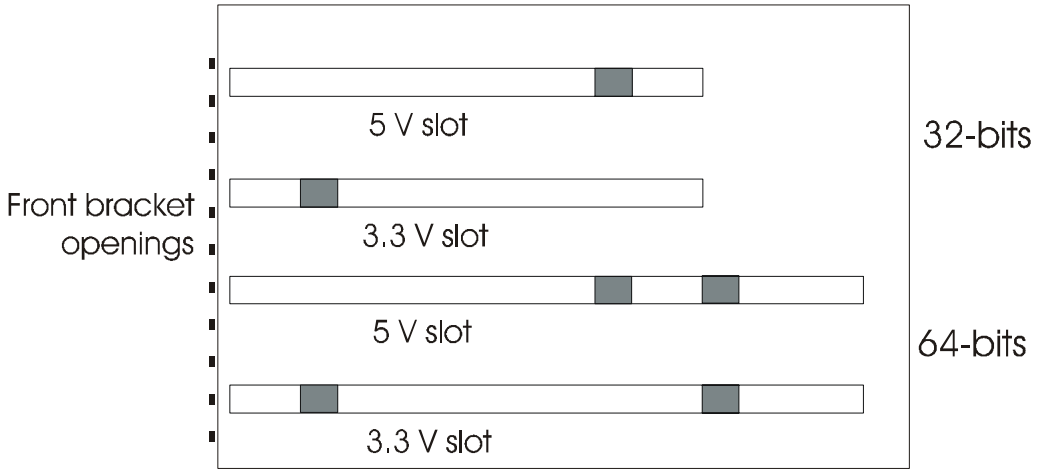
5.1 Opening the PC

- **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.**

5.2 Selecting a free slot

- **Insert the board into a free PCI-5V or PCI-3.3 V slot (32/64-bit).**

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

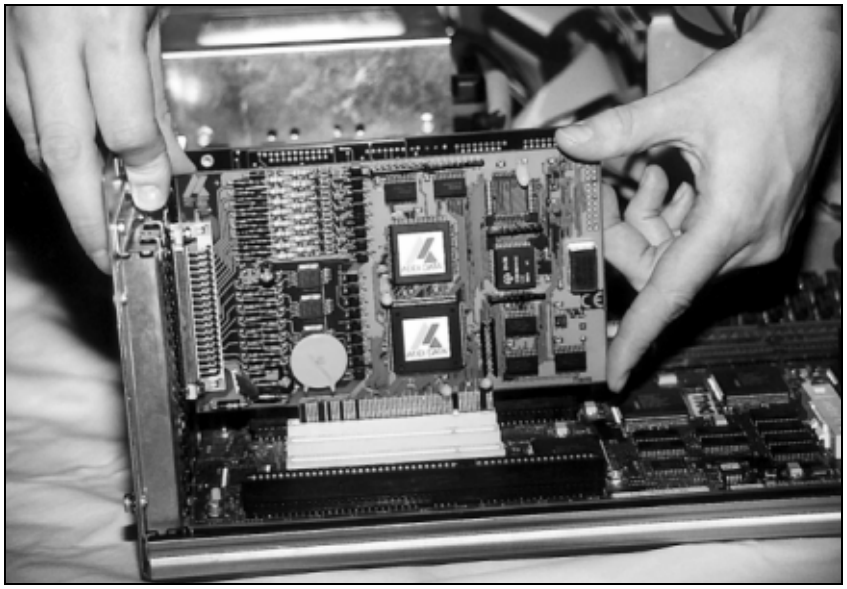


- **Remove the back cover of the selected slot according to the instructions of the PC manufacturer.** Keep the back cover. You will need it if you remove the board
- **Discharge yourself from electrostatic charges.**
- **Take the board out of its protective blister pack.**

5.3 Plugging the board into the slot

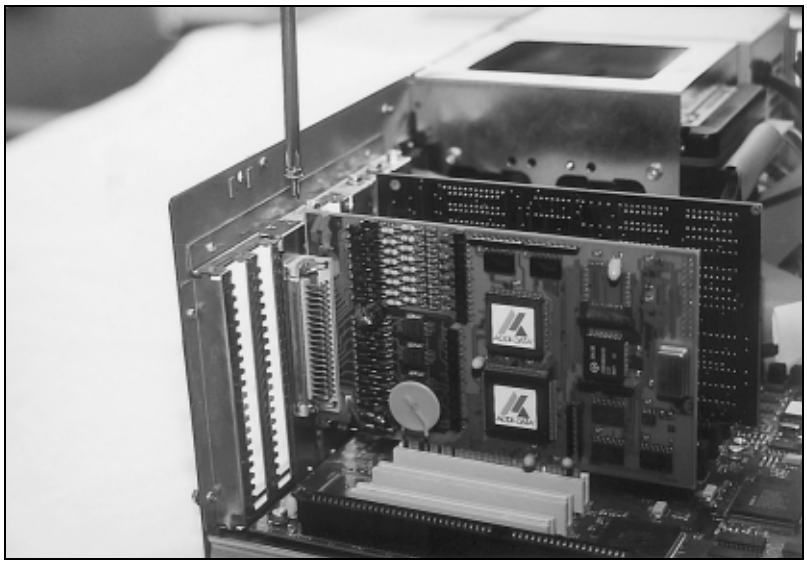
♦ Insert the board **vertically into the chosen slot.**

Abb. 5-1: Inserting the board



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

Fig. 5-2: Fastening the board at the back cover



♦ **Tighten all the loosen screws.**

5.4 Closing the PC

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

6 SOFTWARE

The following chapter describes the software and its application.

IMPORTANT!
You find the most important information about **Installing and Uninstalling of the different drivers** in the supplied brochure **"Installation instructions for the PCI and ISA bus".**

You will find a link to the respecting PDF file in the navigation window (bookmark) of Acrobat Reader.

The board is supplied with a driver-CD-ROM (CD 1), which contains among other things:
- The driver and the 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.

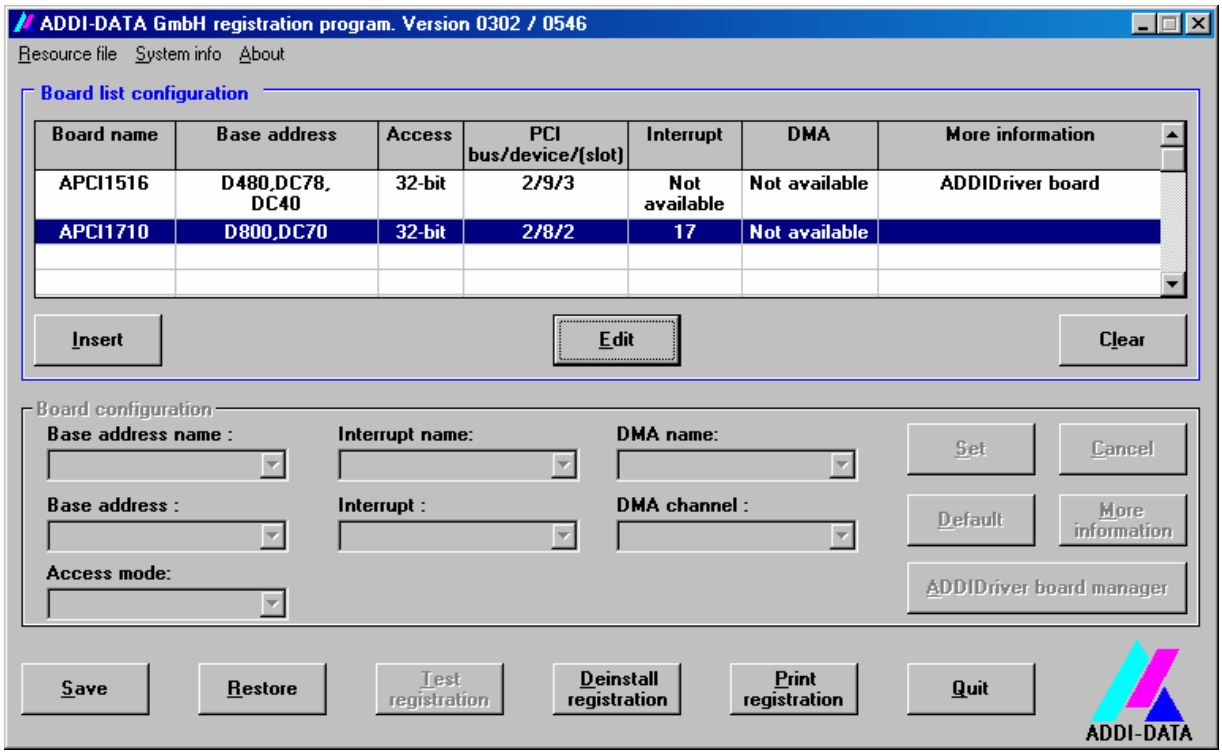
6.1 Board configuration with ADDIREG

6.1.1 Installation description

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.

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

Abb. 6-1: ADDIREG registration program (Example)



The table in the mid lists the boards that are registered and its parameters

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.

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.**

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:

Under the table are 6 text entry, with which you can change the board parameters

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.

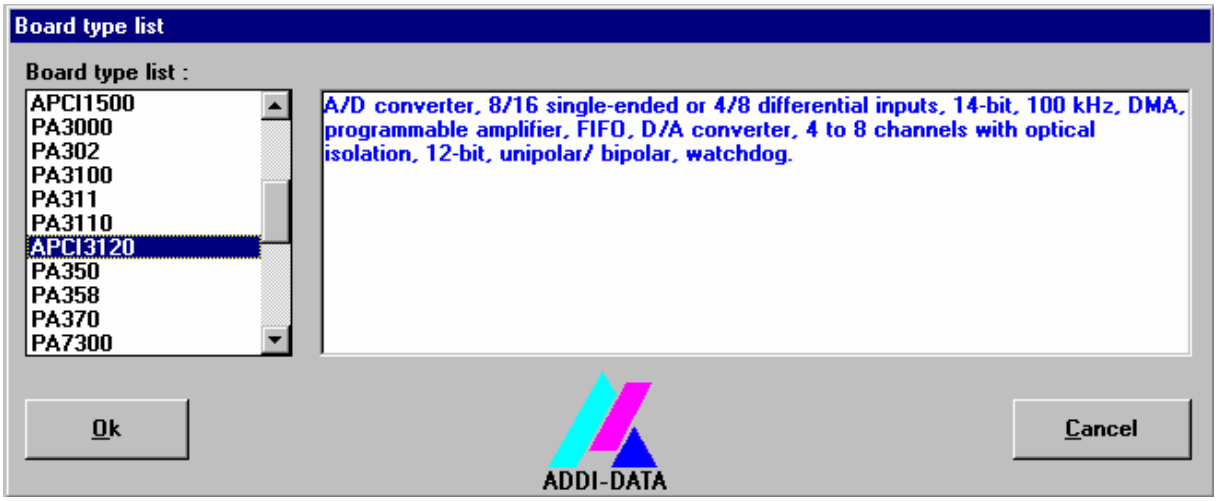
DMA channel (for ISA boards only):
Selection of the used DMA channel.

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. 6-1: Installing a new board



All boards you can register are listed on the left. Select the required board. (The corresponding line is highlighted).
On the right you can read technical information about the board (s).
Activate with "OK"; You return to the former screen.

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

Set:
Sets the parameterized board configuration. The configuration should be set before you save it.

Cancel:
Reactivates the former parameters of the saved configuration.

Default:
Sets the standard parameters of the board.

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

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

6.1.2 Button “More information”

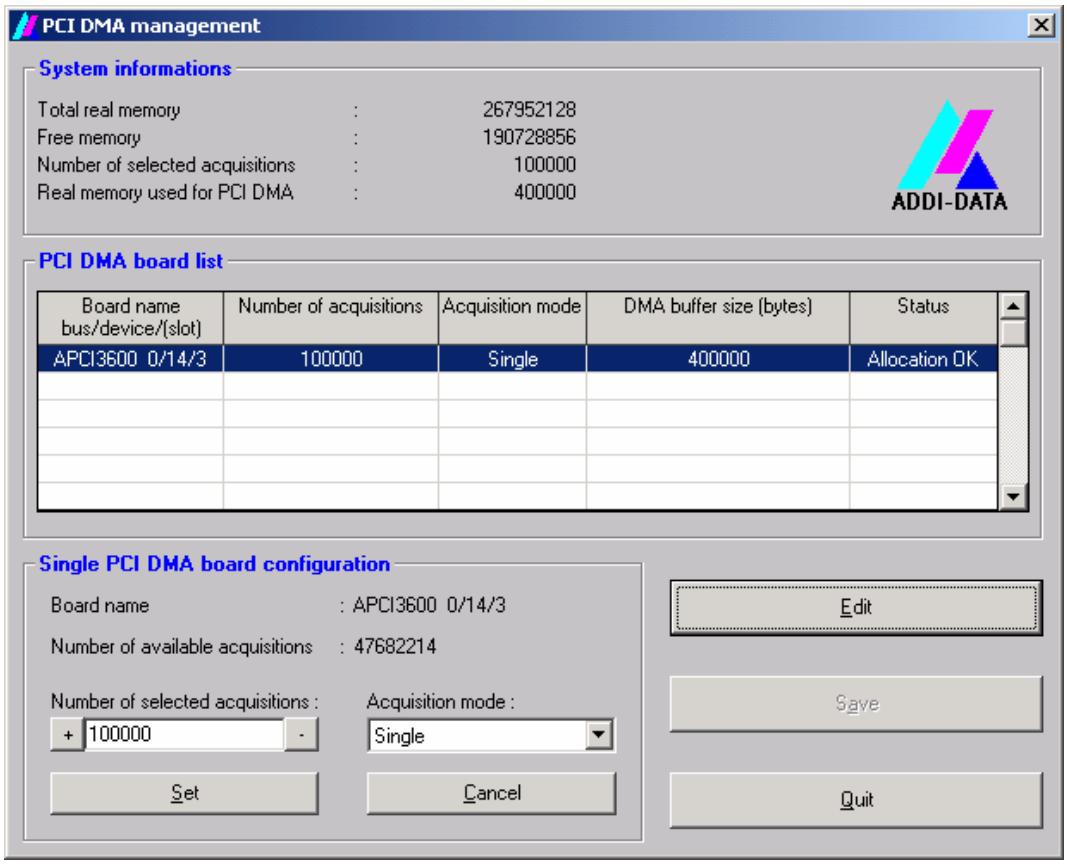
With this button you can change the board specific parameters like the identifier string, the COM number, the operating mode of a communication board, etc. If your board does not support this information, you cannot activate this button.

6.1.3 PCI analog input boards with DMA

If you have inserted an **APCI-3600** and click on “More information”, the dialog box shown below is displayed.
Below is the example of 100 000 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 4 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.
For a single acquisition, only one buffer is allocated.
For a continuous acquisition, two buffers are allocated.

Abb. 6-2: Management of PCI DMA boards



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.

IMPORTANT!
At the moment the „Continuous“ mode is not available.

Set:
Sets the user configuration.

Cancel:
Restores the former configuration

6.1.4 Registration of 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.**

♦ **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.
Confirm your selection with a click.

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

♦ **Save the configuration with "Save".**

You can test if the registration is "OK".
This test controls if the registration is right and if the board is present.
If the test has been successfully completed you can quit the ADDIREG program.
The board is initialised with the set parameters and can now be operated.

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

6.1.5 Changing the registration of a board

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

♦ **Call up the ADDIREG program.**

♦ **Select the board to be changed.**
The board parameters (Base address, DMA channel, ..) are listed in the lower fields.

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

♦ **Select a new value.**

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

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

♦ **Save the configuration with "Save".**
You can test if the registration is "OK".
This test controls if the registration is right and if the board is present.
If the test has been successfully completed you can quit the ADDIREG program.
The board is initialised with the set parameters and can now be operated.

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

6.1.6 Questions and software download in the internet

Do not hesitate to e-mail us your questions.

e-mail: info@addi-data.com or
hotline@addi-data.com

Download in the internet

You can download the latest version of the software for the board **APCI-3600**

http://www.addi-data.com

i

IMPORTANT!
Before using the board or in case of malfunction during operation, check if there is an update of the product (technical description, driver). The current version can be found on the internet or contact us directly.

7 CONNECTING THE PERIPHERAL

7.1 Connector pin assignment

Fig. 7-1: Co-axial SMB male connector on front plane

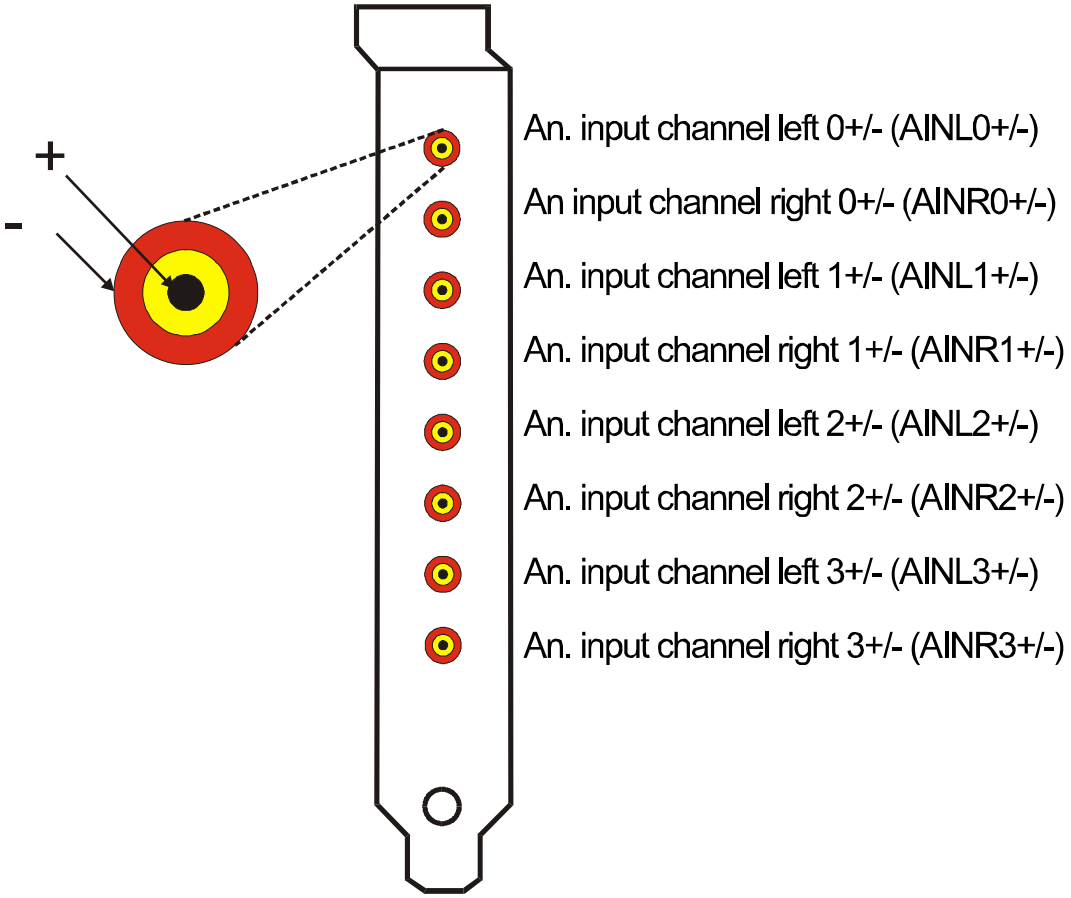


Fig. 7-2: Digital inputs and outputs: 26-pin connector on 37-pin SUB-D male connector (second front plane: Ribbon cable FB3600-D)

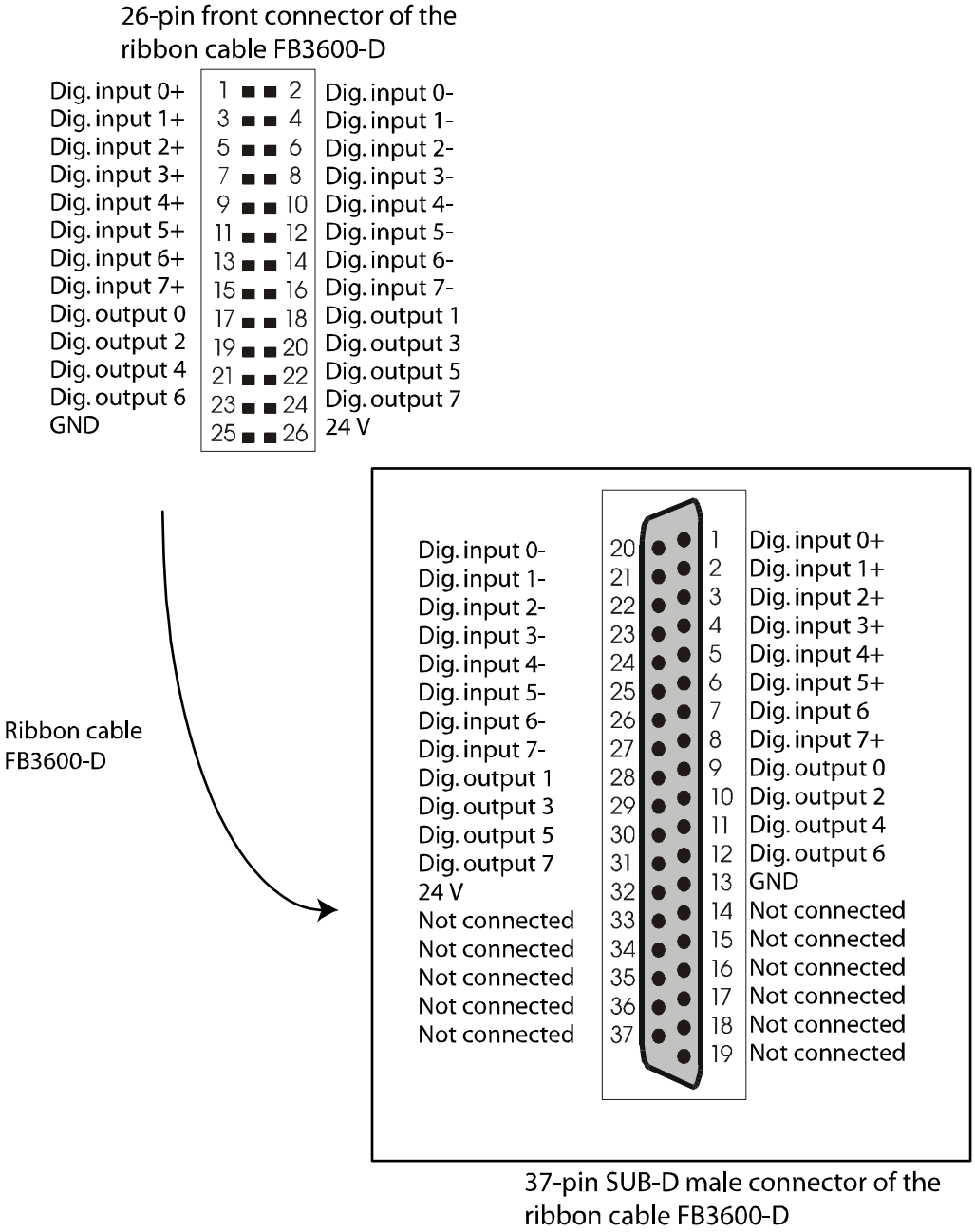


Fig. 7-3: Analog outputs: 14 pin front connector on 15 pin SUB-D female connector
Chronometer inputs: 14 pin front connector on 15 pin SUB-D male connector

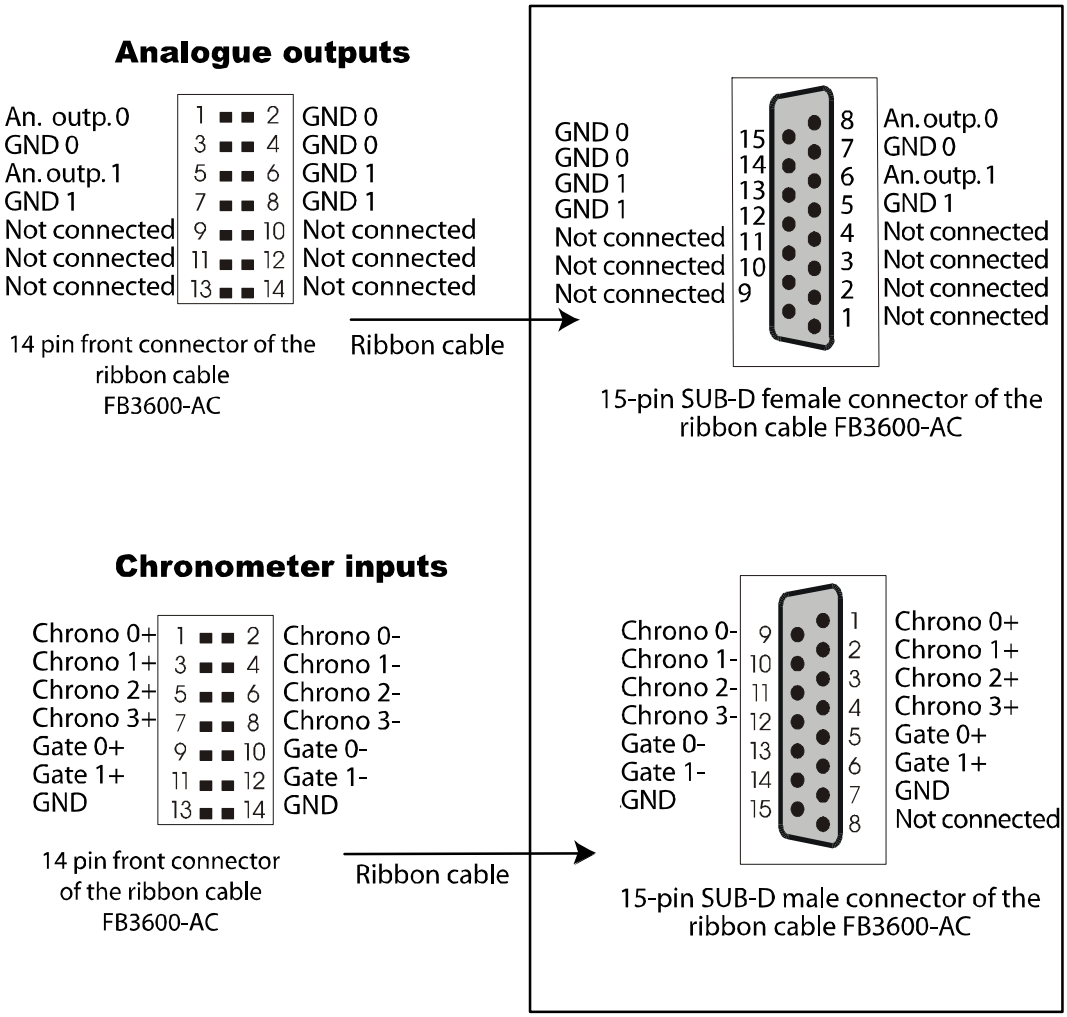
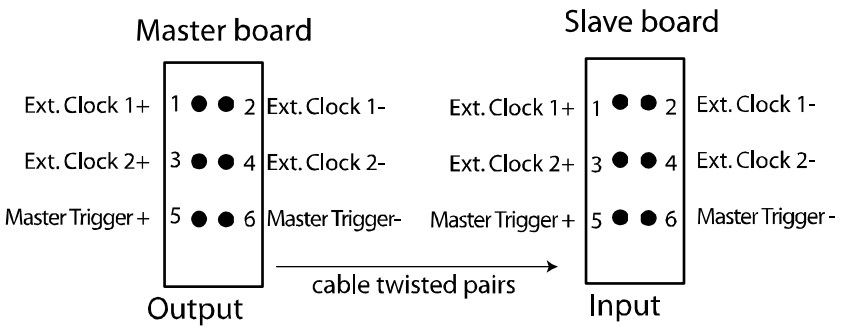


Fig. 7-4: External clock: connector



7.2 Version administration

There are two versions of the board **APCI-3600** (see also chapter 4.3):

APCI-3600: ..... - 8 analog inputs
- 4 current sources for the connection of the ICP sensors
- 2 analog outputs
- 4 chronometer inputs
- 8 digital inputs
- 8 digital outputs
- 128 MBytes SDRAM

APCI-3600-L: ..... - 8 analog inputs
- 4 current sources for the connection of the ICP sensors
- 128 MBytes SDRAM

8 FUNCTIONS OF THE BOARD

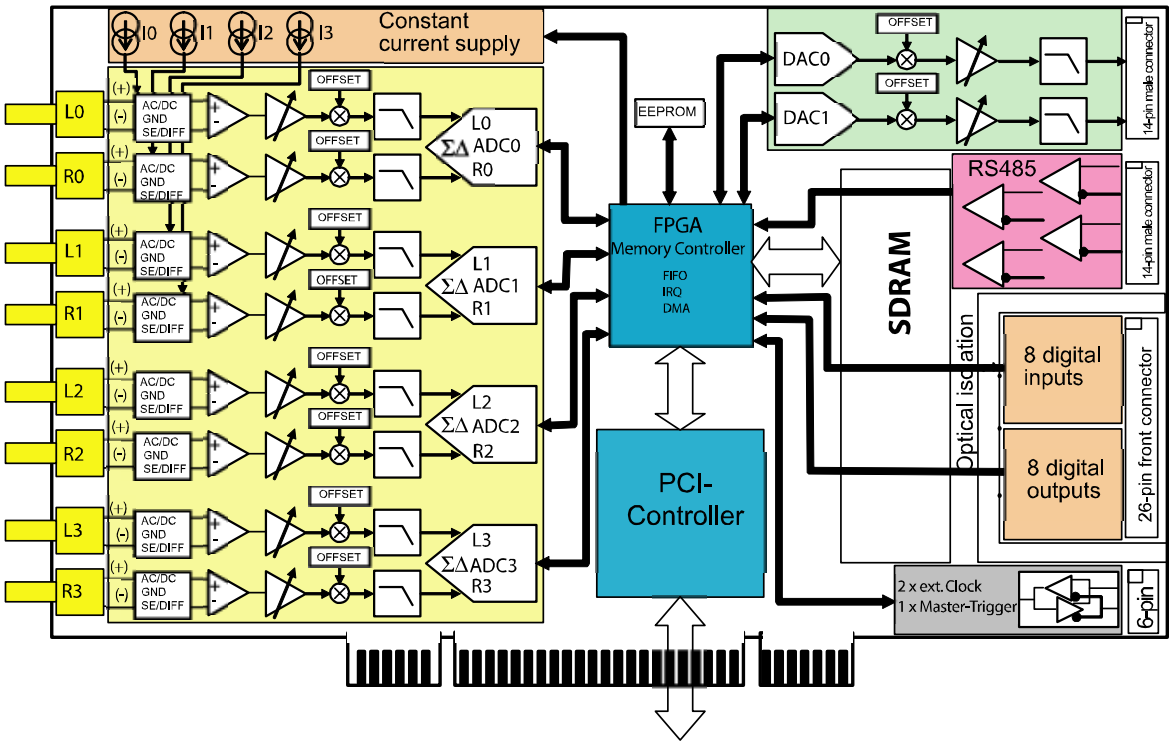
8.1 Overview

The **APCI-3600** is a multifunction board with On Board SDRAM for the acquisition and/or production of analog signals in the sound area. Hereto, you have the following functions, which will be described more detailed in the next chapters:

- 8 analog inputs (chapter 8.3)
- 2 analog outputs (chapter 8.4)
- 4 chronometer inputs (chapter 8.5)
- 8 digital inputs (chapter 8.6)
- 8 digital outputs (chapter 8.7)
- 2 external clocks (chapter 8.8)
- On board buffer SDRAM (chapter 8.9)

8.2 Block diagram

Fig. 8-1: Block diagram of the APCI-3600



8.3 Analog inputs

8.3.1 Distribution of the analog inputs

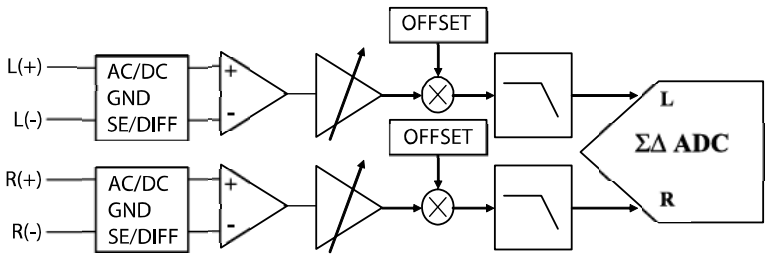
The **APCI-3600** has 8 analog inputs that are allocated in pairs at the 4 stereo A/D converters (ADCs). Each A/D converter has a left channel (L) and a right channel (R).

Table 8-1: Channels and A/D converter

Channels	A/D converter
L0, R0	ADC0
L1, R1	ADC1
L2, R2	ADC2
L3, R3	ADC3

Each stereo input consists of:
- An AC/DC/GND, Single-Ended (SE), differential (DIFF) switch
- An differential input gain
- An Offset-/Gain balancing circuit
- An analog low pass filter

Fig. 8-2: Input circuit of an analog input



8.3.2 Coupling mode

Each input can be coupled in the AC mode (alternative coupling) over a high-pass filter with a cut-off frequency of 0.16 Hz (-3 dB) or in the DC mode (direct coupling).

8.3.3 Configuration

Each input can either set as single-ended input (SE) or as differential input (DIFF).

8.3.4 Input range

The input range is:

In the SE mode: ± 10 V
In the DIFF mode: ± 5 V

8.3.5 Calibration

The **APCI-3600** allows a hardware calibration of the offset and gain errors on all 8 analog inputs (this calibration is done by ADDI-DATA)

8.3.6 Anti-aliasing filter

The **APCI-3600** has an anti-aliasing filter. The function of this filter will be described more detailed in the following chapters by giving basis information about this filter method:

a) Discrete sampling

b) Nyquist's criteria

c) Anti-aliasing filter

d) Analog filter

a) Discrete sampling of analog signals

The concept of discrete time sampling and quantization of an analog signal is as follows:
The continuous analog data must be sampled at discrete intervals (ts = 1/fs) which must be carefully chosen to insure an accurate representation of the original analog signal. It is clear that **the more samples taken (faster sampling rates), the more the digital representation.** But if fewer samples are taken (lower sampling rates), a point is reached where critical information about the signal is actually lost. This leads to the statement of the Nyquist's criteria, which is described in the following chapter.

b) Nyquist's criteria

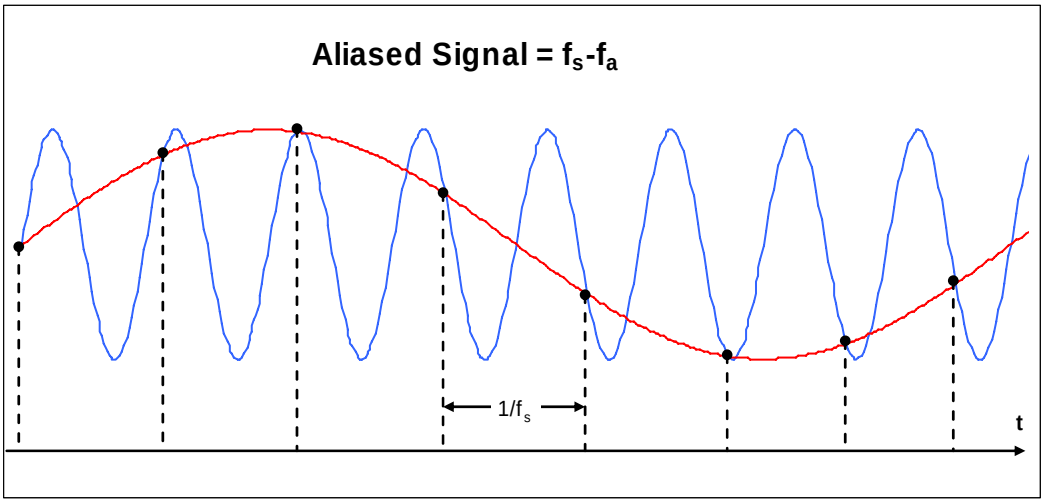
The Nyquist criteria says:

- A signal with a *bandwidth* f_a must be sampled at a rate $f_s > 2f_a$ or information about the signal will be lost.
- Aliasing occurs whenever $\mathbf{f_s < 2f_a}$.

The Nyquist criterion requires that the sampling frequency must be at least twice the signal bandwidth or information about the signal will be lost. If the sampling frequency is less than twice the analog bandwidth, a phenomenon known as "aliasing" will occur.
Aliasing implicates the time and frequency domain. These both aspects are explained in the following sections:

Time domain
In the time domain a single town sinewave sampled is represented (see Fig. 8-3: Aliasing: Time domain). In this example, the sampling frequency f_s is only slightly more than the analog input frequency and therefore the Nyquist's criteria is violated. Please note that that the pattern of the actual samples produces an aliased sinewave at a lower frequency equal to $f_s - f_a$.

Fig. 8-3: Aliasing: Time domain

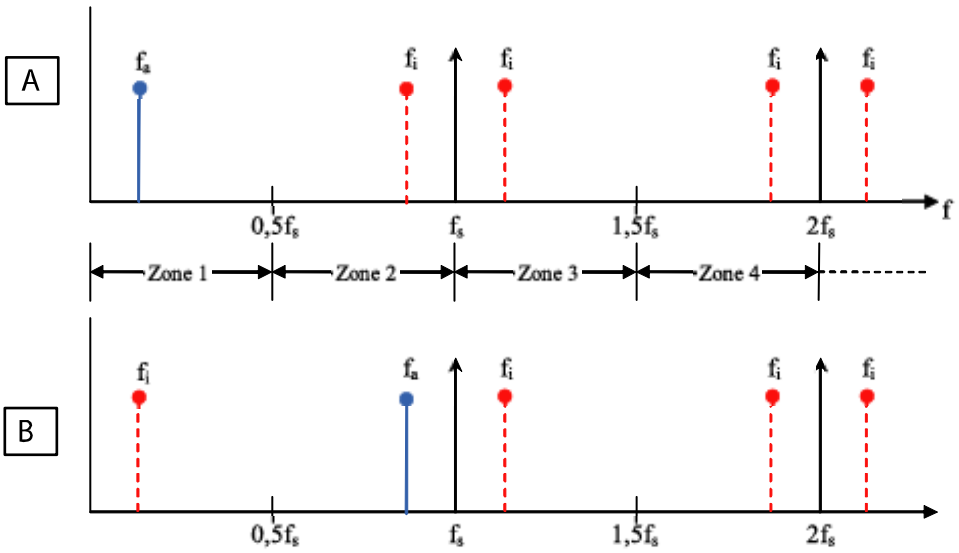


Frequency domain:
The corresponding frequency domain representation of this scenario is shown in Figure…Consider the case of a single frequency sinewave of frequency f_a sampled at frequency f_s by an ideal impulse sampler (see

Fig. 8-4) Also assume that $\mathbf{f_s > 2f_a}$ as shown. The frequency domain output of the sampler shows *aliases* or *images* of the original signal around every multiple of fs, i.e at

frequencies equal to $| \pm Kf_s \pm f_a |$, K = 1, 2, 3, 4 etc.

Fig. 8-4: Sampling of analog signals



f_i = Alias (Image)
f_a = Analog signal frequency
f_s = Sampling frequency

Sampled data systems:
The Nyquist bandwidth is defined to be the frequency spectrum from DC to $f_s/2$.The frequency spectrum is divided into an infinite number of Nyquist zones, each having a width equal to $0.5f_s$ (see

Fig. 8-4). In practice, the ideal sampler is replaced by an ADC followed by an FFT processor. The FFRT processor only provides an output form DC to $f_s/2$, i.e. the signals or aliases which appear in the first Nyquist zone.

Assumption:
The signal is outside the first Nyquist zone (see

Fig. 8-4/B). In this case the signal frequency is only slightly less than the sampling frequency, corresponding to the condition shown in the time domain representation (see Fig. 8-3). Please consider that even, although the signal is outside the first Nyquist zone, its image (or alias) (f_s-f_a) is inside. Returning to

Fig. 8-4/A, it is clear that if an unwanted signal appears at any of the image frequencies of f_a, it will also occur at f_a thereby producing an unwanted component in the first Nyquist zone.

This is similar to the analog mixing process and implies that filtering is required ahead of the sampler (or ADC) to remove frequency components which are outside the Nyqusit bandwidth, but whose aliases components fall inside it. The filter performance will depend on how close the out-of-band signal is to $f_s/2$ and the amount of attenuation required.

c) Anti-aliasing filter

Low-pass filtering to eliminate all components above the Nyquist frequency either before or during the digitization process can guarantee that the digitized data set is free of all aliases. The **APCI-3600** employs both digital and analog low-pass filters to achieve this end.

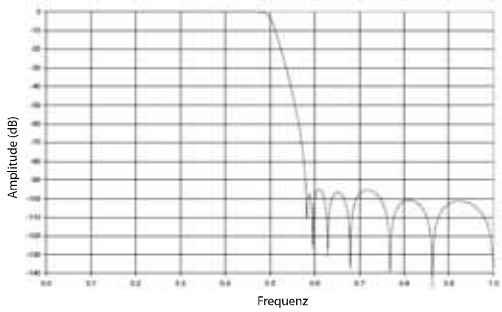
The delta-sigma ADCs on the **APCI-3600** include brick-wall digital filters whose cut-off frequency tracks the sampling rate. Thus, the filter topology automatically adjusts to follow the Nyquist frequency. The table below shows the data for cut-off frequency, passband, stopband and stopband attenuation. The digital filter has a negligible effect on frequency components that lie in the band of interest. Because the filter employs a FIR (Finite Impulse Response) architecture, its phase response is linear.

Table 8-2: Anti-aliasing filter: Input levels

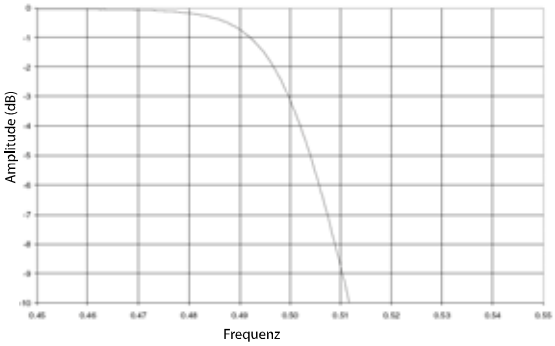
F_s (Sampling frequency)	2 kHz <=f_s<=50 kHz	50 kHz < f_s <=100 kHz	100 kHz < f_s <= 200 kHz
Cut-off frequency (-3 dB)	0.5 xf_s	0.5 xf_s	0.358 xf_s
Passband (-0,1 dB)	DC to 0.47 xf_s	DC to 0.45 xf_s	DC to 0.24 xf_s
Stopband	0.58 xf_s	0.68 xf_s	0.78 xf_s
Stopband attenuation	<95 dB	<-92 dB	<-97 dB

Fig. 8-5: Single Speed Mode (Stoband attenuation)

Fig. 8-6: Single Speed Mode (Transition-Band)



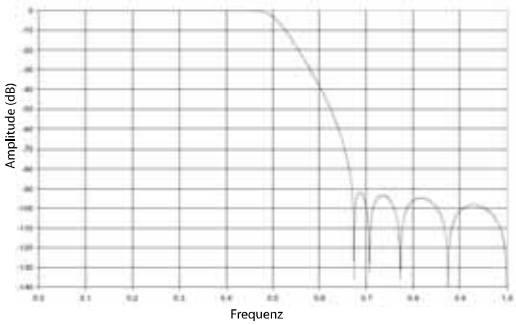
Single Speed Mode Stopband-Dämpfung



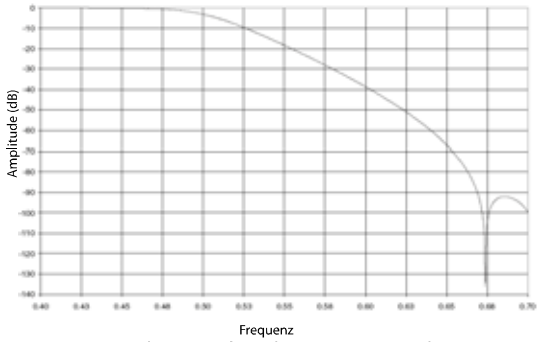
Single Speed Mode Transiton-Band

Fig. 8-7: Double Speed Mode (Stopband)

Fig. 8-8: Double Speed Mode (Transition-Band)



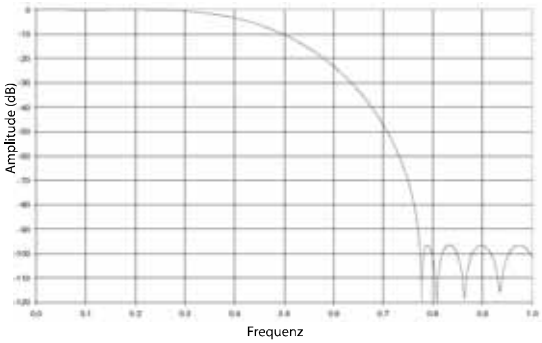
Double Speed Mode Stopband-Dämpfung



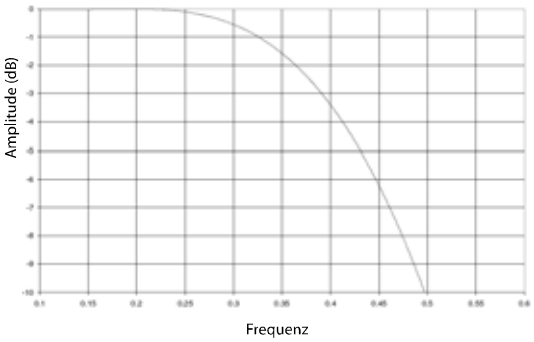
Double Speed Mode Transition-Band

Fig. 8-9: Quad Speed Mode (Stopband- attenuation)

Fig. 8-10: Quad Speed Mode (Transition-Band)



Quad Speed Mode Stopband-Dämpfung



Quad Speed Mode Transition-Band

d) Analog filter

Although the digital filter eliminates almost all out-of-band components, it is still susceptible to aliases from certain narrow frequency bands, especially those bands that lie within plus or minus one Nyquist bandwidth of 64 xf_s.

Example:
If f_s = 10.000 S/s, the digital filter could admit aliases from analog components between 635 kHz an 645 kHz.

In addition, the digital filtering built into ADCs, the **APCI-3600** devices also feature a fixed-frequency analog filter. The analog filter serves to remove high-frequency components in the analog signal path before they reach the ADC. This addresses the possibility of any high-frequency aliasing from the narrow bands not covered by the digital filter. Each input channel on the **APCI-3600** is equipped with a two-pole low-pass filter. The analog filter cut-off frequency is about 250 kHz. This filter has a gradual roll-off with very little attenuation for any frequencies within the 45 kHz input bandwidth of the **APCI-3600**. The high cut-off frequency of the filter guarantees extremely flat amplitude response and minimal phase error for signals of interest. Frequency components that could pass the digital filter usually consist of high frequency noise. The analog low-pass filter removes these high-frequency components before reaching the ADC.

While the frequency response of the digital filters scales directly with the sampling rate, the response of the analog filter is fixed. The analog filter response is optimized to produce good high-frequency alias rejection while maintaining a flat in-band frequency response. Because the analog filter is a two-pole system, its rolloff is not extremely sharp. It has excellent alias rejection at higher sampling rates, where only very high frequencies could pass through the digital filter. At lower sampling rates it does not filter potential aliases completely, but in most cases these residual aliases are noise rather than well-defined harmonics.

8.3.7 ADC (A/D converter)

The ADC (A/D converter) uses a multi bit sigma delta modulator 5th order, after which a digital filter and a decimator (anti-aliasing filter) are topped.

Resolution:
The sigma delta ADC has a resolution of 24-bit.

Sampling frequency:
The sampling frequency can be programmed between 2 kHz and 200 kHz. e sigma delta ADC consists of three sampling frequency ranges (see Table 8-3: Sampling frequency ranges.)

Table 8-3: Sampling frequency ranges

Mode	Frequency range
Single Speed	2 kHz - 50 kHz

Double Speed	50 kHz – 100 kHz
Quad Speed	100 kHz – 200 kHz

In all three speed modes the ADC uses an oversampling frequency of x 64.

The sampling frequency depends from the speed mode and the clock divisor factor. The divisor is selectable for all chronometers.

Table 8-4: Sampling frequency of the analog inputs

		Sampling frequency LRCK (f_s)		
PLD clock	**Divisor factor**	**Single speed mode (in Hz)**	**Double speed mode (in Hz)**	**Quad speed mode (in Hz)**
102400000	4	50000	100000	200000
102400000	5	40000	80000	160000
102400000	6	33333	66667	133333
102400000	8	25000	50000	100000
102400000	10	20000		
102400000	12	16667		
102400000	16	12500		
102400000	20	10000		
102400000	25	8000		
102400000	40	5000		
102400000	50	4000		
102400000	60	3333		
102400000	80	2500		
102400000	100	2000		

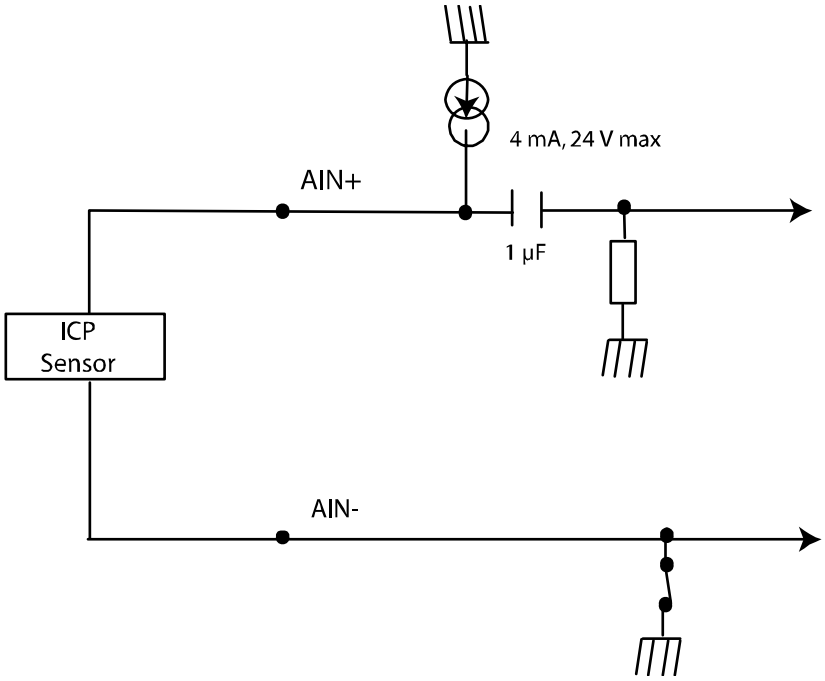
8.3.8 Current sources

On the **APCI-3600** are four current sources available for the supply of the iCP sensors. The current sources are on the four first positive analog inputs:

AIN0 R+
AIN0 L+
AIN1 R+
AIN1 L+

The sources of current stay switched off automatically in the DC mode and can be switched on only in the AC mode. If one current source is activated, the respective positive input switches into AC mode and the negative input switches to ground, so that the circuit can be built over the iCP sensor.

Fig. 8-11: iCP sensor supply with the APCI-3600



8.3.9 Modes

For the analog inputs of the **APCI-3600** you have the choice between the following modes:
a) Simple mode
b) Cyclic mode: Auto Buffer
c) Cyclic mode: Ring Buffer

a) Simple mode

In this mode two values can be read from the selected analog input (left and right channel). The interrupt is not used here.
Used functions:
i_PCI3600_StartAnalogInputModuleSingleAcquisition
i_PCI3600_GetAnalogInputModuleSingleAcquisitionStatus
i_PCI3600_ReadAnalogInputSingleAcquisitionValues
oder
i_PCI3600_GetAnalogInputModuleValues

b) Cyclic mode: Auto stop

In the auto Stop mode (single acquisition) the acquisition is stopped when the value, which you have previously defined for the on board SDRAM is reached.

You can define a compare value through the software:
The compare value defines the number of values for the SDRAM. When these values are reached, a compare interrupt will be called and the driver of the board initializes the DMA transfer. When the transfer is completed, the user interrupt routine will be called. When the on board SDRAM is full, an auto stop interrupt occurs.

The following errors can occur in this mode:
- FIFO error
- PC buffer overflow (when the PC buffer is smaller than the on board SDRAM buffer and the application of the user read the value not fast enough)

c) Cyclic mode: Ring buffer

In the ring buffer mode (see chapter 8.10.2) (continuous acquisition) the acquisition is stopped only through a board error (for example FIFO error or SDRAM overflow) or through the software.

You can define a compare value through the software:
The compare defines the number of values for the SDRAM. When these values are reached, a compare interrupt will be called and the driver of the board initializes the DMA transfer of the value from the SDRAM to the PC buffer. When the transfer is completed, the user interrupt routine will be called.

The following errors can occur in this mode:
- FIFO error
- SDRAM overflow (When the driver transfers the data not fast enough into the PC buffer).
- PC buffer overflow (When the PC buffer is smaller than the on board SDRAM and the application of the user read the value not fast enough)

8.4 Analog outputs

The board **APCI-3600** has two outputs, which are independent from each other.

Resolution:
Each DAC has a resolution of 16-bit with a precision of 13-bit.

Output range:
The output voltage ranges at ±10 V. Each output can supply up to ±10 mA.

Calibration:
The offset and gain error is calibrated at each analog output over a calibration. The calibration is realized by ADDI-DATA.

Power-on:
After the power-on phase the analog outputs will be set on 0 V.

Sampling frequency:
The sampling frequency is programmable between 2 kHz and 200 kHz. The DAC has the following 3 frequency ranges:

Table 8-5: Sampling frequency

Mode	Frequency range
Single Speed	From 2 kHz to 50 kHz
Double Speed	From 50 kHz to 100 kHz
Quad Speed	From 100 kHz to 200 kHz

The sampling frequency depends from the speed mode and the clock divisor factor (see table

Table 8-5).

8.4.1 Modes of the analog outputs

For the analog outputs there are three different modes available:

a) Simple mode
b) Signal generator mode (free run)
c) Signal generator mode (ring buffer)

a) Simple mode

In this mode either the functions
i_PCI3600_GetAnalogOutputReadyBitStatus
i_PCI3600_WriteAnalogOutputValue

or the function

i_PCI3600_Set1AnalogOutputChannel is used

b) Signal generator mode (free run)

In the free run, the defined SDRAM is filled with values from the PC buffer. The value of the SDRAM will be written automatically to the analog output. In this case no errors can occur.

c) Signal generator mode (ring buffer)

At the beginning (through initialization), the SDRAM will be filled with values from the PC buffer.

The compare value defines the size of free place in the SDRAM. When this value is reached, a compare interrupt occurs and the board driver initialises a DMA transfer of the values from the PC buffer to the SDRAM. When the transfer is completed, the user interrupt routine is called.

In this mode the following errors can occur:

- FIFO error
- SDRAM underflow: When the driver does not transfer the new values not fast enough into the SDRAM
- PC buffer underflow: When the user application does not write new values fast enough into the PC buffer.

Table 8-6: Sampling frequency of the analog outputs

			Single Speed	Double Speed	Quad Speed
Master	**Divisor**	**Clock**	**Sampling**	**Sampling**	**Sampling**

clock	factor	generator	frequency (LDAC) (in Hz)	frequency (LDAC) (in Hz)	frequency (LDAC) (in Hz)
102400000	4	25600000	50000	100000	200000
102400000	5	20480000	40000	80000	160000
102400000	6	17066667	33333	66667	133333
102400000	8	12800000	25000	50000	100000
102400000	10	10240000	20000		
102400000	12	8533333	16667		
102400000	16	6400000	12500		
102400000	20	5120000	10000		
102400000	25	4096000	8000		
102400000	40	2560000	5000		
102400000	50	2048000	4000		
102400000	60	1706667	3333		
102400000	80	1280000	2500		
102400000	100	1024000	2000		

8.5 Chronometer inputs

The board **APCI-3600** has 4 chronometer inputs (32-bit) with RS485 interface.

Max. speed:
1 MHz

Resolution:
Each chronometer has a 32-bit depth: On the rising flank of the input signal, the time will be latched into a 32-bit register.

Clear:
Each chronometer can be deleted over „Clear“. Then it counts again from 0 on. When reaching the final value (232 = 4 GB).

Gate function:
Two Gate-inputs with RS485 interface are available. Herewith you can block or release the chronometer-inputs.

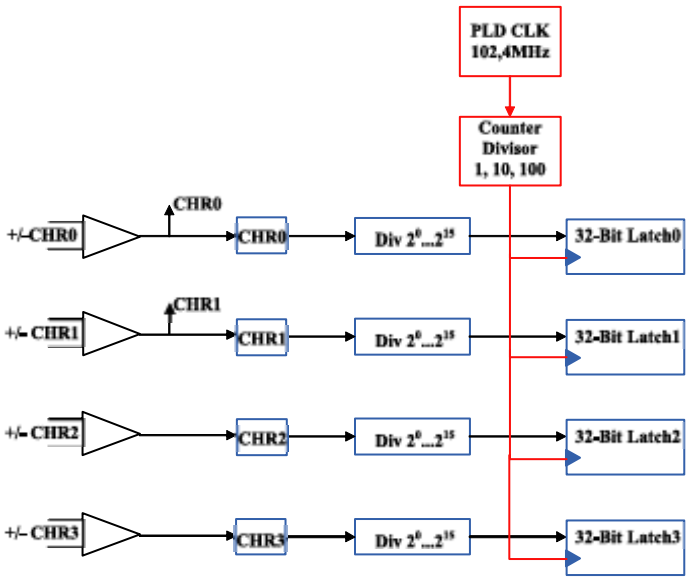
Gate 0 switches chronometer 0
Gate 1 switches chronometer 1

A low-level at Gate 0 or 1 blocks the chronometer input 0 or 1.
A high-level at Gate 0 or 1 releases the chronometer input 0 or 1.

Clock divisor:
Each chronometer input is clocked with 102.4 MHz. This clock also can be divided through 10 or 100 over the software. This is set for all 4 chronometers in common

Chronometer divisor:
Each chronometer input has its own divisor. It can be set from 2^0 to 2^{15} (in 2 power steps) over the software.
The two first chronometer inputs can be laid also on to the bus clock (see chapter 8.8)

Fig. 8-12: Chronometer inputs



8.5.1 Modes of the chronometers

For the chronometers the following modes are available:
a) Simple mode
b) Cyclic mode: Auto stop
c) Cyclic mode: Ring buffer

a) Simple mode

In this mode the values are saved in the internal FIFO (256 values) of the board. If the software does not read these values, a FIFO overflow occurs and the acquisition will be stopped. This mode neither uses the RAM on the board nor the interrupt.

The following software functions are used by this mode:
i_PCI3600_StartChronometerModuleAcquisition()
i_PCI3600_GetChronometerModuleFIFOStatus()

i_PCI3600_ReadChronometerModuleValue()
i_PCI3600_StopChronometerModuleAcquisition()

b) Cyclic mode: Auto stop

In the auto stop mode (single acquisition) the acquisition will be stopped, when the amount of values that you previously defined for the on board SDRAM is reached.

You can define a compare value through the software:
The compare value defines a number of values for the SDRAM. When these values are reached, compare interrupt will be called and the driver of the board initialises the DMA transfer. When the transfer is completed, the user interrupt routine will be called. When the on board SDRAM is full, an auto stop interrupt occurs.

The following errors can occur in this mode:

- FIFO error
- PC buffer overflow (When the PC buffer is smaller than the on board SDRAM buffer and the application of he user did not read the value fast enough)

c) Cyclic mode: Ring buffer

In the ring buffer mode (continuous acquisition), the acquisition will be stopped only through a board error (for example FIFO error or SDRAM overflow) or through the software.

You can define a compare value through the software:
The compare value defines the number of values for the SDRAM. When these values are reached, a compare interrupt will be called and the driver of the board initializes the DMA transfer of the value from the SDRAM to the PC buffer. When the transfer is completed the user interrupt routine will be called.

The following errors can occur in this mode:

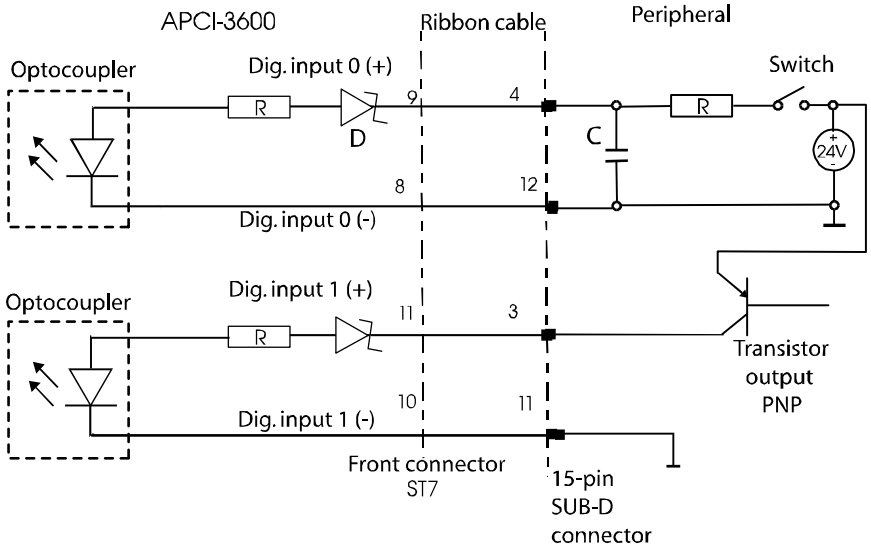
- FIFO error
- SDRAM overflow (when the driver transfers the data not fast enough into the PC buffer).
- PC buffer overflow (When the PC buffer is smaller than the on board SDRAM buffer and the application of the user does not read the value fast enough)

8.6 Digital inputs

The board **APCI-3600** has 8 digital inputs

You can use the first digital input as external trigger in order to start all resources of the board.

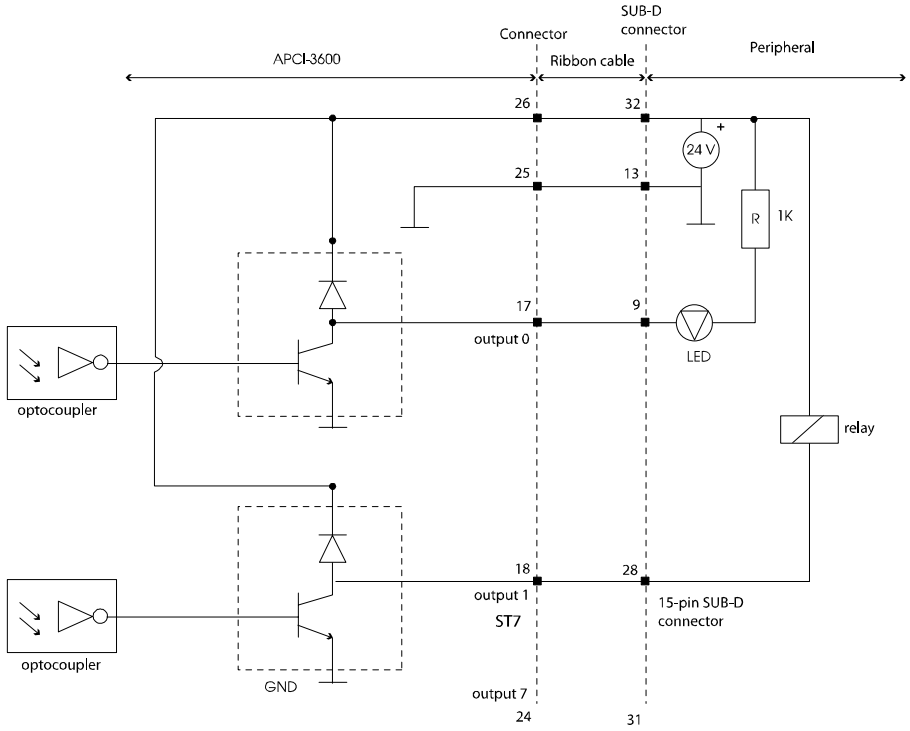
Fig. 8-13: Digital inputs



8.7 Digital outputs

The board **APCI-3600** has 8 digital outputs.

Fig. 8-14: Digital outputs



8.8 External clocks and master trigger

The board **APCI-3600** has 2 external clocks. You can operate the board in the following 2 modes:
- Master mode (see chapter 8.8.1)
- Slave mode (see chapter 8.8.2)

Each ADC or DAC can select a bus clock from 2 bus clocks (bus clocks ½). Each bus clock signal can be from 4 different and from each other independent sources:

a) From clock generator 1
Comes from the 102.4 MHz clock and is clocked down over the divisor generator 1 (see table below).

Table 8-7: Clock generator values

PLD_CLK (Hz)	Divisor clock Gen 1/2	Clock generator 1/2 (Hz)
102400000	4	25600000
102400000	5	20480000
102400000	6	17066667
102400000	8	12800000
102400000	10	10240000
102400000	12	8533333
102400000	16	6400000
102400000	20	5120000
102400000	25	4096000
102400000	40	2560000
102400000	50	2048000
102400000	60	1706667
102400000	80	1280000
102400000	100	1024000

b) From clock generator 2
Comes from the 102.4 MHz clock and is clocked down over the divisor generator 2.

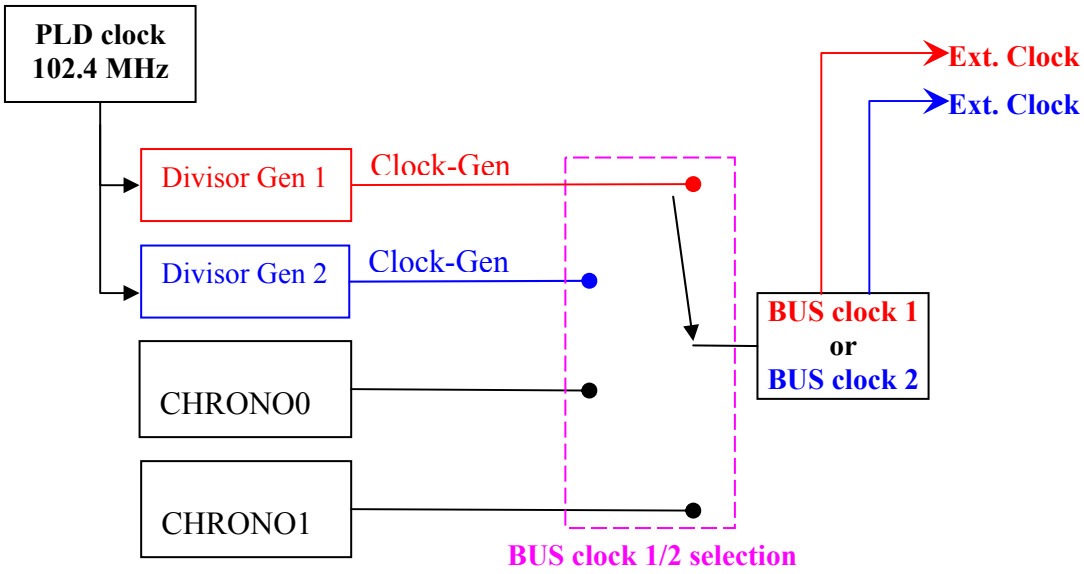
c) From the chronometer input 0

d) From the chronometer input 1

8.8.1 Master mode

In the master mode the **APCI-3600** lays the bus clock ½ on to the external clocks ½.
The external clocks ½ will be configured as output. When you configure the board as output, all functionalities of the board will be configured in the same clock **102.4 MHz**.

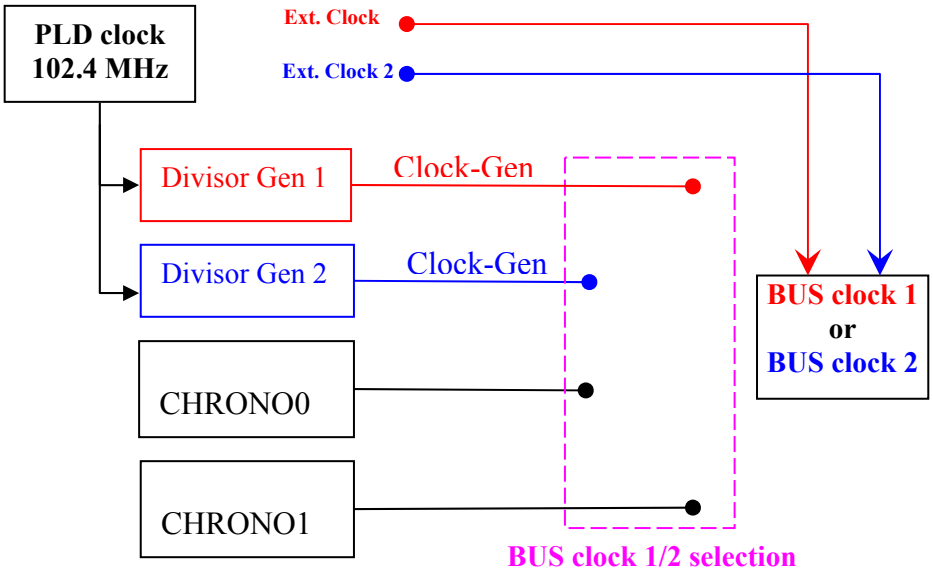
Fig. 8-15: Board in the master mode



8.8.2 Slave mode

In the slave mode the bus clocks ½ are connected directly to the external clocks ½. The external clocks ½ will be configured as input. In this mode different APCI-3600 can be synchronized. Hereto on board is set in the master mode and all others in the slave mode.
If you configure the board as slave, all analog input and output channels will be configured with the same clock of the master board. The chronometer uses always the clock on the slave board: **102.4 MHz**.

Fig. 8-16: Board in the slave mode



8.9 On board buffer (SDRAM)

The **APCI-3600** supports SDRAM buffer in order to transfer the high data flow from the board to the computer.
The on board buffer is a 144 pin SO-DIMM SDRAM module (notebook memory). The **APCI-3600** is supplied with a 128 Mbytes SDRAM module in the standard version. The **APCI-3600** supports 144 pin SO-DIMM SDRAM module with different buffer sizes:

- 64 MBytes
- 128 MBytes (standard)
- 256 MBytes
- 512 Mbytes

8.10 Buffer concept

The following chapter presents the basic buffer concepts of the **APCI-3600** board:

- Global buffer
- Ring buffer

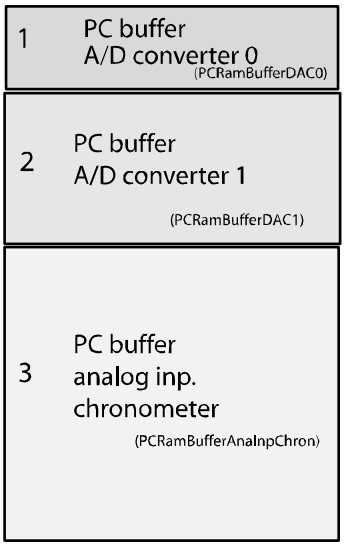
8.10.1 Buffer concept: Global buffer

After the installation of your board, you must register it with the ADDIREG program (see chapter 6.1)

If you want to use the cyclic acquisition or the signal generator of the board, you firstly must reserve storage location (RAM) on your computer.
In the ADDIREG program you can define the size of the global buffer that is allocated to the RAM in the computer. This buffer is called "PCRamGlobalBuffer"
The size of the buffer is defined according to the following criteria:

- The buffer size that you want to receive bye each interrupt for each ADC / Chronometer.
- The buffer size that that shall be reserved for each ADC.

Fig. 8-17: Global buffer



The PC buffer „analog inputs/chronometer" (3) PCRamBufferAnaInpChrono Buffer) is used from the analog inputs and the chronometer together.

8.10.2 Buffer concept: Ring buffer

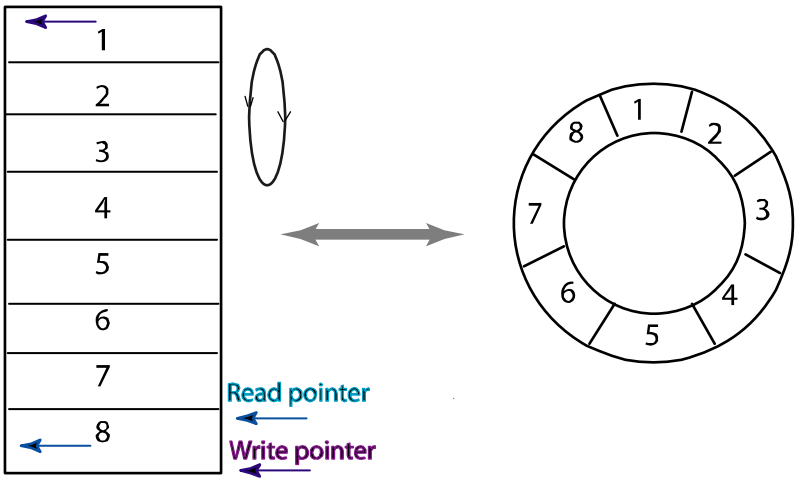
In the following the ring buffer concept will be explained firstly in general and then on the **APCI-3600**.

Overview: Ring Buffer

A ring buffer is a part of a memory with FIFO access (FIFO: First In First Out).

A ring buffer consists of the following components:
- buffer part with a defined size
- A read pointer
- A write pointer
- Flags: Empty and filled

Fig. 8-18: Ring Buffer



Management of the flags:
The read process sets the empty flag, in the case, if after the read, the read pointer is equal to the write pointer. The read process resets the full flag.
The write process sets the full flag if after the write, the write pointer is equal to the read pointer. The write process resets the empty flaag.

Use of the ring buffer:
The ring buffer is used when two processes want to exchange data and when these two processes run asynchronously.

b) Ring buffer on the APCI-3600

When the ring buffer mode is used for the analog inputs and outputs and chronometers, this will be managed as follows:
The on board SDRAM and the PC buffer will be used as a ring buffer.

On Board SDRAM ring buffer:
- The firmware manages the flags of the on board On Board SDRAM ring buffer
- The firmware generates an interrupt by overflow/underflow of the ring buffer and stops the functionality that uses this buffer.

These two processes are:
- The firmware of the board
- The driver of the board (interrupt function)

PC buffer ring buffer:
The driver of the board (interrupt function) starts a DMA transfer to write new values into the PC. The user application reads the value from the PC buffer. If the PC buffer overflows, the driver of the board stops the resource and informs the user application by calling the user interrupt routine.
The driver of the board manages the flags of the PC buffer.
These two processes are:
- The driver of the board (interrupt function)
- The user application

8.11 Max. data transfer speed

The data transfer speed of the **APCI-3600** depends on different parameters, for example:
- Sampling frequency
- Number of channels
- Acqusition mode

In the worst case all channels will operate in the ring buffer mode with a max sampling frequency of 200 kHz, this means
- 8 analog inputs with always 32-bit per channel and 200 kHz sampling frequency
- 2 analog outputs with always 32-bit per channel and 200 kHz sampling frequency
- 4 chronometer inputs with always 32-bit per channel and 1 MHz input frequency

Table 8-8: Data transfer speed

Function	Channel	Depth	Max. sampling frequency	Data transfer
Analog input	8	32-bit	200 kHz	6.4 MB/s
Analog output	2	32-bit	200 kHz	1.6 MB/s
Chronometer	4	32-bit	1 MHz	16 MB/s

In the worst case **24 MBytes/s** will be transferred. In order to achieve this rate, your computer requires the following:
- Fast processor
- Fast hard disk
- Sufficient SDRAM buffer in order ot allocate enough PC buffer for the **APCPI-3600**.

9 SOFTWARE

9.1 Software functions

9.1.1 General functions

1) i_PCI3600_InitCompiler()

INT i_PCI3600_InitCompiler(BYTE b_CompilerDefine)

Parameters:

- inputs:

BYTE	b_CompilerDefine	The user has to choose the language under Windows in which he/she wants to program - DLL_COMPILER_C: The user programs in C. - DLL_COMPILER_VB_4: The user programs in Visual Basic for Windows. - DLL_COMPILER_VB_5_VB_6: The user programs in Visual Basic 5/6 for Windows NT/2000/XP or Windows 98. - DLL_COMPILER_PASCAL: The user programs in Pascal or Delphi. - DLL_LABVIEW : The user programs in Labview. - DLL_COMPILER_DOT_NET : The user programs in .NET

- outputs:

Task:
Initialise the compiler used.

Calling convention:
INT i_ReturnValue;

i_ReturnValue = i_PCI3600_InitCompiler (DLL_COMPILER_C);

Return Value:
0 : No Error
-1: Wrong compiler parameter

2) i_PCI3600_GetBoardList ()

Syntax:

```
_INT_ i_PCI3600_GetBoardList (PBYTE pb_NbrOfBoard,
                              PBYTE pb_PCISlotNbr)
```

Parameters:
- inputs:
No input
- output

PBYTE	pb_NbrOfBoard	Return the number of board found
PBYTE	pb_PCISlotNbr	Return the PCI Slot number of each PCI board found

Task:
Return the number of **APCI-3600** board found (*pb_NbrOfBoard*) and the PCI Slot number of each board (*pb_PCISlotNbr*).

Calling convention:

```
BYTE b_NbrOfBoard;
BYTE b_PCISlotArray [20];
INT     i_ReturnValue;

i_ReturnValue = i_PCI3600_GetBoardList (&b_NbrOfBoard, b_PCISlotArray);
```

Return Value:
0 : No Error

3) i_PCI3600_OpenBoard ()

Syntax:

```
_INT_ i_PCI3600_OpenBoard (BYTE b_BoardIndex,
                           PDWORD pdw_BoardHandle)
```

Parameters:
- inputs:

BYTE	b_BoardIndex	Index of the board to open

- output

PDWORD	pdw_BoardHandle	Handle of the board **APCI-3600** for using the functions

Task:
Open the board with the index : *b_BoardIndex*. A handle is returned to the user which allows using the following functions.
Handles allows operation several boards.

Calling convention:

```
DWORD dw_BoardHandle;
INT       i_ReturnValue;

i_ReturnValue = i_PCI3600_ OpenBoard (0,
                                      &dw_BoardHandle);
```

Return Value:
0 : No Error
-1 : Not available board index
-2 : Board not present
-3 : No handle is available for the board
-4 : Can not open the Windows driver
-5 : Map the physical memory error
-6 : This version of the board in not supported by this driver. Please do an update of it
-7 : Error by allocate memory for the descriptor list used by DMA transfer.
-8 : Error to create the end-debug call back function
-9 : Error to install the end-debug call back function
-10 : Cannot create driver Mutex
-11 : Driver Shared Memory Allocation Error
-12 : Functionality Mutex creation Error
-13 : General Open Error

4) i_PCI3600_InitBoard ()

Syntax:

```
_INT_ i_PCI3600_InitBoard (DWORD dw_BoardHandle,
                           BYTE b_MasterSlaveMode,
                           BYTE b_ClockGen1_Divisor,
                           BYTE b_ClockGen2_Divisor,
                           BYTE b_SamplingClock1_Selection,
                           BYTE b_SamplingClock2_Selection,
                           BYTE_ b_SlaveSamplingClock1_Selection,
                           BYTE_ b_SlaveSamplingClock2_Selection,
                           BYTE b_Chronometer_MasterClockDivisor)
```

Parameters:
- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_MasterSlaveMode	Selection of the mode of the board: 0 : Master Mode 1 : Slave Mode In slave mode, the Clock1&2 selection and divisor must be the same that the master board.
BYTE	b_ClockGen1_Divisor	Selection of the divisor for the clock generator 1 See Table
BYTE	b_ClockGen2_Divisor	Selection of the divisor for the clock generator 2 See Table
BYTE	b_SamplingClock1_Selection	Selection of the clock for the sampling clock 1 See Table
BYTE	b_SamplingClock2_Selection	Selection of the clock for the sampling clock 2 See Table
BYTE	b_SlaveSamplingClock1_Selection	Selection of the clock for the sampling clock 1 for the slave board 0 : PCI3600_EXT_CLOCK_1 1 : PCI3600_EXT_CLOCK_2
BYTE	b_SlaveSamplingClock2_Selection	Selection of the clock for the sampling clock 2 for the slave board 0 : PCI3600_EXT_CLOCK_1 1 : PCI3600_EXT_CLOCK_2
BYTE	b_Chronometer_MasterClockDivisor	

Selection of the divisor for the master clock for the Chronometer:
0 : PCI3600_CHRONOMETER_DIVISOR_1
1 : PCI3600_CHRONOMETER_DIVISOR_10
2 : PCI3600_CHRONOMETER_DIVISOR_100

- output
No output

Task:
Initialises the board: Does the initialisation of the two master clock used by the analog input and the analog output and the divisor to use for the clock of the chronometer.
And define if the board work as Master or as Slave.

Table 9-1: Clock divisor

Definition	Meaning (clock generator value in MHz)
PCI3600_DIVISOR_4	25.6
PCI3600_DIVISOR_5	20.48
PCI3600_DIVISOR_6	17.06
PCI3600_DIVISOR_8	12.8
PCI3600_DIVISOR_10	10.24
PCI3600_DIVISOR_12	8.53
PCI3600_DIVISOR_16	6.4
PCI3600_DIVISOR_20	5.12
PCI3600_DIVISOR_25	4.096
PCI3600_DIVISOR_40	2.56
PCI3600_DIVISOR_50	2.048
PCI3600_DIVISOR_60	1.706
PCI3600_DIVISOR_80	1.28
PCI3600_DIVISOR_100	1.024

Table 9-2: Clock selection

Definition	Meaning
PCI3600_CLK_GENERATOR_1	Clock generator 1 = 25.6MHz / Clock Gen 1 Divisor
PCI3600_CLK_GENERATOR_2	Clock generator 2 = 25.6MHz / Clock Gen 2 Divisor
PCI3600_CHRONOMETER_0_LATCH_INPUT	Chronometer 0 latch input
PCI3600_CHRONOMETER_1_LATCH_INPUT	Chronometer 1 latch input

Calling convention:

```
DWORD dw_BoardHandle;
INT     i_ReturnValue;

i_ReturnValue = i_PCI3600_InitBoard (dw_BoardHandle,
                                      0,
                                      PCI3600_DIVISOR_1,
                                      PCI3600_DIVISOR_10,
                                      PCI3600_CLK_GENERATOR_1,
                                      PCI3600_CLK_GENERATOR_2,
                                      0,
                                      0,

PCI3600_CHRONOMETER_DIVISOR_1);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Master slave mode parameter is wrong
-3 : One or both of the selected bus clock divisor parameter is wrong
-4 : One or both of the clock selection is wrong
-5 : Slave sampling clock selection is wrong
-6 : Chronometer master clock divisor parameter is wrong
-7 : The board is already initialised by another process
-8 : The board is in use : Release all channels to reinitialise it

5) i_PCI3600_GetBoardInitialisation ()

Syntax:

```
_INT_ i_PCI3600_GetBoardInitialisation (DWORD dw_BoardHandle,
                                        PBYTE pb_MasterSlaveMode,
                                        PBYTE pb_ClockGen1_Divisor,
                                        PBYTE pb_ClockGen2_Divisor,
                                        PBYTE
pb_SamplingClock1_Selection,
                                        PBYTE
pb_SamplingClock2_Selection,
                                        PBYTE_
                    pb_SlaveSamplingClock1_Selection,
                                        PBYTE_
        pb_SlaveSamplingClock2_Selection,
                                        PBYTE_
pb_Chronometer_MasterClockDivisor)
```

Parameters:
- input:

DWORD	dw_BoardHandle	Handle of the board

- output

PBYTE	pb_MasterSlaveMode	Get the mode of the board: 0 : Master Mode 1 : Slave Mode
PBYTE	pb_ClockGen1_Divisor	Get the divisor for the clock generator 1 See Table
PBYTE	pb_ClockGen2_Divisor	Get the divisor for the clock generator 2 See Table
PBYTE	pb_SamplingClock1_Selection	Get the clock for the sampling clock 1 See Table
PBYTE	pb_SamplingClock2_Selection	Get the clock for the sampling clock 2 See Table
PBYTE	pb_SlaveSamplingClock1_Selection	Get the clock for the sampling clock 1 for the slave board 0 : PCI3600_EXT_CLOCK_1 1 : PCI3600_EXT_CLOCK_2
PBYTE	pb_SlaveSamplingClock2_Selection	Get the clock for the sampling clock 2 for the slave board

0 : PCI3600_EXT_CLOCK_1
1 : PCI3600_EXT_CLOCK_2

PBYTE pb_Chronometer_MasterClockDivisor
Get the divisor for the master clock
for the Chronometer:
0 : PCI3600_CHRONOMETER_DIVISOR_1
1 :
PCI3600_CHRONOMETER_DIVISOR_10
2 :
PCI3600_CHRONOMETER_DIVISOR_100

Task:
Gets the initialisation information of the board.

Calling convention:

```
DWORD dw_BoardHandle;
INT     i_ReturnValue;
BYTE b_MasterSlaveMode;
BYTE b_ClockGen1_Divisor;
BYTE b_ClockGen2_Divisor;
BYTE b_SamplingClock1_Selection;
BYTE b_SamplingClock2_Selection;
BYTE b_SlaveSamplingClock1_Selection;
BYTE b_SlaveSamplingClock2_Selection;
BYTE b_Chronometer_MasterClockDivisor;

i_ReturnValue = i_PCI3600_GetBoardInitialisation (dw_BoardHandle,
                                                  &b_MasterSlaveMode,
                                                  &b_ClockGen1_Divisor,
                                                  &b_ClockGen2_Divisor,

&b_SamplingClock1_Selection,

&b_SamplingClock2_Selection,

&b_SlaveSamplingClock1_Selection,

&b_SlaveSamplingClock2_Selection,

&b_Chronometer_MasterClockDivisor);
```

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong
-2 : The board is not initialised

6) i_PCI3600_GetBoardInformation ()

Syntax:

```
_INT_ i_PCI3600_GetBoardInformation(DWORD dw_BoardHandle,
                                    PDWORD
pdw_OnBoardRAMTotalSize,
                                    PBYTE
pb_NbrOfComputerBufferAllocated,
                                    PDWORD
pdw_ComputerBufferSizeArray )
```

Parameters:
- inputs:
DWORD dw_BoardHandle Handle of the board
- output
PDWORD pdw_OnBoardRAMTotalSize
Return the size of the RAM of the board (Size in MBytes).
PBYTE pb_NbrOfComputerBufferAllocated
Return the number of buffer allocated in the computer RAM
PDWORD pdw_ComputerBufferSizeArray
Return the size of the buffer allocated in the computer RAM.(Size in Byte).

Task:
Return the memory information of the board and the allocated memory from the computer for this board.

Calling convention:

```
DWORD dw_BoardHandle;
DWORD dw_OnBoardRAMSize;
DWORD pdw_ComputerBufferSize[100];
BYTE      b_NbrOfComputerBuffer;
INT       i_ReturnValue;

i_ReturnValue = i_PCI3600_GetBoardInformation (dw_BoardHandle,
                                               &dw_OnBoardRAMSize,
                                               &b_NbrOfComputerBuffer,
                                               pdw_ComputerBufferSize);
```

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong

7) i_PCI3600_CloseBoard ()

Syntax:
INT i_PCI3600_CloseBoard(DWORD dw_BoardHandle)

Parameters:
- inputs:
DWORD dw_BoardHandle Handle of the board
- output:
No output

Task:
Releases the handle of the board. Blocks the access to the board.

Calling convention:
DWORD dw_BoardHandle;
INT i_ReturnValue;

i_ReturnValue = i_PCI3600_CloseBoard (dw_BoardHandle);

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong
-2 : The function must be called in application level

8) i_PCI3600_SetBoardIntRoutine ()

Syntax:

```
_INT_ i_PCI3600_SetBoardIntRoutine
                        (BYTE b_BoardHandle,
                         BYTE b_UserCallingMode,
                         DWORD dw_UserSharedMemorySize,
                         VOID **  ppv_UserSharedMemoryAppLevel,
                         VOID **  ppv_UserSharedMemoryKernelLevel,
                         VOID (WINAPI *v_FunctionName)
                                    (DWORD_   dw_BoardHandle,
                                     BYTE_   b_UserCallingMode,
                                     VOID * pv_UserSharedMemory,
                                     DWORD_  dw_InterruptSource,
                                     DWORD_  dw_InterruptMask,
                                     DWORD_  dw_NbrOfComputerBuffer,
                                     PDWORD_
pdw_ComputerBufferAddressKernelLevel,
                                     PDWORD_
pdw_ComputerBufferAddressApplicLevel))
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_UserCallingMode	PCI3600_SYNCHRONOUS_MODE: The user routine is called directly by the driver interrupt routine. PCI3600_ASYNCHRONOUS_MODE: The user routine is called by the driver interrupt thread.
DWORD	dw_UserSharedMemorySize	Determines the size in bytes of the user shared memory. Only used if you have selected PCI3600_SYNCHRONOUS_MODE
VOID	v_FunctionName	Name of the user interrupt routine

- output

VOID **	ppv_UserSharedMemoryAppLevel	User shared memory address for the application level (Ring 3) Only used if you have selected PCI3600_SYNCHRONOUS_MODE
VOID **	ppv_UserSharedMemoryKernelLevel	User shared memory address for the kernel level (Ring 0)

Only used if you have selected
PCI3600_SYNCHRONOUS_MODE

Task:
If you use Visual Basic 5.0/6.0:
- only the asynchronous mode is available.

IMPORTANT!
Windows 32-bit information
For Windows NT/2000/XP and Windows 98, 4 rings (ring 0 to ring 3) are available.

- The user application operates in ring 3. This ring does not give access to hardware.
- VXD and SYS driver operate in ring 0 and give access to hardware.
- Ring 0 has no direct access to global variable from ring 3. It has to use a shared memory.
- Ring 0 and ring 3 have a pointer that points on this shared memory. The 2 pointers are not configured under the same address.

This function must be called up for each **APCI-3600** for which an interrupt is to be enabled. It installs one user interrupt function in all boards on which an interrupt is to be enabled.
First calling (first board):
- the user interrupt routine is installed
- interrupts are enabled
- user shared memory is allocated if PCI3600_SYNCHROUNOUS_MODE has been selected.

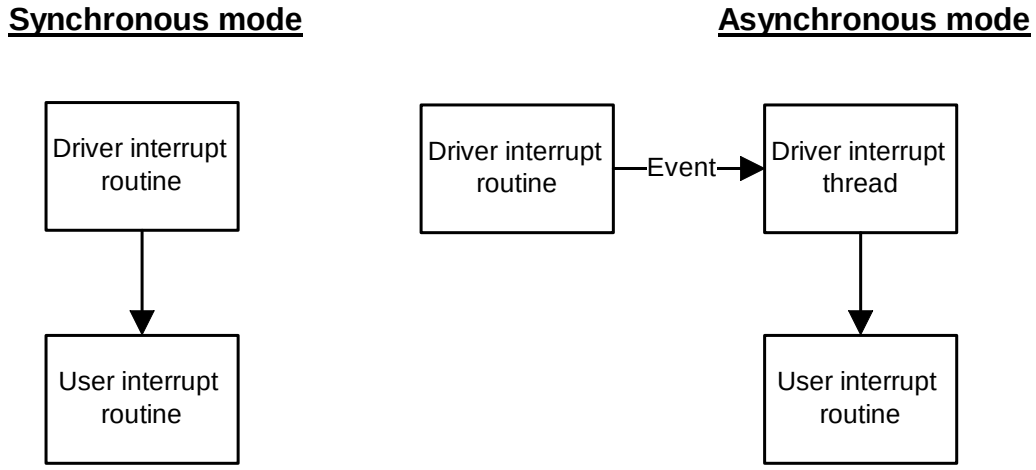
If you operate several **APCI-3600** boards which have to react to interrupts, call up the function as often as you operate the board boards **APCI-3600**. The variable *v_FunctionName* is only relevant **for the first calling**.
From the second call on of the function (next board):
- interrupts are enabled.

Interrupt
The user interrupt routine is called up by the system when an interrupt is generated.
If several boards are operated and if they have to react to interrupts, the variable *dw_BoardHandle* returns the identification number (handle) of the board which has generated the interrupt.

The user interrupt routine can be called:
- directly by the driver interrupt routine (synchronous mode). The code of the user interrupt routine directly operates in ring 0.
- by the driver interrupt thread (asynchronous mode). An event is generated and the interrupt thread calls up the user interrupt routine. The code of the user

interrupt routine operates in ring 3.
The driver interrupt thread has the highest priority (31) in the system.

Fig. 9-1: Synchronous and asynchronous mode



	SYNCHRONOUS MODE
ADVANTAGE	The code of the user interrupt routine is directly called by driver interrupt routine (ring 0). The time between interrupt and the user interrupt routine is reduced.
RESTRICTION	The user cannot debug the user interrupt routine.
	The user routine cannot call Windows API functions.
	The user routine cannot call functions which give access to global variables. The user can still use a shared memory.
	This mode is not available for Visual Basic

	ASYNCHRONOUS MODE
ADVANTAGE	The user can debug the user interrupt routine provided he has not programmed in Visual Basic 5
	The user routine can call Windows API functions.
	The user routine can call functions which give access to global variables.
	The user routine can call all **APCI-3600** driver functions with the following extension: "i_PCI3600_XXXX"
RESTRICTION	The code of the user interrupt routine is called by driver interrupt thread routine (ring 3). The time between interrupt and the user interrupt routine is increased.

Shared memory

If you have selected the PCI3600_SYNCHRONOUS_MODE you cannot have access to global variables. But you have the possibility to create a shared memory (*ppv_UserSharedMemory*). The user shared memory can have all predefined compiler types or user define types.

The variable *dw_UserSharedMemorySize* indicates the size in bytes of the selected user type. A pointer of the variable *ppv_UserSharedMemory* is given to the user interrupt routine with the variable *pv_UserSharedMemory.*This is not possible for Visual Basic.

The user interrupt routine must have the following syntax:

```
VOID    v_FunctionName    (DWORD      dw_BoardHandle,
                           BYTE       b_UserCallingMode,
                           VOID *     pv_UserSharedMemory,
                           DWORD      dw_InterruptSource,
                           DWORD      dw_InterruptMask,
                           DWORD      dw_NbrOfComputerBuffer,
                           PDWORD
                          pdw_ComputerRAMAddressKernelLevel,
                           PDWORD
        pdw_ComputerRAMAddressApplicLeve)
```

v_FunctionName	Name of the user interrupt routine
dw_BoardHandle	Handle of the **APCI-3600** which has generated the interrupt
b_UserCallingMode	PCI3600_SYNCHRONOUS_MODE: The user routine is directly called by driver interrupt routine. PCI3600_ASYNCHRONOUS_MODE: The user routine is called by driver interrupt thread
pv_UserSharedMemory	Pointer of the user shared memory.

dw_InterruptSource Source of the interrupt.
See Table 9-3
dw_InterruptMask Mask of the events which have generated the interrupt.
See Table 9-4
dw_NbrOfComputerBuffer Number of computer buffer whose contains the values
pdw_ComputerRAMAddressKernelLevel
Array of address to the memory whose contains the values for
the kernel level.(Ring 0)
pdw_ComputerRAMAddressAppLevel
Array of address to the memory whose contains the values for
the application level (Ring 3)

Table 9-3: Interrupt source

Value	Meaning
0000 0000 0000 0000 0000 0000 0000 0001	Analog Output 0
0000 0000 0000 0000 0000 0000 0000 0010	Analog Output 1
0000 0000 0000 0000 0000 0000 0000 0100	Analog Input Module 0
0000 0000 0000 0000 0000 0000 0000 1000	Analog Input Module 1
0000 0000 0000 0000 0000 0000 0001 0000	Analog Input Module 2
0000 0000 0000 0000 0000 0000 0010 0000	Analog Input Module 3
0000 0000 0000 0000 0000 0000 0100 0000	Chronometer 0
0000 0000 0000 0000 0000 0000 1000 0000	Chronometer 1
0000 0000 0000 0000 0000 0001 0000 0000	Chronometer 2
0000 0000 0000 0000 0000 0010 0000 0000	Chronometer 3

Table 9-4: Interrupt mask

Functionality	Value	Meaning
Analog Output	0000 0000 0000 0000 0000 0000 0000 0001	Compare interrupt has occur and a transfer has been do.
	0000 0000 0000 0000 0000 0000 0000 0010	Internal FIFO error interrupt
	0000 0000 0000 0000 0000 0000 0000 0100	SDRAM Underflow interrupt
	0000 0000 0000 0000 0000 0000 0000 1000	PC Buffer Underflow interrupt
Analog Input module / Chronometer	0000 0000 0000 0000 0000 0000 0000 0001	DMA interrupt
	0000 0000 0000 0000 0000 0000 0000 0010	Internal FIFO error interrupt
	0000 0000 0000 0000 0000 0000 0000 0100	SDRAM Overflow interrupt
	0000 0000 0000 0000 0000 0000 0000 1000	PC Buffer Overflow interrupt
	0000 0000 0000 0000 0000 0000 0001 0000	Auto-Stop interrupt

Table 9-5: Buffer

The buffer given in parameter in the interrupt routine has the following format:

Address	Write/Read																															
	D 31	D 30	D 29	D 28	D 27	D 26	D 25	D 24	D 23	D 22	D 21	D 20	D 19	D 18	D 17	D 16	D 15	D 14	D 13	D1 2	D 11	D 10	D 9	D 8	D 7	D6	D5	D 4	D 3	D 2	D1	D0
Address + 0	Number of values																												Index		Source	
Address + 4	Value 0																															
Address + 8	Value 1																															
Address + 12	…																															

Source: Source of the buffer:

0: ADC

1: Chronometer

Index: Index of the source

0: ADC/Chronometer 1

1: ADC/Chronometer 2

…

Number of values: Number of values transferred into the buffer.

```
BYTE      b_UserCallingMode,
DWORD     dw_UserSharedMemorySize,
VOID **   ppv_UserSharedMemoryApp,
VOID **   ppv_UserSharedMemoryKernel,
VOID      v_FunctionName(DWORD     dw_BoardHandle,
                         BYTE      b_UserCallingMode,
                         VOID *    pv_UserSharedMemory,
                         DWORD     dw_InterruptMask,
                         DWORD     dw_BufferHandle,
                         DWORD     dw_NbrOfComputerBuffer,
                         PDWORD
                         pdw_ComputerRAMAddressKernelLevel,
                         PDWORD
pdw_ComputerRAMAddressApplicLevel)
```

.

Calling convention:

```
typedef struct
        {
        .
        .
        .
        }str_UserStruct;

str_UserStruct * ps_UserSharedMemory;

void    v_FunctionName      (unsigned long   dw_BoardHandle,
                             unsigned char  b_UserCallingMode,
                             void *            pv_UserSharedMemory,
                             unsigned long dw_InterruptSource,
                             unsigned long dw_InterruptMask,
                             unsigned long dw_NbrOfComputerBuffer,
                             unsigned long
                            *pdw_ComputerRAMAddressKernelLevel,
                             unsigned long
*pdw_ComputerRAMAddressApplicLevel)
                {
                str_UserStruct * ps_InterruptSharedMemory;

                ps_InterruptSharedMemory = (str_UserStruct *)
pv_UserSharedMemory;
                .
                .
                }


DWORD dw_BoardHandle;
INT          i_ReturnValue;

i_ReturnValue = i_PCI3600_SetBoardIntRoutine (dw_BoardHandle,

PCI3600_SYNCHRONOUS_MODE,
                                                sizeof (str_UserStruct),
                                                (void **)
&ps_UserSharedMemoryApp,
                                                (void **)
&ps_UserSharedMemoryKernel,
                                                v_FunctionName);
```

Return Value:
0: No error
-1 : The handle parameter of the board is wrong
-2 : This function cannot be called in kernel level

-3 : Wrong calling mode
-4 : Not any memory free for the user global buffer
-5 : The interrupt management is not supported by this compiler
-6 : User interrupt routine calling mode selection wrong
-7 : Interrupt already installed
-8 : Interrupt not attributed
-9 : ACPI active but Windows NT 4 used
-10 : PNP OS active but Windows NT 4 used
-11 : Creates user shared memory error
-12 : Prepares user function to kernel execution error
-13 : Creates API interrupt event error
-14 : Creates API interrupt thread function error
-15 : Creates call back API function error
-16 : Installs API interrupt function error
-17 : Creates Interrupt Parameter shared memory error
-18 : Critical section preparation error

9) i_PCI3600_ResetBoardIntRoutine ()

Syntax:
INT i_PCI3600_ResetBoardIntRoutine(DWORD dw_BoardHandle)

Parameters:
- inputs:
DWORD dw_BoardHandle Handle of the board
- output
No output

Task:
Stops the interrupt management of the board **APCI-3600.**
Deinstalls the user interrupt routine if the management of interrupts of all boards **APCI-3600** is stopped.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;

i_ReturnValue = i_PCI3600_ResetBoardIntRoutine (dw_BoardHandle);
```

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong
-2 : This function cannot be call in kernel level
-3 : Interrupt Routine not installed

10) i_PCI3600_Write32BitPortValue()

Syntax:

```
_INT_ i_PCI3600_Write32BitPortValue(DWORD dw_BoardHandle,
                                    BYTE      b_AddressSelection,
                                    DWORD dw_Offset,
                                    DWORD dw_Value)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_AddressSelection	Selection of the address of the board to do the access.
DWORD	dw_Offset	Offset at which the value must be written.
DWORD	dw_Value	Value to write.

- Output

No output

Task:

Does a 32-bit output access (I/O or memory) on the board at offset *dw_Offset* from the address *b_AddressSelection*.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;

i_ReturnValue = i_PCI3600_Write32BitPortValue (dw_BoardHandle,
                                               0,
                                               0,
                                               0xFFF0000FUL);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong address selection
-3 : Offset selected not available

11) i_PCI3600_Read32BitPortValue()

Syntax:

```
_INT_ i_PCI3600_Read32BitPortValue(DWORD dw_BoardHandle,
                                   BYTE     b_AddressSelection,
                                   DWORD dw_Offset,
                                   PDWORD pdw_Value)
```

Parameters:

- inputs:

DWORD dw_BoardHandle	Handle of the board
BYTE b_AddressSelection	Selection of the address of the board to do the access.
DWORD dw_Offset	Offset at witch the value must be read.

- Output

PDWORD pdw_Value	Value read.

Task:

Does a 32-bit input access (I/O or memory) on the board at offset *dw_Offset* from the address *b_AddressSelection*.

Calling convention:

```
DWORD dw_BoardHandle;
INT         i_ReturnValue;
DWORD dw_Value;

i_ReturnValue = i_PCI3600_Read32BitPortValue (dw_BoardHandle,
                                              0,
                                              0,
                                              &dw_Value);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong address selection
-3 : Offset selected not available

9.1.2 Analog input

12) i_PCI3600_InitAnalogInputModule()

Syntax:

```
_INT_ i_PCI3600_InitAnalogInputModule(DWORD dw_BoardHandle,
                                      BYTE      b_Module,
                                      PBYTE     pb_SingleDiffMode,
                                      PBYTE     pb_Coupling,
                                      PBYTE     pb_EnableCurrentSource,
                                      PBYTE     pb_GainSelection,
                                      BYTE
b_SamplingClockSelection,
                                      BYTE      b_SpeedMode)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Module	Index of the analog input module (0 to 3)
PBYTE	pb_SingleDiffMode	Single-Ended / Differential Mode of each channel pb_SingleDiffMode[0] : Left channel pb_SingleDiffMode[1] : Right channel 0 : Single-Ended 1 : Differential
PBYTE	pb_Coupling	Coupling mode of each channel pb_Coupling [0] : Left channel pb_Coupling [1] : Right channel 0 : DC 1 : AC
PBYTE	pb_EnableCurrentSource	Enable / Disable the current source for each channel pb_EnableCurrentSource[0] : Left channel pb_EnableCurrentSource[1] : Right channel 0 : Disable 1 : Enable Only available for the analog input module 0 and 1.
PBYTE	pb_GainSelection	Gain selection pb_GainSelection[0] : Left Channel pb_GainSelection[1] : Right Channel 0 : Gain 1 1 : Gain 10
BYTE	b_SamplingClockSelection	Sampling clock selection for the analog input module

		0: Sampling Clock 1 1 : Sampling Clock 2
BYTE	b_SpeedMode	ADC Speed mode selection 00 : Single speed mode (2KHz – 50KHz) 01 : Double speed mode (50KHz – 100KHz) 10 : Quadruple speed mode (100KHz – 200KHz)

Table 9-6: ADC Clocks

			Single Speed	Double Speed	Quad Speed
			00	01	10
Master clock (MHz)	**Divisor factor**	**Clock generator 1/2**	**Sampling frequency (Hz)**	**Sampling frequency (Hz)**	**Sampling frequency (Hz)**
102.4	4	25600000	50000	100000	200000
	5	20480000	40000	80000	160000
	6	17066667	33333	66667	133333
	8	12800000	25000	50000	100000
	10	10240000	20000		
	12	8533333	16667		
	16	6400000	12500		
	20	5120000	10000		
	25	4096000	8000		
	40	2560000	5000		
	50	2048000	4000		
	60	1706667	3333		
	80	1280000	2500		
	100	1024000	2000		

- Output
No output

Task:
Initialises the analog input module *b_Module*.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;
BYTE   pb_SingleDiffMode[2];
BYTE    pb_Coupling[2];
BYTE    pb_EnableCurrentSource[2];
BYTE    pb_EnableCurrentSource[2];
BYTE    pb_GainSelection[2];

i_ReturnValue = i_PCI3600_InitAnalogInputModule(dw_BoardHandle,
                                                0,
                                                pb_SingleDiffMode,
                                                pb_Coupling,
                                                pb_EnableCurrentSource,
                                                pb_GainSelection,
                                                0,

PCI3600_SAMPLING_CLOCK_1,

PCI3600_SINGLE_SPEED_MODE);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : Wrong Single/Diff Mode selection
-4 : Wrong coupling selection
-5 : Wrong current source flag
-6 : Wrong gain selection
-7 : Wrong clock selection
-8 : Wrong speed mode parameter
-9 : The board is not initialised
-10: Speed mode not available
-11: The current source is only available on the 2 first modules
-12: When the current source is used, the configuration of the channel must be single-ended and AC
-13: The module is already initialised by another process
-14: Initialisation Error

13) i_PCI3600_ReleaseAnalogInputModule()

Syntax:

```
_INT_ i_PCI3600_ReleaseAnalogInputModule (DWORD dw_BoardHandle,
                                          BYTE      b_Module)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Module	Index of the analog input module (0 to 3)

- **Output**

No output

Task:

Releases the analog input module *b_Module*.

Calling convention:

```
DWORD dw_BoardHandle;
INT   i_ReturnValue;

i_ReturnValue = i_PCI3600_ReleaseAnalogInputModule (dw_BoardHandle,
                                                     0);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : The module is not initialised or initialised by another process

14) i_PCI3600_GetAnalogInputModuleCalibrationStatus()

Syntax:

```
_INT_ i_PCI3600_GetAnalogInputModuleCalibrationStatus (DWORD
dw_BoardHandle,
                                                        BYTE
b_Module,
                                                        PBYTE
pb_Status)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Module	Index of the analog input module (0 to 3)

- Output

PBYTE	pb_Status	Status of the calibration 1 : Calibration is running 0 : End of the calibration : the module is ready

Task:

Gets the status of the calibration of the analog input module *b_Module*.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;
BYTE    b_Status;

i_ReturnValue =
i_PCI3600_GetAnalogInputModuleCalibrationStatus(dw_BoardHandle,
                                                                    0,

&b_Status);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number

15) i_PCI3600_StartAnalogInputModuleSingleAcquisition()

Syntax:

```
_INT_ i_PCI3600_StartAnalogInputModuleSingleAcquisition(DWORD
dw_BoardHandle,
                                                        BYTE
b_Module)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Module	Index of the analog input module (0 to 3)

- Output

No output

Task:

Start on the analog input module *b_Module* a single acquisition.

Calling convention:

```
DWORD dw_BoardHandle;
INT         i_ReturnValue;

i_ReturnValue =
i_PCI3600_StartAnalogInputModuleSingleAcquisition(dw_BoardHandle,
                                                                  0);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : The module is not initialised
-4 : The module is initialised for the cyclic acquisition mode
-5 : A conversion is already started
-6 : The calibration is running on the module

16) i_PCI3600_GetAnalogInputModuleSingleAcquisitionStatus()

Syntax:

```
_INT_ i_PCI3600_GetAnalogInputModuleSingleAcquisitionStatus
                                        (DWORD dw_BoardHandle,
                                         BYTE       b_Module,
                                         PBYTE    pb_Status)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Module	Index of the analog input module (0 to 3)

- Output

PBYTE	pb_Status	Status of the single acquisition (equal to the status of the FIFO) 0 : Value not available 1 : Value available

Task:

Returns the status of the single acquisition.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;
BYTE       b_Status;

i_ReturnValue = i_PCI3600_GetAnalogInputModuleSingleAcquisitionStatus
                                                    (dw_BoardHandle,
                                                     0,
                                                     &b_Status);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : The module is not initialised
-4 : The calibration is running on the module

17) i_PCI3600_ReadAnalogInputSingleAcquisitionValues()

Syntax:

```
_INT_ i_PCI3600_ReadAnalogInputSingleAcquisitionValues
                                    (DWORD dw_BoardHandle,
                                     BYTE b_Module,
                                     PDWORD pdw_Values)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Module	Index of the analog input module (0 to 3)

- Output

PDWORD	pdw_Values	Values of the single acquisition pdw_Values[0] : Value of the left channel pdw_Values[1] : Value of the right channel

Task:

Read the value of the right/left channel.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;
DWORD pdw_Values[2];

i_ReturnValue = i_PCI3600_ReadAnalogInputSingleAcquisitionValues
(dw_BoardHandle,
                                                                    0,

pdw_Values);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : The module is not initialised
-4 : The values are not available
-5 : The calibration is running on the module

18) i_PCI3600_GetAnalogInputModuleValues()

Syntax:

```
_INT_ i_PCI3600_GetAnalogInputModuleValues(DWORD dw_BoardHandle,
                                           BYTE     b_Module,
                                           PDWORD pdw_Values)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Module	Index of the analog input module (0 to 3)

- Output

PDWORD	pdw_Values	Values of the acquisition pdw_Values[0] : Value of the left channel pdw_Values[1] : Value of the right channel

Task:

Start and read the value of the right/left channel of the analog input module *b_Module*.
The function wait that the values are available to read it.

Calling convention:

```
DWORD dw_BoardHandle;
INT       i_ReturnValue;
DWORD pdw_Values[2];

i_ReturnValue = i_PCI3600_GetAnalogInputModuleValues(dw_BoardHandle,
                                                     0,
                                                     pdw_Values);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : The module is not initialised
-4 : Start analog input module acquisition error
-5 : Time function error
-6 : Get analog input module status error
-7 : Read analog input module values error
-8 : Time out occur

19) i_PCI3600_InitAnalogInputModuleCyclicAcquisition()

Syntax:

```
_INT_ i_PCI3600_InitAnalogInputModuleCyclicAcquisition
                                    (DWORD dw_BoardHandle,
                                     BYTE      b_Module,
                                     BYTE      b_Mode,
                                     DWORD dw_OnBoardBufferSize,
                                     DWORD dw_CompareValue)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Module	Index of the analog input module (0 to 3)
BYTE	b_Mode	Define the mode used 0: AUTO-STOP Mode In this mode, the On board buffer is filled with value. When the On board buffer is full, the acquisition is stopped and an interrupt occur. 1 : RING Buffer Mode In this mode, the on board buffer is always fill with value. In both modes: Each time the number of the value in the buffer reach the dw_CompareValue, an interrupt occur to permit to the software to read the value from the buffer.
DWORD	dw_OnBoardBufferSize	Size (in nbr of bytes : 1024 * 2 $^{dw\_OnBoardBufferSize}$) of the On-board buffer (in the on-board RAM) to used for the analog input module.
DWORD	dw_CompareValue	Define the number of acquisition to have in the on-board Buffer to generate an interrupt for start a DMA transfer.

- Output

No output

Task:

Initialise the cyclic acquisition for the analog input module *b_Module*.

Table 9-7: On-Board RAM buffer size

dw_OnBoardBufferSize	Nbr of values
0	256
1	512
2	1024
3	2048
..	…

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;

i_ReturnValue = i_PCI3600_ InitAnalogInputModuleCyclicAcquisition
                                        (dw_BoardHandle,
                                         0,
                                         PCI3600_AUTO_STOP_MODE,
                                         0,
                                         10);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : Wrong mode parameter
-4 : The module is not initialised
-5 : No computer buffer available
-6 : The compare value cannot be :
- greater that the size of the on board buffer in Auto Stop mode
- greater or equal that the size of the on board buffer in Ring Buffer mode
- null

-7 : The cyclic acquisition is already initialised by another process
-8 : The cyclic acquisition is running
-9 : Not enough place in the on board RAM
-10 : The interrupt routine is not installed
-11 : Primary Init Error
-12 : The calibration is running on the module

20) i_PCI3600_StartAnalogInputModuleCyclicAcquisition()

Syntax:

```
_INT_ i_PCI3600_StartAnalogInputModuleCyclicAcquisition(DWORD
dw_BoardHandle,
                                                        BYTE
b_Module)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Module	Index of the analog input module (0 to 3).

- Output

No output

Task:

Starts the analog input cyclic acquisition for the module *b_Module*.

Calling convention:

```
DWORD dw_BoardHandle;
INT         i_ReturnValue;

i_ReturnValue = i_PCI3600_StartAnalogInputCyclicAcquisition
(dw_BoardHandle,
                                                        0);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : The Cyclic acquisition is not initialised or not by this process

21) i_PCI3600_GetAnalogInputModuleCyclicAcquisitionStatus()

Syntax:

```
_INT_ i_PCI3600_GetAnalogInputModuleCyclicAcquisitionStatus
                                        (DWORD dw_BoardHandle,
                                         BYTE b_Module,
                                         PBYTE pb_Status,
                                         PDWORD pdw_NbrOfValue)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Module	Index of the analog input module (0 to 3)

- Output

PBYTE	pb_Status	Status of the cyclic acquisition 0: not started 1: started
PDWORD	pdw_NbrOfValue	Number of value in the buffer in the on-board RAM for the module.

Task:

Returns the status and the number of value in the on-board RAM for the module *b_Module*.

Calling convention:

```
DWORD dw_BoardHandle;
INT       i_ReturnValue;
BYTE      b_Status;
DWORD     dw_NbrOfValue;

i_ReturnValue = i_PCI3600_GetAnalogInputCyclicAcquisitionStatus
(dw_BoardHandle,
                                                                    0,

&b_Status,

&dw_NbrOfValue);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : The module is not initialised

22) i_PCI3600_IncrementAnalogInputCyclicAcquisitionBufferReadPointer()

Syntax:

```
_INT_ i_PCI3600_IncrementAnalogInputCyclicAcquisitionBufferReadPointer
                                        (DWORD dw_BoardHandle,
                                         DWORD dw_NbrOfValue)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
DWORD	dw_NbrOfValue	Number of value.

- Output

Task:

Permits to increment the read pointer of the ring buffer used to do the DMA transfer.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;
DWORD dw_BufferHandle;

i_ReturnValue =
i_PCI3600_IncrementAnalogInputCyclicAcquisitionBufferReadPointer

(dw_BoardHandle,
                                                        1000);
```

Return Value:

0 : No Error

-1 : The handle parameter of the board is wrong

23) i_PCI3600_StopAnalogInputModuleCyclicAcquisition()

Syntax:
INT i_PCI3600_StopAnalogInputModuleCyclicAcquisition(DWORD dw_BoardHandle,
BYTE b_Module)

Parameters:
- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Module	Index of the analog input module (0 to 3)

- Output
No output

Task:
Stops the analog input cyclic acquisition for the module *b_Module*.

Calling convention:
```
DWORD dw_BoardHandle;
INT          i_ReturnValue;

i_ReturnValue = i_PCI3600_StopAnalogInputModuleCyclicAcquisition
(dw_BoardHandle,
                                                                0);
```

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : The Cyclic Acquisition is not started or started by another process

24) i_PCI3600_ReleaseAnalogInputModuleCyclicAcquisition()

Syntax:

```
_INT_ i_PCI3600_ReleaseAnalogInputModuleCyclicAcquisition
                                                (DWORD dw_BoardHandle,
                                                 BYTE       b_Module)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Module	Index of the analog input module (0 to 3)

- Output

No output

Task:

Releases the analog input cyclic acquisition for the module *b_Module*.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;

i_ReturnValue = i_PCI3600_ReleaseAnalogInputModuleCyclicAcquisition
                                                (dw_BoardHandle,
                                                 0);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : The module is not initialised

25) i_PCI3600_AnalogInput_EnableDisableDigitalInputExternalTrigger()

Syntax:

```
_INT_ i_PCI3600_AnalogInput_EnableDisableDigitalInputExternalTrigger
                                (DWORD dw_BoardHandle,
                                 BYTE      b_Module,
                                 BYTE      b_ExternTriggerFlag)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Module	Index of the analog input module (0 to 3)
BYTE	b_ExternTriggerFlag	Enable/disable flag for the digital input external trigger 0 : Disable 1 : Enable

- Output

No output

Task:

Enables/disables the digital input external trigger (digital input 0) for the analog input module *b_Module*.

Calling convention:

```
DWORD dw_BoardHandle;
INT         i_ReturnValue;

i_ReturnValue =
i_PCI3600_AnalogInput_EnableDisableDigitalInputExternalTrigger
                                                    (dw_BoardHandle,
                                                     0,
                                                     PCI3600_ENABLE);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : Extern trigger flag is wrong
-4 : The module is not initialised
-5 : The digital external trigger is not initialised

9.1.3 Analog output

26) i_PCI3600_InitAnalogOutputChannel()

Syntax:

```
_INT_ i_PCI3600_InitAnalogOutputChannel(DWORD dw_BoardHandle,
                                        BYTE      b_Channel,
                                        BYTE
b_SamplingClockSelection,
                                        BYTE      b_SpeedMode)
```

Parameters:

- inputs:

DWORD dw_BoardHandle — Handle of the board

BYTE b_Channel — Index of the analog output channel (0 to 1)

BYTE b_SamplingClockSelection — Clock selection for the analog output channel
0 : Sampling clock 1
1 : Sampling clock 2

BYTE b_SpeedMode — DAC speed mode selection
00 : Single speed mode (2KHz – 50KHz)
01 : Double speed mode (50KHz – 100KHz)
10 : Quadruple speed mode (100KHz – 200KHz)

Table 9-8: DAC sampling clock

			DAC Sampling Clock		
			Single speed	Double speed	Quad speed
			00	01	10
PLD_CLK	Divisor	Clock Gen 1/2	LDAC (Hz)	LDAC (Hz)	LDAC (Hz)
			CLK GEN/512	CLK GEN/256	CLK GEN/128
102400000	4	25600000	50000	100000	200000
102400000	5	20480000	40000	80000	160000
102400000	6	17066667	33333	66667	133333
102400000	8	12800000	25000	50000	100000
102400000	10	10240000	20000		
102400000	12	8533333	16667		
102400000	16	6400000	12500		
102400000	20	5120000	10000		
102400000	25	4096000	8000		
102400000	40	2560000	5000		
102400000	50	2048000	4000		
102400000	60	1706667	3333		
102400000	80	1280000	2500		
102400000	100	1024000	2000		

- **Output**
  No output

Task:
Initialises the analog output channel *b_Channel*.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;

i_ReturnValue = i_PCI3600_InitAnalogOutputChannel
                                        (dw_BoardHandle,
                                         0,
                                         PCI3600_SAMPLING_CLOCK_1,

PCI3600_SINGLE_SPEED_MODE);
```

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong channel number
-3 : Wrong clock selection
-4 : Wrong Speed mode parameter
-5 : The board is not initialised
-6 : Speed mode not available
-7 : The channel is already initialised by another process
-8 : Initialisation Error

27) i_PCI3600_ReleaseAnalogOutputChannel()

Syntax:

```
_INT_ i_PCI3600_ReleaseAnalogOutputChannel(DWORD dw_BoardHandle,
                                           BYTE      b_Channel)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Index of the analog output channel (0 to 1)

- Output

No output

Task:

Releases the analog output channel *b_Channel*.

Calling convention:

```
DWORD dw_BoardHandle;
INT   i_ReturnValue;

i_ReturnValue = i_PCI3600_ReleaseAnalogOutputChannel (dw_BoardHandle,
                                                       0);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong channel number
-3 : The channel is not initialised or initialised by another process

28) i_PCI3600_GetAnalogOutputReadyBitStatus()

Syntax:

```
_INT_ i_PCI3600_GetAnalogOutputReadyBitStatus(DWORD dw_BoardHandle,
                                              BYTE      b_Channel,
                                              PBYTE
pb_ReadyBitStatus)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Index of the analog output channel (0 to 1)

- Output

PBYTE	pb_ReadyBitStatus	Status of the ready bit for the analog output channel 0: channel not ready to receive a value 1: channel ready to receive a value

Task:

Returns the status of the ready bit to know if a value can be written on the analog output channel *b_Channel*.

Calling convention:

```
DWORD dw_BoardHandle;
INT       i_ReturnValue;
BYTE      b_ReadyBitStatus;

i_ReturnValue = i_PCI3600_GetAnalogOutputReadyBitStatus (dw_BoardHandle,
                                                          0,

&b_ReadyBitStatus);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong channel number
-3 : The channel is not initialised
-4 : The channel is initialised for the signal generator mode

29) i_PCI3600_WriteAnalogOutputValue()

Syntax:

```
_INT_ i_PCI3600_WriteAnalogOutputValue(DWORD dw_BoardHandle,
                                       BYTE      b_Channel,
                                       WORD      w_Value)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Index of the analog output channel (0 to 1)
WORD	w_Value	Value to write on the analog output (0 to 65535)

- Output

No output

Task:

Writes the value *w_Value* on the analog output *b_Channel*.

Calling convention:

```
DWORD dw_BoardHandle;
INT   i_ReturnValue;

i_ReturnValue = i_PCI3600_WriteAnalogOutputValue (dw_BoardHandle,
                                                   0,
                                                   16383);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong channel number
-3 : The channel is not initialised
-4 : This function cannot work if the digital external trigger is enabled
-5 : The channel is initialised for the signal generator mode
-6 : The channel is not ready to receive a new value

30) i_PCI3600_Set1AnalogOutputChannel()

Syntax:

```
_INT_ i_PCI3600_Set1AnalogOutputChannel (DWORD dw_BoardHandle,
                                          BYTE     b_Channel,
                                          WORD     w_Value)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Index of the analog output channel (0 to 1)
WORD	w_Value	Value to write on the analog output (0 to 65535)

- Output

No output

Task:

Sets the analog output channel *b_Channel* to the value *w_Value*. The ready bit is tested in the function.

Calling convention:

```
DWORD dw_BoardHandle;
INT   i_ReturnValue;

i_ReturnValue = i_PCI3600_Set1AnalogOutputChannel (dw_BoardHandle,
                                                    0,
                                                    16383);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong channel number
-3 : The channel is not initialised
-4 : This function cannot work if the digital external trigger is enabled
-5 : The channel is initialised for the signal generator mode
-6 : Time function error
-7 : Time out occur

31) i_PCI3600_ReserveAnalogOutputSignalGeneratorComputerBuffer()

Syntax:

```
_INT_ i_PCI3600_ReserveAnalogOutputSignalGeneratorComputerBuffer
                                        (DWORD dw_BoardHandle,
                                         BYTE      b_Channel,
                                         DWORD   dw_BufferSize)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Index of the analog output channel (0 to 1)
DWORD	dw_BufferSize	Size (in nbr of acquisition) of the buffer to used for the analog output module.

- Output

No output

Task:

Reserves a part of the computer buffer allocated by ADDIREG for the analog output *b_Channel*.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;

i_ReturnValue =
i_PCI3600_ReserveAnalogOutputSignalGeneratorComputerBuffer
                                        (dw_BoardHandle,
                                         0,
                                         1000);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong channel number
-3 : Buffer Size error
-4 : The buffer is already reserved by another process
-5 : The PC DMA Buffer is in use
-6 : The channel is initialised in signal generator mode

32) i_PCI3600_FreeAnalogOutputSignalGeneratorComputerBuffer()

Syntax:

```
_INT_ i_PCI3600_FreeAnalogOutputSignalGeneratorComputerBuffer
                                                    (DWORD
dw_BoardHandle,
                                                     BYTE      b_Channel)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Index of the analog output channel (0 to 1)

- Output

No output

Task:

Releases the part of the computer buffer allocated by ADDIREG for the analog output *b_Channel*.

Calling convention:

```
DWORD dw_BoardHandle;
INT         i_ReturnValue;

i_ReturnValue = i_PCI3600_FreeAnalogOutputSignalGeneratorComputerBuffer
                                        (dw_BoardHandle,
                                          0);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong channel number
-3 : The buffer is not reserved or reserved by another process
-4 : The channel is initialised in signal generator mode

33) i_PCI3600_WriteAnalogOutputSignalGeneratorComputerBufferValues()

Syntax:

```
_INT_ i_PCI3600_WriteAnalogOutputSignalGeneratorComputerBufferValues
                                                    (DWORD
dw_BoardHandle,
                                                     BYTE      b_Channel,
                                                     DWORD
                                    dw_NbrOfValue,
                                                     PWORD   pw_ValueArray,
                                                     PDWORD
                                    pdw_NbrOfWrittenValues,
                                                     PDWORD
                                    pdw_NbrOfFreeValues)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Index of the analog output channel (0 to 1)
DWORD	dw_NbrOfValue	Number of value to write in the buffer
PWORD	pw_ValueArray	Value to write in the computer RAM buffer.

- Output

PDWORD pdw_NbrOfWrittenValues
Number of values written in the computer buffer.

PDWORD pdw_NbrOfFreeValues
Number of values who can be written in the computer buffer.

Task:

Writes analog output values for the analog output *b_Channel* in the computer RAM buffer for this channel.

Calling convention:

```
DWORD dw_BoardHandle;
INT         i_ReturnValue;
WORD      pw_ValueArray[200];
DWORD   dw_NbrOfWrittenValues;
DWORD   dw_NbrOfFreeValues;

i_ReturnValue =
i_PCI3600_WriteAnalogOutputSignalGeneratorComputerBufferValues
                                        (dw_BoardHandle,
                                         0,
                                         100,
                                         pw_ValueArray,
                                         &dw_NbrOfWrittenValues,
```

&dw_NbrOfFreeValues);

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong channel number
-3 : The buffer is not reserved
-4 : The buffer is full

34) i_PCI3600_InitAnalogOutputSignalGenerator()

Syntax:

```
_INT_ i_PCI3600_InitAnalogOutputSignalGenerator(DWORD
dw_BoardHandle,
                                                BYTE      b_Channel,
                                                BYTE      b_Mode,
                                                DWORD
dw_OnBoardBufferSize,
                                                DWORD
                                  dw_CompareValue)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Index of the analog output channel (0 to 1)
BYTE	b_Mode	Mode of the analog output channel **0 : Free Run Mode** In this mode, the on-board RAM buffer is filled one time and then this value are always given to the analog output channel. The reserved computer buffer size for this analog output must be equal to the On-board buffer size. No under run error can occur in this mode. And no interrupt occur in this mode. **1 : Ring Buffer Mode** In this mode, the on-board RAM buffer is filled before the start and then the value are given to the analog output channel. When the minimal number of value is reached, an interrupt occur and the software load the buffer with new values of the computer buffer (the user loads new values in the computer buffer with the following function: i_PCI3600_WriteAnalogOutputSignalGeneratorComputerBufferValues(…)). When the local DMA transfer is too speed, an under run error occur.
DWORD	dw_OnBoardBufferSize	Size (= n from 1024 * 2^n bytes) of the On-board buffer (in the on-board RAM) to used for the analog output module.
DWORD	dw_CompareValue	Define the number minimum of free place of the on-board RAM to generate an interrupt.

- Output

No output

Task:

Initialise the signal generator for the analog output channel *b_Channel*.
Load the on-board RAM buffer with the values of the computer buffer.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;

i_ReturnValue = i_PCI3600_InitAnalogOutputSignalGenerator
                                        (dw_BoardHandle,
                                         0,
                                         PCI3600_FREE_MODE,
                                         0,
                                         15);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong channel number
-3 : Wrong mode parameter
-4 : The channel is not initialised
-5 : The buffer is not reserved
-6 : The compare value cannot be greater that the size of the on board buffer and cannot be null
-7 : The Signal generator is already initialised by another process
-8 : The signal generator is running
-9 : Not enough place in the on board RAM
-10 : A cycle acquisition or a signal generator is running
-11 : The compare value must be greater than the actual free place in the on board RAM.
-12 : The size of the PC RAM buffer must be the same that the On Board Buffer size and it must be full
-13 : The interrupt routine is not installed
-14 : Time function error
-15 : Time out occur
-16 : Prepare DMA transfer Error

35) i_PCI3600_StartAnalogOutputSignalGenerator()

Syntax:
INT i_PCI3600_StartAnalogOutputSignalGenerator(DWORD dw_BoardHandle,
BYTE b_Channel)

Parameters:
- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Index of the analog output channel (0 to 1)

- **Output**
No output

Task:
Start the signal generator of the analog output channel *b_Channel*.

Calling convention:
```
DWORD dw_BoardHandle;
INT   i_ReturnValue;

i_ReturnValue = i_PCI3600_StartanalogOutputSignalGenerator
(dw_BoardHandle,
                                                    0);
```

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong channel number
-3 : The signal generator is not initialised or not by this process
-4 : The signal generator is already started

36) i_PCI3600_GetAnalogOutputSignalGeneratorStatus()

Syntax:

```
_INT_ i_PCI3600_GetAnalogOutputSignalGeneratorStatus
                                        (DWORD dw_BoardHandle,
                                         BYTE      b_Channel,
                                         PBYTE    pb_Status,
                                         PDWORD
pdw_NbrOfValueInOnBoardRAM,
                                         PDWORD
pdw_NbrOfValueInComputerRAM)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Index of the analog output channel (0 to 1)

- Output

PBYTE	pb_Status	Status of the signal generator 0 : not started 1 : started
PDWORD	pdw_NbrOfValueInOnBoardRAM	Number of the value in the on-board RAM buffer.
PDWORD	pdw_NbrOfValueInComputerRAM	Number of the value in the computer RAM buffer.

Task:

Return the status and the number of value in the on-board RAM and in the computer RAM for the analog output channel *b_Channel*.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;
BYTE       b_Status;
DWORD   dw_NbrOfValueOnBoardRAM;
DWORD   dw_NbrOfValueComputerRAM;

i_ReturnValue = i_PCI3600_GetAnalogOutputSignalGeneratorStatus
                                        (dw_BoardHandle,
                                         0,
                                         &b_Status,
```

&dw_NbrOfValueOnBoardRAM,

&dw_NbrOfValueComputerRAM);

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong channel number
-3 : The signal generator is not started or started by another process

37) i_PCI3600_StopAnalogOutputSignalGenerator()

Syntax:
INT i_PCI3600_StopAnalogOutputSignalGenerator(DWORD dw_BoardHandle,
BYTE b_Channel)

Parameters:
- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Index of the analog output channel (0 to 1)

- Output
No output

Task:
Start the signal generator of the analog output channel *b_Channel*.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;

i_ReturnValue =
i_PCI3600_StopAnalogOutputSignalGenerator(dw_BoardHandle,
                                                                    0) ;
```

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong channel number
-3 : The signal generator is not started or started by another process

38) i_PCI3600_ReleaseAnalogOutputSignalGenerator()

Syntax:
INT i_PCI3600_ReleaseAnalogOutputSignalGenerator(DWORD dw_BoardHandle,
BYTE b_Channel)

Parameters:
- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Index of the analog output channel (0 to 1)

- Output
No output

Task:
Release the signal generator of the analog output channel *b_Channel*.

Calling convention:
```
DWORD dw_BoardHandle;
INT   i_ReturnValue;

i_ReturnValue = i_PCI3600_ReleaseAnalogOutputSignalGenerator
(dw_BoardHandle,
                                                        0);
```

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong channel number
-3 : The signal generator is not initialised or not initialised by this process

39) i_PCI3600_AnalogOutput_EnableDisableDigitalInputExternalTrigger()

Syntax:

```
_INT_ i_PCI3600_AnalogOutput_EnableDisableDigitalInputExternalTrigger
                                                        (DWORD
dw_BoardHandle,
                                                         BYTE
b_Channel,
                                                         BYTE
b_ExternTriggerFlag)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Index of the analog output channel (0 to 1)
BYTE	b_ExternTriggerFlag	Enable/disable Flag for the digital input external trigger 0 : Disable 1 : Enable

- Output

No output

Task:

Enabled/Disable the digital input external trigger (digital input 0) for the analog output channel *b_Channel*.

- Output

No output

Calling convention:

```
DWORD dw_BoardHandle;
INT         i_ReturnValue;

i_ReturnValue =
i_PCI3600_AnalogOutput_EnableDisableDigitalInputExternalTrigger
                                                    (dw_BoardHandle,
                                                     0,
                                                     PCI3600_ENABLE);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong channel number
-3 : Extern trigger flag is wrong
-4 : The channel is not initialised
-5 : The digital external trigger is not initialised

9.1.4 Chronometer module

40) i_PCI3600_InitChronometerModule()

Syntax:

```
_INT_ i_PCI3600_InitChronometerModule(DWORD dw_BoardHandle,
                                      BYTE      b_ChronometerModule,
                                      BYTE      b_ClockDivisor)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_ChronometerModule	Index of the chronometer module (0 to 3)
BYTE	b_InputDivisor	Input divisor (0 to 15) 0 : 2^0 -> divisor = 1 1 : 2^1 -> divisor = 2 …

- Output

No output

Task:

Initialises the chronometer module *b_ChronometerModule*.

Calling convention:

```
DWORD dw_BoardHandle;
INT       i_ReturnValue;

i_ReturnValue = i_PCI3600_InitChronometerModule (dw_BoardHandle,
                                                 0,
                                                 1);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong chronometer number
-3 : Wrong clock divisor selection
-4 : The Chronometer is already initialised by another process
-5 : Initialisation Error

41) i_PCI3600_ReleaseChronometerModule()

Syntax:
INT i_PCI3600_ReleaseChronometerModule (DWORD dw_BoardHandle,
BYTE b_ChronometerModule)

Parameters:
- inputs:
DWORD dw_BoardHandle Handle of the board
BYTE b_ChronometerModule
Index of the chronometer module (0 to 3)

- Output
No output

Task:
Release the chronometer module *b_ChronometerModule*.

Calling convention:
DWORD dw_BoardHandle;
INT i_ReturnValue;

i_ReturnValue = i_PCI3600_ReleaseChronometerModule (dw_BoardHandle,
0);

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong Chronometer number
-3 : The Chronometer is not initialised or initialised by another process

42) i_PCI3600_StartChronometerModuleAcquisition()

Syntax:
INT i_PCI3600_StartChronometerModuleAcquisition(DWORD dw_BoardHandle,
BYTE b_ChronometerModule)

Parameters:
- inputs:
DWORD dw_BoardHandle Handle of the board
BYTE b_ChronometerModule
Index of the chronometer module (0 to 3)

- Output
No output

Task:
Starts the chronometer module acquisition.
When this function is used, the value of the chronometer is latched and saved in the internal FIFO of the board.
You can stop the acquisition with the *i_PCI3600_StopChronometerModuleAcquisition()*.
The acquisition is automatically stopped when the FIFO is full.

Calling convention:
DWORD dw_BoardHandle;
INT i_ReturnValue;

i_ReturnValue = i_PCI3600_StartChronometerModuleAcquisition (dw_BoardHandle,
0);

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong Chronometer number
-3 : The Chronometer is not initialised
-4 : The acquisition is already started

43) i_PCI3600_GetChronometerModuleFIFOStatus()

Syntax:

```
_INT_ i_PCI3600_GetChronometerModuleFIFOStatus(DWORD
dw_BoardHandle,
                                                BYTE
b_ChronometerModule,
                                                PBYTE
pb_FIFOStatus)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_ChronometerModule	Index of the chronometer module (0 to 3)

- Output

PBYTE	pb_FIFOStatus	internal FIFO status 00 : FIFO empty 01 : FIFO not empty 10 : FIFO error (Overflow)

Task:

Returns the status of the internal FIFO to know if the chronometer values are available.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;
BYTE       b_FIFOStatus;

i_ReturnValue = i_PCI3600_GetChronometerModuleFIFOStatus
(dw_BoardHandle,
                                                    0,

&b_FIFOStatus);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong Chronometer number
-3 : The Chronometer is not initialised

44) i_PCI3600_ReadChronometerModuleValue()

Syntax:

```
_INT_ i_PCI3600_ReadChronometerModuleValue
                                        (DWORD dw_BoardHandle,
                                         BYTE      b_ChronometerModule,
                                         PDWORD pdw_ChronometerValue)
```

Parameters:

- inputs:

DWORD dw_BoardHandle — Handle of the board

BYTE b_ChronometerModule — Index of the chronometer module (0 to 3)

- Output

PDWORD pdw_ChronometerValue — Chronometer value read from the internal FIFO.

Task:

Reads one chronometer value from the internal FIFO.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;
DWORD   dw_ChronometerValue;

i_ReturnValue = i_PCI3600_ReadChronometerModuleValue (dw_BoardHandle,
                                                                    0,

&dw_ChronometerValue);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong Chronometer number
-3 : The Chronometer is not initialised
-4 : FIFO is empty

45) i_PCI3600_StopChronometerModuleAcquisition()

Syntax:
INT i_PCI3600_StopChronometerModuleAcquisition(DWORD dw_BoardHandle,
BYTE b_ChronometerModule)

Parameters:
- inputs:
DWORD dw_BoardHandle Handle of the board
BYTE b_ChronometerModule
Index of the chronometer module (0 to 3)

- Output
No output

Task:
Stop the acquisition of the chronometer module value..

Calling convention:
DWORD dw_BoardHandle;
INT i_ReturnValue;

i_ReturnValue = i_PCI3600_StopChronometerModuleAcquisition (dw_BoardHandle,
0);

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong Chronometer number
-3 : The Chronometer is not initialised or not initialised by this process

46) i_PCI3600_InitChronometerModuleCyclicAcquisition

Syntax:

```
_INT_ i_PCI3600_InitChronometerModuleCyclicAcquisition
                                        (DWORD dw_BoardHandle,
                                         BYTE b_ChronometerModule,
                                         BYTE b_Mode,
                                         DWORD
dw_OnBoardBufferSize,
                                         DWORD dw_CompareValue)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_ChronometerModule	Index of the chronometer module (0 to 3)
BYTE	b_Mode	Define the mode used **0 : AUTO-STOP Mode** In this mode, the On board buffer is fill with value. When the On board buffer is full, the acquisition is stopped and an interrupt occur. **1 : RING Buffer Mode** In this mode, the on board buffer is always fill with value. **In both modes:** Each time the number of the value in the buffer reach the dw_CompareValue, an interrupt occur to permit to the software to read the value from the buffer.
DWORD	dw_OnBoardBufferSize	Size (in nbr of bytes : 1024 * 2 $^{dw\_OnBoardBufferSize}$) of the On-board buffer (in the on-board RAM) to used for the chrono.
DWORD	dw_CompareValue	Define the number of acquisition to have in the on-board Buffer to generate an interrupt for start a DMA transfer.

- Output

No output

Task:

Initialise the cyclic acquisition for the chronometer module *b_ChronometerModule*.

Calling convention:

```
DWORD dw_BoardHandle;
INT       i_ReturnValue;
```

```
i_ReturnValue = i_PCI3600_ InitChronometerModuleCyclicAcquisition
                                        (DWORD dw_BoardHandle,
                                         0,
                                         PCI3600_AUTO_STOP_MODE,
                                         0,
                                         10);
```

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : Wrong mode parameter
-4 : The module is not initialised
-5 : No computer buffer available
-6 : The compare value cannot be :
- greater that the size of the on board buffer in Auto Stop mode
- greater or equal that the size of the on board buffer in Ring Buffer mode
- null

-7 : The cyclic acquisition is already initialised by another process
-8 : The cyclic acquisition is running
-9 : Not enough place in the on board RAM
-10 : The interrupt routine is not installed
-11 : Init Chronometer Error

47) i_PCI3600_StartChronometerModuleCyclicAcquisition()

Syntax:

```
_INT_ i_PCI3600_StartChronometerModuleCyclicAcquisition
                                                    (DWORD dw_BoardHandle,
                                                     BYTE
b_ChronometerModule)
```

Parameters:

- inputs:

DWORD dw_BoardHandle Handle of the board

BYTE b_ChronometerModule
Index of the chronometer module (0 to 3)

- Output

No output

Task:

Starts the cyclic acquisition for the chronometer module *b_ChronometerModule*.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;

i_ReturnValue = i_PCI3600_StartChronometerModuleCyclicAcquisition
(dw_BoardHandle,
                                                                    0);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : The Cyclic acquisition is not initialised or not by this process
-4 : The Cyclic acquisition is already started

48) i_PCI3600_GetChronometerModuleCyclicAcquisitionStatus()

Syntax:

```
_INT_ i_PCI3600_GetChronometerModuleCyclicAcquisitionStatus
                                                    (DWORD
dw_BoardHandle,
                                                     BYTE
b_ChronometerModule,
                                                     PBYTE    pb_Status,
                                                     PDWORD
pdw_NbrOfValue)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_ChronometerModule	Index of the chronometer module (0 to 3)

- Output

PBYTE	pb_Status	Status of the cyclic acquisition 0: not started 1: started
PDWORD	pdw_NbrOfValue	Number of value in the buffer in the on-board RAM for the chronometer.

Task:

Return the status and the number of value in the on-board RAM for the chronometer module *b_ChronometerModule*.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;
BYTE b_Status;
DWORD dw_NbrOfValue;

i_ReturnValue = i_PCI3600_GetChronometerModuleCyclicAcquisitionStatus

(dw_BoardHandle,
                                                    0,
                                                    &b_Status,

&dw_NbrOfValue);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : The Cyclic Acquisition is not started or started by another Process.

49) i_PCI3600_IncrementChronometerCyclicAcquisitionBufferReadPointer()

Syntax:

```
_INT_ i_PCI3600_IncrementChronometerCyclicAcquisitionBufferReadPointer
                                        (DWORD dw_BoardHandle,
                                         DWORD dw_NbrOfValue)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
DWORD	dw_NbrOfValue	Number of value.

- **Output**

Task:

Permits to increment the read pointer of the ring buffer used to do the DMA transfer.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;
DWORD dw_BufferHandle;

i_ReturnValue =
i_PCI3600_IncrementChronometerCyclicAcquisitionBufferReadPointer

(dw_BoardHandle,
                                                        1000);
```

Return Value:

0 : No Error

-1 : The handle parameter of the board is wrong

50) i_PCI3600_StopChronometerModuleCyclicAcquisition()

Syntax:

```
_INT_ i_PCI3600_StopChronometerModuleContinuousAcquisition
                                                    (DWORD
dw_BoardHandle,
                                                     BYTE
b_ChronometerModule)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_ChronometerModule	Index of the chronometer module (0 to 3)

- Output

No output

Task:

Stops the cyclic acquisition for the chronometer module *b_ChronometerModule*.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;

i_ReturnValue =
i_PCI3600_StopChronometerModuleContinuousAcquisition(dw_BoardHandle,
                                                                    0);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : The Cyclic Acquisition is not started or started by another process.

51) i_PCI3600_ReleaseChronometerModuleCyclicAcquisition()

Syntax:

```
_INT_ i_PCI3600_ReleaseChronometerModuleCyclicAcquisition
                                                    (DWORD dw_BoardHandle,
                                                     BYTE
b_ChronometerModule)
```

Parameters:

- inputs:

DWORD dw_BoardHandle Handle of the board
BYTE b_ChronometerModule
Index of the chronometer module (0 to 3)

- Output

No output

Task:

Releases the cyclic acquisition for the chronometer module *b_ChronometerModule*.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;

i_ReturnValue = i_PCI3600_ReleaseChronometerModuleCyclicAcquisition

(dw_BoardHandle,
                                                        0);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong module number
-3 : The Cyclic Acquisition is not initialised or not initialised by this process

52) i_PCI3600_Chronometer_EnableDisableDigitalInputExternalTrigger()

Syntax:

```
_INT_ i_PCI3600_Chronometer_EnableDisableDigitalInputExternalTrigger
                                (DWORD dw_BoardHandle,
                                 BYTE      b_ChronometerModule,
                                 BYTE      b_ExternTriggerFlag)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_ChronometerModule	Index of the chronometer module (0 to 3)
BYTE	b_ExternTriggerFlag	Enable/disable Flag for the digital input external trigger 0 : Disable 1 : Enable

- Output

No output

Task:

Enables/disables the digital input external trigger (digital input 0) for the chronometer module *b_ChronometerModule*.

Calling convention:

```
DWORD dw_BoardHandle;
INT         i_ReturnValue;

i_ReturnValue =
i_PCI3600_Chronometer_EnableDisableDigitalInputExternalTrigger
                                        (dw_BoardHandle,
                                         0,
                                         PCI3600_ENABLE);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong chronometer number
-3 : External trigger flag is wrong
-4 : The chronometer is not initialised
-5 : The digital external trigger is not initialised

9.1.5 Digital inputs

53) i_PCI3600_Read1DigitalInput()

Syntax:

```
_INT_ i_PCI3600_Read1DigitalInput(DWORD dw_BoardHandle,
                                  BYTE b_Channel,
                                  PBYTE pb_ChannelValue)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Channel to read (0-7)

- Output

PBYTE	pb_ChannelValue	Channel value : 0 : Low 1 : High

Task:

Indicates the status of an input channel. The variable *b_Channel* passes the input channel to be read (0 to 7). A value is returned by the variable *pb_ChannelValue* : 0 (low) or 1 (high).

Calling convention:

```
DWORD dw_BoardHandle;
INT       i_ReturnValue;
BYTE b_ChannelValue;

i_ReturnValue = i_PCI3600_Read1DigitalInput (dw_BoardHandle,
                                              0,
                                              &b_ChannelValue);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : The channel number parameter is wrong

54) i_PCI3600_Read4DigitalInputs()

Syntax:

```
_INT_ i_PCI3600_Read4DigitalInputs(DWORD dw_BoardHandle,
                                   BYTE b_Port,
                                   PBYTE pb_PortValue)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Port	Number of the input port you want to read (0 or 1)

- Output

PBYTE	pb_PortValue	State of the digital input port (0 to 15)

Task:

Indicates the status of a port. The variable *b_Port* passes the port to be read (1 or 2). A value is returned with the variable *pb_PortValue* .

Calling convention:

```
DWORD dw_BoardHandle;
INT       i_ReturnValue;
BYTE      b_PortValue;

i_ReturnValue = i_PCI3600_Read4DigitalInputs (dw_BoardHandle,
                                              0,
                                              &b_PortValue);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : The channel number parameter is wrong

55) i_PCI3600_Read8DigitalInputs()

Syntax:

```
_INT_ i_PCI3600_Read8DigitalInputs(DWORD dw_BoardHandle,
                                   PBYTE pb_Value)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board

- Output

PBYTE	pb_Value	State of the digital inputs (0 to 255)

Task:

Indicates the state of the digital inputs.

Calling convention:

```
DWORD dw_BoardHandle;
INT       i_ReturnValue;
BYTE      b_Value;

i_ReturnValue = i_PCI3600_Read8DigitalInputs (dw_BoardHandle,
                                              &b_Value);
```

Return Value:

0 : No Error

-1 : The handle parameter of the board is wrong

9.1.6 Digital outputs

56) i_PCI3600_EnableDisableDigitalOutputMemory()

Syntax:
INT i_PCI3600_EnableDisableDigitalOutputMemory (DWORD dw_BoardHandle,
BYTE b_Enable)

Parameters:
- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Enable	0 : deactivates the digital output memory 1 : activates the digital output memory

- Output
No output

Task:
Enables/disables the digital output memory.
After calling activate the digital output memory, the output channels you have previously activated with the functions "i_PCI3600_SetXDigitalOutputOn" are not reset.
You can reset them with the function "i_PCI3600_SetXDigitalOutputOff".

Calling convention:
DWORD dw_BoardHandle;
INT i_ReturnValue;

i_ReturnValue = i_PCI3600_EnableDisableDigitalOutputMemory (dw_BoardHandle,

PCI3600_ENABLE);

Return Value:
0 : No Error
-1 : The handle parameter of the board is wrong

57) i_PCI3600_Set1DigitalOutputOn()

Syntax:

```
_INT_ i_PCI3600_Set1DigitalOutputOn (DWORD dw_BoardHandle,
                                     BYTE b_Channel)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Number of the output channel you want to set (0 to 7).

- Output

No output

Task:

Sets the output channel which has been passed with *b_Channel*.
Setting an output channel means setting an output channel on high.
***Switching on the digital output memory* (ON)**
see function "i_PCI3600_EnableDisableDigitalOutputMemory (...)"
b_Channel= 1
The output channel 1 is set. The others output channels hold their state.
***Switching off the digital output memory* (OFF)**
see function " i_PCI3600_EnableDisableDigitalOutputMemory (...)"
b_Channel= 1
The output channel 1 is set. The others output channels are reset.
If you have switched off the digital output memory (OFF), all the others input channels are set to "0".

Calling convention:

```
DWORD dw_BoardHandle;
INT   i_ReturnValue;

i_ReturnValue = i_PCI3600_Set1DigitalOutputOn(dw_BoardHandle,
                                              0);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : The channel number is not between 0 and 7

58) i_PCI3600_Set1DigitalOutputOff()

Syntax:

```
_INT_ i_PCI3600_Set1DigitalOutputOff(DWORD dw_BoardHandle,
                                     BYTE b_Channel)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Number of the output channel you want to reset (0 to 7).

- Output

No output

Task:

Resets the output channel you have passed with *b_Channel*. Resetting an output channel means setting to low.

Calling convention:

```
DWORD dw_BoardHandle;
INT   i_ReturnValue;

i_ReturnValue = i_PCI3600_Set1DigitalOutputOff (dw_BoardHandle,
                                                0);
```

Return Value:

0 : No Error
1 : The handle parameter of the board is wrong
2 : Wrong channel number
3 : Digital output memory is not on

59) i_PCI3600_Set4DigitalOutputsOn()

Syntax:

```
_INT_ i_PCI3600_Set4DigitalOutputsOn(DWORD dw_BoardHandle,
                                     BYTE b_Port,
                                     BYTE b_PortValue)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Port	Number of the output port (0 or 1)
BYTE	b_PortValue	Output value (0 to 15)

- Output

No output

Task:

Sets one or several output channels of a port. Setting an output channel means setting on high.

Calling convention:

```
DWORD dw_BoardHandle;
INT   i_ReturnValue;

i_ReturnValue = i_PCI3600_Set4DigitalOutputsOn (dw_BoardHandle,
                                                 0,
                                                 15);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : The port number is wrong
-3 : The port value is wrong

60) i_PCI3600_Set4DigitalOutputsOff()

Syntax:

```
_INT_ i_PCI3600_Set4DigitalOutputsOff(DWORD dw_BoardHandle,
                                      BYTE b_Port,
                                      BYTE b_PortValue)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Port	Number of the output port (0 or 1)
BYTE	b_PortValue	Output value (0 to 15)

- Output

No output

Task:

Resets one or several output channels of one port. Resetting means setting to low.

Calling convention:

```
DWORD dw_BoardHandle;
INT       i_ReturnValue;

i_ReturnValue = i_PCI3600_Set4DigitalOutputsOff (dw_BoardHandle,
                                                  0,
                                                  15);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Digital output memory is not on
-3 : The port number is wrong
-4 : The port value is wrong

61) i_PCI3600_Set8DigitalOutputsOn()

Syntax:

```
_INT_ i_PCI3600_Set8DigitalOutputsOn(DWORD dw_BoardHandle,
                                     BYTE b_Value)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Value	Output value (0 to 255)

- Output

No output

Task:

Sets one or several output channels of board **APCI-3600**

Calling convention:

```
DWORD dw_BoardHandle;
INT   i_ReturnValue;

i_ReturnValue = i_PCI3600_Set8DigitalOutputsOn (dw_BoardHandle,
                                                255);
```

Return Value:

0 : No Error

-1 : The handle parameter of the board is wrong

62) i_PCI3600_Set8DigitalOutputsOff()

Syntax:

```
_INT_ i_PCI3600_Set8DigitalOutputsOff(DWORD dw_BoardHandle,
                                      BYTE b_Value)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Value	Output value (0 to 255)

- Output

No output

Task:

Resets one or several output channels of the board **APCI-3600**.

Calling convention:

```
DWORD dw_BoardHandle;
INT   i_ReturnValue;

i_ReturnValue = i_PCI3600_Set8DigitalOutputsOff (dw_BoardHandle,
                                                 15);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : The digital output memory is not on

9.1.7 External trigger

63) i_PCI3600_InitDigitalInputExternalTrigger()

Syntax:

```
_INT_ i_PCI3600_InitDigitalInputExternalTrigger(DWORD dw_BoardHandle,
                                                BYTE      b_TriggerLevel,
                                                WORD
w_TriggerCountValue)
```

Parameters:

- inputs:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_TriggerLevel	Define the front of the digital input whose activate the external trigger. 01 : rising edge 10 : falling edge 11 : both edge
WORD	w_TriggerCountValue	Define the number of trigger to wait that the firmware take in account the trigger.

- Output

No output

Task:

Initialises the digital input external trigger (digital input 0).

Calling convention:

```
DWORD dw_BoardHandle;
INT         i_ReturnValue;

i_ReturnValue = i_PCI3600_InitDigitalInputExternalTrigger (dw_BoardHandle,
                                                           1,
                                                           1);
```

Return Value:

0 : No Error
-1 : The handle parameter of the board is wrong
-2 : Wrong trigger level selection
-3 : Wrong trigger count value
-4 : The external trigger is already initialised by another process

9.1.8 Master trigger

64) _INT_ i_PCI3600_AnalogInput_EnableDisableMasterTrigger () :

Syntax:

```
_INT_ i_PCI3600_AnalogInput_EnableDisableMasterTrigger
                                (DWORD dw_BoardHandle,
                                 BYTE      b_Module,
                                 BYTE      b_MasterTriggerFlag)
```

Parameter:

-Input:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Module	Index of the analog input module (0 to 3)
BYTE	b_MasterTriggerFlag	Enables/disables the flag fort he master trigger 0: Disable 1: Enable

- **Output**

No output

Task:

Enables/disables the master trigger for the analog input module *b_Module*.

Calling convention:

```
DWORD dw_BoardHandle;
INT         i_ReturnValue;

i_ReturnValue = i_PCI3600_AnalogInput_EnableDisableMasterTrigger
                                                    (dw_BoardHandle,
                                                     0,
                                                     PCI3600_ENABLE);
```

Return value:

0: No error
-1: Handle parameter of the board is wrong
-2: Wrong module number
-3: External trigger flag is wrong
-4: The module is not initialised

65) _INT_ i_PCI3600_AnalogOutput_EnableDisableMasterTrigger () :

Syntax:

```
_INT_ i_PCI3600_AnalogOutput_EnableDisableMasterTrigger
                                        (DWORD    dw_BoardHandle,
                                        BYTE      b_Channel,
                                        BYTE      b_MasterTriggerFlag)
```

Parameter:

- Input:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_Channel	Index of the analog output channel (0 to 1)
BYTE	b_MasterTriggerFlag	Enables/disables the flag fort he master trigger 0: Disable 1: Enable

- Output

There is no output

Task:

Enables/disables the master trigger for the analog output channel *b_Channel*.

- Output

There is no output

Calling convention:

```
DWORD dw_BoardHandle;
INT         i_ReturnValue;

i_ReturnValue = i_PCI3600_AnalogOutput_EnableDisableMasterTrigger
                                        (dw_BoardHandle,
                                         0,
                                         PCI3600_ENABLE);
```

Return value:

0: No error
-1: Handle parameter of the board is wrong
-2: Wrong channel number
-3: External trigger flag is wrong
-4: The channel is not initialised

66) _INT_ i_PCI3600_Chronometer_EnableDisableMasterTrigger () :

Syntax:

```
_INT_ i_PCI3600_Chronometer_EnableDisableMasterTrigger
                                (DWORD dw_BoardHandle,
                                 BYTE      b_ChronometerModule,
                                 BYTE      b_MasterTriggerFlag)
```

Parameter:

-Input:

DWORD	dw_BoardHandle	Handle of the board
BYTE	b_ChronometerModule	Index of the chronometer module (0 to 3)
BYTE	b_MasterTriggerFlag	Enables/disables the flag fort he master trigger 0: Disable 1: Enable

- Output

There is no output

Task:

Enables/disables the master trigger for the chronometer module *b_ChronometerModule*.

Calling convention:

```
DWORD dw_BoardHandle;
INT         i_ReturnValue;

i_ReturnValue = i_PCI3600_Chronometer_EnableDisableMasterTrigger
                                        (dw_BoardHandle,
                                         0,
                                         PCI3600_ENABLE);
```

Return value:

0: No error
-1: Handle parameter of the board is wrong
-2: Wrong chronometer number
-3: External trigger flag is wrong
-4: The chronometer is not initialised

67) _INT_ i_PCI3600_MasterTrigger () :

Syntax:
INT i_PCI3600_MasterTrigger (DWORD dw_BoardHandle)

Parameter:

- Input:
DWORD dw_BoardHandle Handle of the board

- Output:
There is no output

Task:
Generates a master trigger. Can be used only if the board is initialised in the master mode.

Calling convention:

```
DWORD dw_BoardHandle;
INT          i_ReturnValue;

i_ReturnValue = i_PCI3600_MasterTrigger (dw_BoardHandle);
```

Return value:
0: No error
-1: Handle parameter of the board is wrong
-2: The board is not initialised as master

10 APPENDIX

10.1 Used abbreviations

Abbreviation	
ADC	A/D converter
Clk	Clock
CNT	Counter
DIFF	Differential
DMA	Direct Memory Access
DUT	Device Under Test
EEPROM	Electrically erasable PROM
FFT	Fast Fourier Transformation
FIFO	First In First Out
Gen	Generator
ICP	Integrated Circuit Piezoelectric
IRQ	Interrupt
L	Left channel (of an A/D converter)
LRCK	Sampling frequency
M1	Mode signal 1
MCLK	Clock signal
PCI	Peripheral Component Interconnect
PLD	Programmable Logic Device
R	Right channel (of an A/D converter)
RAM	Random Access Memory
RMS	Root Mean Square
RS485	Interface
SCLK	Clock Signal
SDRAM	Synchronous Dynamic Random Access Memory
SE	Single- Ended
SINAD = S/N + D	Signal-to-Noise-and-Distortion Ratio

10.2 Glossary

Term	Description
A/D converter (ADC)	= *ADC* An electronic device that produces a digital output directly proportional to an analog signal output.
ADC	= *A/D converter*
Alias frequency	A false lower frequency component that appears in analog data reconstructed from original data acquired at an insufficient sampling rate.
Aliasing	A corollary is that an A/D system of maximum sample rate F

	cannot reconstruct a signal in the frequency domain if the frequency components in the signal are higher than F/2. If the input signal frequency does exceed the F/2 Nyquist limit, the digitized signal does not contain the same frequencies as the analog signal.
Anti-aliasing filter	Analog pre-filters that avoid aliasing effects. Here before the sampling, frequencies that are higher than half the sampling frequency are filtered.
Bandwidth	The highest frequency signal component that can pass through input amplifiers and/or filters without being attenuated
Buffer	A temporary storage device used to compensate for a difference in data rate and data flow between two devices; also called a spooler.
Clock	A circuit that generates time and clock pulses for the synchronisation of the conversion
Crosstalk	An unwanted signal on a channel cause by interferences from a signal or signals on other channels.
Differential Inputs (DIFF)	An analog input with two input terminals, neither of which is grounded, whose value is the difference between the two terminals.
DMA	= *Direct Memory Access* A method by which information can be transferred from the computer memory to a device on the bus without using the processor.
DUT	= *Device Under Test* Test object (in noise measurement for example a mobile phone)
FFT	= *Fast Fourier-Transformation* Is a mathematical algorithm used to calculate the Fourier transform of a time domain signal. In other words, an FFT transforms a time domain signal to the frequency domain.
FIFO	= *First In First Out* The first data into the buffer is the first data out of the buffer.
Gain	The factor by which an incoming signal is multiplied.
Interrupt	A signal to the CPU indicating that the board detected the occurrence of a specified condition or event.
Low-pass filter	An operation that blurs details in an image (image processing) or an operation that attenuates high frequency components of an analog signal (data acquisition)
Noise	An undesirable electrical signal from an external source such as an AC power line, motors, generators, transformers, computers and other
Nyquist Sampling Theorem	If a continuous bandwidth limited signal contains no frequency components higher than a specified frequency, then the original signal can be recovered without distortion if it is sampled at a rate of at least twice the specified frequency.
On Board Memory	On high speed boards, data is acquired at much higher rates than can be written on PC memory. Data is acquired in and stored on static RAM on the A/D board until the PC can access it.
Optocoupler	A device containing light-emitting and light-sensitive

	components, used to couple isolated circuits.
Oversampling	The technique of increasing the apparent sampling frequency of a digital signal by repeating each digit a number of times, in order to facilitate the subsequent filtering of unwanted noise.
PLD	= *Programmable Logic Device*
Resolution	The smallest change that can be identified by an A/D converter or be produced by a D/A converter.
RMS	= *Root Mean Square.* The square root of the arithmetic mean of the squares of a set of values
RS485	A RS485 interface is a bus as the number of members for data transfer is limited on two.
Sampler	An electronic switch, which is switched on and switched off with high speed fort he generation of analog sampling pulses.
Sampling	Process, in which through A/D conversion values are sampled from analog audio material and saved digitally.
Sampling frequency	= *Sample rate* The rate at which a continuous-time signal is sampled.
SINAD	= *Sound in Noise and Distortion* SINAD is the ratio of the power in the fundamental bins to all other bins, including harmonics.
Single Ended Input (SE)	An analog input with one input terminal whose value is measured with respect to a common ground
SNR	= *Signal to noise ratio* Is a measure of the broad band noise that is introduced into the signal through the ADC (A/D converter), the front-end electronics and the sampling process. The magnitude of the input sinusoid is compared to the sum of all other frequencies, except for those respecting harmonics of the fundamental frequency. The ratio of the signal to the sum of the noise bins is the SNR.
Total Harmonic Distortion	= *THD* THD is measured in a similar fashion to SNR, except for the fact that in the analysis, the fundamental is compared to the size of the sum of the harmonics. If some of the harmonics lie outside the 0 to Fs/2 range, it is the aliased versions of the harmonics that are seen in the base band.
Trigger	An event that starts or stops an operation. A trigger can be specific, analog, digital or software condition.

11 INDEX

A

B

C

D

E

F