



DIN EN ISO 9001:2000
certified



ADDI-DATA GmbH
Airpark Business Center
Airport Boulevard B210
77836 Rheinmünster
Germany



Technical support:
+49 7229 1847- 0

Function description

APCI-/CPCI-1710

Multifunction counter board
- Pulse Counter -

Edition: 02.02 - 05/2008

Product information

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

Warranty and liability

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

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

Copyright

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

ADDI-DATA software product licence

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

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

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

Trademarks

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

WARNING

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



- ◆ Personal injury
- ◆ Damage to the 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.

1	DEFINITION OF APPLICATION	7
1.1	Intended use	7
1.2	Usage restrictions.....	7
1.3	Technical description	7
1.4	Function description	8
1.5	Used abbreviations	8
2	PULSE COUNTER.....	9
2.1	General description	9
2.1.1	Block diagram of the ETM function	9
2.1.2	Typical applications	10
2.2	Used signals	10
2.3	Pin assignment for all modules with pulse counter	11
2.4	Connection example	12
2.5	I/O mapping	13
2.6	Description of the I/O functions.....	13
2.6.1	WRITE register.....	13
2.6.2	LATCH Register	14
2.6.3	CTRL Register.....	14
2.6.4	SET Register	14
2.6.5	STATUS-REGISTER	15
2.6.6	Filter Register (Base + 60).....	15
2.6.7	Version Register (Base + 60)	17
2.7	Working with pulse counter function.....	18
3	STANDARDSOFTWARE	19
3.1	Define values.....	19
3.2	Interruptmask	20
3.3	Initialisation.....	21
	1) i_APCI1710_InitpulseEncoder (...)	21
	2) i_APCI1710_EnablePulseEncoder (...)	23
	3) i_APCI1710_DisablePulseEncoder (...)	25
3.3.2	Reading the pulse counter	26
	4) i_APCI1710_ReadPulseEncoderStatus (...)	26
	5) i_APCI1710_ReadPulseEncoderValue (...)	27
3.3.3	Writing into the pulse counter.....	28
	6) i_APCI1710_WritePulseEncoderValue (...)	28
3.4	Interrupt kernel routine for Windows NT/9x	29
3.4.1	Reading the pulse counter	29
	1) i_APCI1710_KRNL_ReadPulseEncoderValue (...)	29

- 3.4.2 Reading into the pulse counter 30
 - 2) i_APCI1710_KRNL_WritePulseEncoderValue (...) 30

Figures

Fig. 2-1: Block diagram of the pulse counter9
 Fig. 2-2: Pin assignment of the 50-pin SUB-D connector..... 11
 Fig. 2-3: Connection example 12
 Fig. 3-1: Timing output H; Inv_out = 0..... 22
 Fig. 3-2: Timing output H; Inv_out = 1..... 22

Tables

Table 1-1: Delivered manuals.....8
 Table 2-1: Used signals 10
 Table 2-2: I/O mapping of the pulse counter 13
 Table 3-1 Define value 19
 Table 3-2: Interruptmask of the function „pulse counter” 20

1 DEFINITION OF APPLICATION

1.1 Intended use

The board **APCI-1710** must be inserted in a PC with PCI 5V/32-bit slots, which is used as electrical equipment for measurement, control and laboratory pursuant to the norm IEC 61010-1.

The board **CPCI-1710** must be inserted in a CompactPCI system with PCI 5V/32-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 board **APCI-/CPCI-1710** must not be used as safety related part for securing emergency stop functions

The board **APCI-/CPCI-1710** must not be used in potentially explosive atmospheres.

1.3 Technical description

This manual refers to the **APCI-1710** as well as to the **CPCI-1710** board. Make sure that you have received the following items:

- The CD 1 "Standard Software Drivers" with the ADDISET parameterizing program and the required software drivers.
- The CD 2 "Technical Manuals". This CD contains the following:

- 1) The technical description **ADDICOUNT APCI-1710 / CPCI-1710: Function-programmable counter board for the PCI bus** (containing general information on the operation of the board)
- 2) A function description for each function which you want to program on the APCI-/CPCI-1710 board
- 3) The yellow leaflet "Safety precautions"

According to the used function you will find the required assignment and programming functions in the different manuals for each function:

Table 1-1: Delivered manuals

Function	PDF file (CD2 technical manuals)		Function description in SET1710	CFG file
	German	English		
Incremental counter	Inkr_zähler_d.pdf	Incr_counter_e.pdf	Incremental counter	inc_cpt.cfg
SSI	SSI_d.pdf	ssi_e.pdf	SSI	ssi.cfg
SSI monitor	SSI-Monitor_d	SSIMonitor_e.pdf	SSI_Monitor	ssi_mon.cfg
Chronos	chronos_d.pdf	chronos_e.pdf	Chronos	chronos.cfg
Counter/timer	Zähler_timer_d.pdf	Counter_timer_e.pdf	counter/timer	82x54.cfg
TOR	TOR_d.pdf	TOR_e.pdf	TOR	tor.cfg
PWM	PWM_d.pdf	PWM_e.pdf	Pulse width modulation	PWM.cfg
TTL	TTL_IO_d.pdf	TTL_IO_e.pdf	TTL I/O	ttl_io.cfg
Digital I/O	dig_EA_d.pdf	dig_IO_e.pdf	Digital I/O	dig_IO.cfg
Pulse counter	Impulszähler_d.pdf	pulseCounter_e.pdf	Pulse counter	imp_cpt.cfg
ETM (Edge time measurement)	ETM_d.pdf	ETM_e.pdf	Edge time measurement	etm.cfg

Please note:

The board **CPCI-1710** is compatible with the board **APCI-1710** as far as the installation of the software is concerned. The ADDIREG and SET1710 programs make no difference between PCI and CompactPCI boards. The API functions of the standard software are also identical.

1.4 Function description

Apart from a global description of the functions this manual contains:

- the pin assignment of the front connector
- a list of the used signals
- the I/O mapping
- a chapter about the API software functions of the standard software.

1.5 Used abbreviations

The signals on the 50 pin SUB-D connector refer always to one function module.

Please note the used abbreviations:

- UAS: Interference signal
- CLK: Clock
- REF: Reference point logic
- ENA: Enable

C1+ is a signal for **function module 1**.

2 PULSE COUNTER

2.1 General description

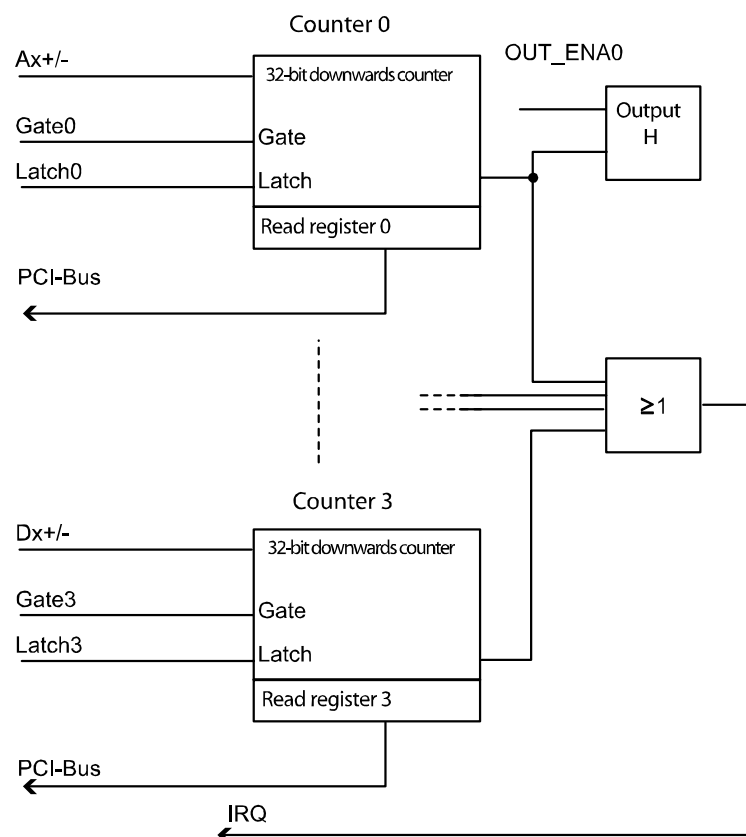
The function "pulse counter" allows the counting of single pulses or pulse sequences. Here fore the counter is programmed with a value that is defined by the user. Each pulse decrements the counter with one counting value. At the transition from 0 to -1 an interrupt can be generated. Up to 4 x 23-bit counters can be used per function module. Through parameterization you can define whether the counter stops at the transition from 0 to -1 or reloads automatically and restarts.

Properties:

- Interrupt status at the transition of the counter from 0 to -1.
- Signals up to 5 MHz (TTL, diff.) can be processed
- Start of the counter after loading a new counting word and setting of the gate bit
- At the transition of counter 0 from 1 to 0, the 24 V output H can be set (only pulse counter 0)
- Single (counter stops at the transition from 0 to -1) or continuous mode (counter loads starting value and counts newly downwards).

2.1.1 Block diagram of the ETM function

Fig. 2-1: Block diagram of the pulse counter



2.1.2 Typical applications

- Simple pulse counting
- Pulse depending trigger

2.2 Used signals

The function “pulse counter” occupies **4 inputs (for 4 counters, channels A to D) and 1 output (channel H)** of the respecting function module of the **APCI-/CPCI-1710**.

On one board you can connect max. 16 pulse counters.

Table 2-1: Used signals

SIGNALS	AT THE CONNECTOR	POLARITY	FUNCTION
Counter 0	Ax +/-	Diff./TTL/optional 24 V	Input of counter 0
Counter 1	Bx +/-	Diff./TTL/optional 24 V	Input of counter 1
Counter 2	Cx +/-	Diff./TTL/optional 24 V	Input of counter 2
Counter 3	Dx +/-	Diff./TTL/optional 24 V	Input of counter 3
Trigger output	Hx	24V / optional 5V	Digital output of counter 0

x: Number of the function module.

2.3 Pin assignment for all modules with pulse counter



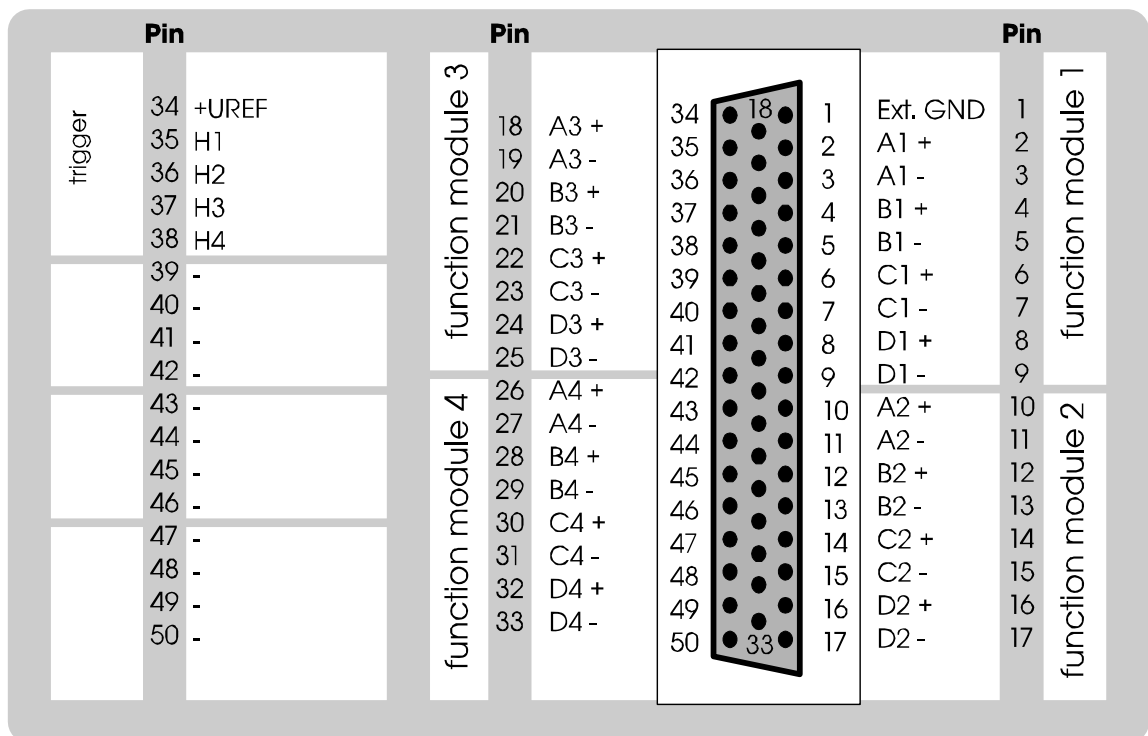
IMPORTANT!

The function modules are defined differently in the hardware and software descriptions.

For the pin assignment (hardware) the modules from 1 to 4 are numbered. For the SET1710 program or the software functions (software) the module numbering **BEGINS** with 0.

The figure below is a connection example. The function “pulse counter” is implemented on all function modules.

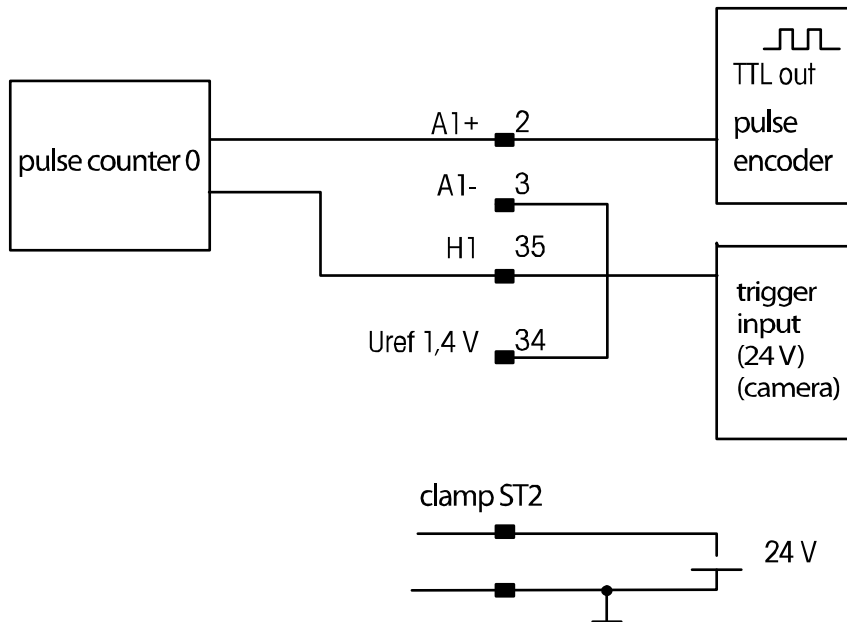
Fig. 2-2: Pin assignment of the 50-pin SUB-D connector



-: not connected

2.4 Connection example

Fig. 2-3: Connection example



2.5 I/O mapping

Table 2-2: I/O mapping of the pulse counter

	IORD		IOWR	
	D31.....D8	D7.....D0	D31.....D8	D7.....D0
BASEx + 0	LATCH Register 0		WRITE Register 0	
BASEx + 4	LATCH Register 1		WRITE Register 1	
BASEx + 8	LATCH Register 2		WRITE Register 2	
BASEx + 12	LATCH Register 3		WRITE Register 3	
BASEx + 16	-	STATUS Register	-	CTRL Register
BASEx + 20	-		SET Register	
.....	-		-	
BASEx + 60	FILTER Register		Filter Register	
	VERSION Register			

-: No function; y: Data of no importance, x: Number of the function module

The accesses are always read or written in 32-bit.

2.6 Description of the I/O functions

This function contains 4 independent 32-bit counters. Each counter is allocated to an external line. The counter is decremented through this line when the signal indicates an edge change from 0 to 1 or from 1 to 0.

The counter can be initialised on a fix value through software, from which it counts downwards. At the transition from value 0 to -1 an interrupt can be generated and the output can be set. At this event the respecting counter is stopped. The counter is enabled again when it is loaded with a new value and the respecting GATE bit is set on "1".

2.6.1 WRITE register

By writing on this register the counters are loaded and enabled. The registers are 32-bit and cannot be reread. After reset they are to zero.

- Base + 0 → Counter 0
- Base + 4 → Counter 1
- Base + 8 → Counter 2
- Base + 12 → Counter 3

2.6.2 LATCH Register

With reading on these addresses the 32-bit values of the counters are read out.

- Base + 0 → Counter 0
- Base + 4 → Counter 1
- Base + 8 → Counter 2
- Base + 12 → Counter 3

2.6.3 CTRL Register

On base address + 16 is the "Control Register". It is written in 32-bit, but is not rereadable. After the reset all bits are set to zero.

Bits	D31.. D8	D7	D6	D5	D4	D3	D2	D1	D0
Description	-	CYCL3	CYCL2	CYCL1	CYCL0	GATE3	GATE2	GATE1	GATE0

Per counter one bit is available:

GATE_y = "0" the counter is locked
 "1" the counter is released

CYCL_y = "0" single mode
 "1" continuous mode

y= counter number (0 to 3)

2.6.4 SET Register

On the base address + 20 is the "SET Register"; it is written in 32-bit – not rereadable.

After reset all bits are set to zero. No interrupt can be generated. The digital inputs are not inverted. The counter is decremented with the rising edge. The digital output is switched off.

Bits	D31..D5	D12	D11	D10	D9	D8	D7...D5	D4	D3	D2	D1	D0
Description	-	INV_ OUT	INV_ IN3	INV_ IN2	INV_ IN1	INV_ IN0	-	OUT_ ENA0	INT_ ENA3	INT_ ENA2	INT_ ENA1	INT_ ENA0

For each counter are two bits available in order to start an activity at zero cross

INT_ENA_y = "0" no interrupt at zero cross (status after reset)
 "1" interrupt at zero cross

OUT_ENA0 = "0" the output of counter 0 will not be set (status after reset)
 "1" the output will be set at zero cross of counter 0

INV_IN_y "0" the input of counter y will not be inverted.
 The counter will be decremented with rising edge (status after reset)
 "1" the input will be inverted. The counter with the falling edge will be decremented.

INV_OUT "0" the output will be not inverted (status after reset)
 At the zero cross of counter 0, the output will be switched

from 1 to 0.
 "1" At the zero cross the output will be switched from 0 to 1.

The zero cross of counter 0 only affects output H when the OUT_ENA0 bit is set.

y= counter number (0 to 3)

The interrupt request is a common interrupt of the 4 counters ("ODER" connection).

2.6.5 STATUS-REGISTER

Reading the base address + 16 gives information about the current counter status. Indicated is the counter with one zero cross.

Bits	D31..D5	D4	D3	D2	D1	D0
Description	-	CB3	CB2	CB1	CB0	G_CB

- G_CB : "0" Global zero cross bit, no zero cross on one of the 4 counters
 "1" zero cross
- CB 0 : "0" no zero cross on counter 0
 "1" zero cross on counter 0
- CB 1 : "0" no zero cross on counter 1
 "1" zero cross on counter 1
- CB 2 : "0" no zero cross on counter 2
 "1" zero cross on counter 2
- CB 3 : "0" no zero cross on counter 3
 "1" zero cross on counter 3

After loading the counter/s with a new value, the respecting bits are reset. This information reflects the interrupt request and of the digital output.

2.6.6 Filter Register (Base +60)

By a simple reading access to Register 60 the Version Register described in chapter 2.6.7 can be read. By a writing access to Register 60, a digital filter can be parameterized for the inputs A, B, C, D. The set filter values refer to positive and negative pulses. All pulses that are smaller than the set time are filtered. For reading back this range, the Bit DQ15 must be set. Then the filter status can be read through reading access. to the register

Through the filter value „0000“, the filter is disabled. Bit DQ8 indicates if a 40 MHz quartz is available on the used board. Through DQ9, the filter clock to used can be set.

- DQ3..0 : Filter values for the inputs A, B, C, D base 40 MHz
- 0 0000 = Filter disabled
- 1 0001 = Filter of 100 ns
- 2 0010 = Filter of 150 ns
- 3 0011 = Filter of 200 ns

4	0100	=	Filter of 250 ns
5	0101	=	Filter of 300 ns
6	0110	=	Filter of 350 ns
7	0111	=	Filter of 400 ns
8	1000	=	Filter of 450 ns
9	1001	=	Filter of 500 ns
10	1010	=	Filter of 550 ns
11	1011	=	Filter of 600 ns
12	1100	=	Filter of 650 ns
13	1101	=	Filter of 700 ns
14	1110	=	Filter of 750 ns
15	1111	=	Filter of 800 ns

DQ3..0 : Filter values for the inputs A, B, C, D base 33 MHz

0	0000	=	Filter disabled
1	0001	=	Filter of 121 ns
2	0010	=	Filter of 182 ns
3	0011	=	Filter of 242 ns
4	0100	=	Filter of 303 ns
5	0101	=	Filter of 364 ns
6	0110	=	Filter of 424 ns
7	0111	=	Filter of 485 ns
8	1000	=	Filter of 545 ns
9	1001	=	Filter of 606 ns
10	1010	=	Filter of 667 ns
11	1011	=	Filter of 727 ns
12	1100	=	Filter of 788 ns
13	1101	=	Filter of 848 ns
14	1110	=	Filter of 909 ns
15	1111	=	Filter of 970 ns

DQ3..0 : Filter values for the inputs A, B, C, D base 30 MHz

0	0000	=	Filter disabled
1	0001	=	Filter of 133 ns
2	0010	=	Filter of 200 ns
3	0011	=	Filter of 267 ns
4	0100	=	Filter of 333 ns
5	0101	=	Filter of 400 ns
6	0110	=	Filter of 467 ns
7	0111	=	Filter of 533 ns
8	1000	=	Filter of 600 ns
9	1001	=	Filter of 667 ns
10	1010	=	Filter of 733 ns
11	1011	=	Filter of 800 ns
12	1100	=	Filter of 867 ns
13	1101	=	Filter of 933 ns
14	1110	=	Filter of 1000 ns
15	1111	=	Filter of 1067 ns

DQ8 : 40 MHz status (can only be read)

- 1: 40 MHz available
- 0: 40 MHz not available

DQ9 : Filter clock

- 1: 40 MHz
- 0: 33 MHz (or 30 MHz with former mainboards)

DQ15 : Enable to read the filter status

- 1: next RD access to Register 60 reads the filter status
- 0: next RD access to Register 60 reads the Version Register

2.6.7 Version Register (Base + 60)

The function and revision are recognised (reading command, ASCII format)

Bits	D31... D17	D16...D0
Description	Function	Revision

Example:

BASE + 60 "I" "Z" "1" "1"

Meaning: Pulse counter 1.1

2.7 Working with pulse counter function

1. Loading the counting value
2. Defining the input signal of the counter (rising or falling edge)
3. Defining the action of the counter at zero cross (activating the output or generating an interrupt).
4. Releasing the counter over the gate bit.
5. Selecting the counter mode (continuous or single)

3 STANDARDSOFTWARE

3.1 Define values



IMPORTANT!

Note the following style conventions in the text:

Function: *"i_APCI1710_SetBoardInformation"*

Variable *ui_Address*

Table 3-1 Define value

Define name	Decimal value	Hexadecimal value
DLL_COMPILER_C	0	0
DLL_COMPILER_VB	1	1
DLL_COMPILER_PASCAL	2	2
DLL_LABVIEW	3	3
DLL_COMPILER_VB_5	4	4
APCI1710_DISABLE	0	0
APCI1710_ENABLE	1	1

3.2 Interruptmask

Each pulse counter can generate an interrupt. In order to obtain this interrupt, you shall enable the interrupt and the interrupt routine with the function "i_APCI1710_SetBoardIntRoutineX"

Table 3-2: Interruptmask of the function „pulse counter“

b_ModuleMask	ul_InterruptMask	Meaning
0000 0001	0000 0001 0000 0000	Zero cross of pulse counter 0 – module 0
0000 0001	0000 0010 0000 0000	Zero cross of pulse counter 1 - module 0
0000 0001	0000 0100 0000 0000	Zero cross of pulse counter 2 - module 0
0000 0001	0000 1000 0000 0000	Zero cross of pulse counter 3 - module 0
0000 0010	0000 0001 0000 0000	Zero cross of pulse counter 0 - module 1
0000 0010	0000 0010 0000 0000	Zero cross of pulse counter 1 - module 1
0000 0010	0000 0100 0000 0000	Zero cross of pulse counter 2 - module 1
0000 0010	0000 1000 0000 0000	Zero cross of pulse counter 3 - module 1
0000 0100	0000 0001 0000 0000	Zero cross of pulse counter 0 - module 2
0000 0100	0000 0010 0000 0000	Zero cross of pulse counter 1 - module 2
0000 0100	0000 0100 0000 0000	Zero cross of pulse counter 2 - module 2
0000 0100	0000 1000 0000 0000	Zero cross of pulse counter 3 - module 2
0000 1000	0000 0001 0000 0000	Zero cross of pulse counter 0 -module 3
0000 1000	0000 0010 0000 0000	Zero cross of pulse counter 1 - module 3
0000 1000	0000 0100 0000 0000	Zero cross of pulse counter 2 - module 3
0000 1000	0000 1000 0000 0000	Zero cross of pulse counter 3 - module 3

3.3 Initialisation

1) i_APCI1710_InitpulseEncoder (...)

Syntax:

```
<Return Wert> = i_APCI1710_InitpulseEncoder
                (BYTE      b_BoardHandle,
                 BYTE      b_ModulNbr,
                 BYTE      b_PulseEncoderNbr,
                 BYTE      b_InputLevelSelection,
                 BYTE      b_TriggerOutputAction,
                 ULONG     ul_StartValue)
```

Parameter:

- Eingabe:

BYTE	b_BoardHandle	Handle of the xPCI-1710 board
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_PulseEncoderNbr	Selection of the pulse counter (0 to 3)
BYTE	b_InputLevelSelection	Selection of the pulse level (0 or 1) 0: Pulse counter counts the pulse at Low. 1: Pulse counter counts the pulse at High
BYTE	b_TriggerOutputAction	Action of the digital trigger output. Only for pulse counter 0 0: Not enabled 1: Sets the trigger output to "1" (high) after the pulse counter have decremented from 1 to 0. See Fig. 3-1 2: Sets the trigger output to "0" (low) after the pulse counter has decremented from 1 to 0. See Fig. 3-1.
ULONG	ul_StartValue	Start value of the pulse counter (1 to 4294967295)

-Output:

There is no output.

Task:

Configures the operating mode of the pulse counter through *b_ModulNbr* and *b_PulseEncoderNbr*. The pulse counter decrements the counter with 1 value after each pulse. Call this function before calling other functions that access the pulse counter.

Fig. 3-1: Timing output H; Inv_out = 0

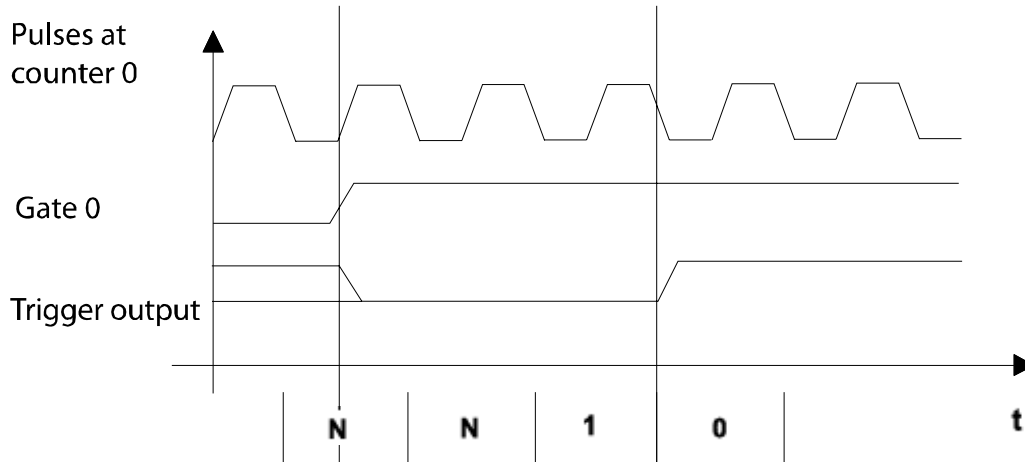
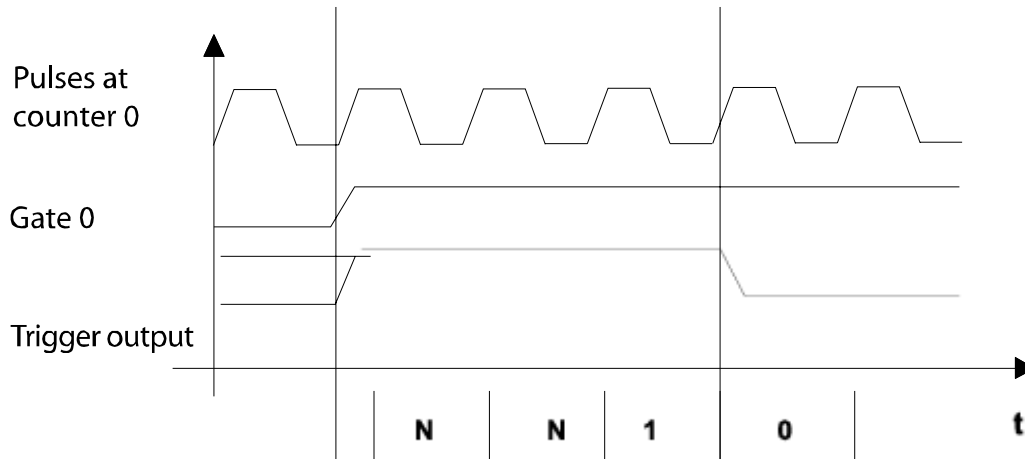


Fig. 3-2: Timing output H; Inv_out = 1



Calling convention:

ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
```

```
i_ReturnValue = i_APCI1710_InitPulseEncoder
                (b_BoardHandle,
                 0,
                 0,
                 1,
                 0);
```

Return value:

- 0: No error
- 1: Handle parameter of the board is wrong
- 2: The selected module is no "pulse counter" module.
- 3: The selected pulse counter is wrong.
- 4: Selection of the input level is wrong
- 5: The action selection of the digital trigger is wrong.
- 6: The start value of the pulse counter is wrong.

2) i_APCI1710_EnablePulseEncoder (...)

Syntax:

```
<Return Wert> = i_APCI1710_EnablePulseEncoder
                                     (BYTE b_BoardHandle,
                                     BYTE b_ModulNbr,
                                     BYTE b_PulseEncoderNbr,
                                     BYTE b_CycleSelection,
                                     BYTE b_InterruptHandling)
```

Parameter:

- Input

BYTE	b_BoardHandle	Handle of the xPCI-1710 board
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_PulseEncoderNbr	Selection of the pulse counter (0 to 3)
BYTE	b_CycleSelection	APCI1710_CONTINUOUS: At the transition of the counter value to "0", the pulse counter loads the start value after the next pulse automatically new. APCI1710_SINGLE: At the transition of the counter to "0", the pulse counter is stopped.
BYTE	b_InterruptHandling	Interrupts can be generated when the pulse counter run out. The user decides if he uses an interrupt or not. APCI1710_ENABLE: Interrupts enabled APCI1710_DISABLE: Interrupts disabled

- Output

There is no output.

Task:

Enables the selected pulse counter (*b_PulseEncoderNbr*) of the indicated module (*b_ModulNbr*). At each input pulse the counting value of the pulse counter is decremented with 1. If the interrupt function is released (*b_InterruptHandling*), an interrupt will be generated as soon as the pulse counter has run out.

Calling convention:

ANSI C:

```
int          i_ReturnValue;
```

```
i_ReturnValue = i_APCI1710_EnablePulseEncoder
               (b_BoardHandle,
               0,
               0,
               APCI1710_CONTINUOUS,
               APCI1710_DISABLE);
```

Return value:

0: No error

-1: Handle parameter of the board is wrong.

-2: Selected module number is wrong.

-3: Selected pulse counter is wrong.

-4: Pulse counter not initialised. See function "i_APCI1710_InitPulseEncoder"

-5: Selected cycle mode is wrong

-6: Mode of the interrupt management is wrong.

-7: Interruptroutine not installed.

See function "i_APCI1710_SetBoardIntRoutineX"

3) i_APCI1710_DisablePulseEncoder (...)

Syntax:

```
<Return Wert> = i_APCI1710_DisablePulseEncoder
                    (BYTE b_BoardHandle,
                     BYTE b_ModulNbr,
                     BYTE b_PulseEncoderNbr)
```

Parameter:

- Input

BYTE	b_BoardHandle	Handle of the xPCI-1710 board
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_PulseEncoderNbr	Selection of the pulse counter (0 to 3)

- Output

There is no output.

Task:

Disables the selected pulse counter (*b_PulseEncoderNbr*) of the indicated module (*b_ModulNbr*).

Calling convention:

ANSI C:

```
int i_ReturnValue;
```

```
i_ReturnValue = i_APCI1710_DisablePulseEncoder
                    (b_BoardHandle,
                     0,
                     0);
```

Return value

- 0: No error
- 1: Handle parameter of the board is wrong.
- 2: Selected module number is wrong.
- 3: Selected pulse counter is wrong.
- 4: Pulse counter not initialised. See function "i_APCI1710_InitPulseEncoder"

3.3.2 Reading the pulse counter

4) i_APCI1710_ReadPulseEncoderStatus (...)

Syntax:

```
<Return Wert> = i_APCI1710_ReadPulseEncoderStatus
                    (BYTE    b_BoardHandle,
                     BYTE    b_ModulNbr,
                     BYTE    b_PulseEncoderNbr,
                     PBYTE   pb_Status)
```

Parameter:

- Input

BYTE	b_BoardHandle	Handle of the xPCI-1710 board
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_PulseEncoderNbr	Selection of the pulse counter (0 to 3)

- Output

PBYTE	pb_Status	Status of the pulse counter 0 : No zero cross occurred. 1 : Zero cross occurred.
-------	-----------	--

Task:

Reads the status of the selected pulse counter (*b_PulseEncoderNbr*) of the selected module (*b_ModulNbr*).

Calling convention:

ANSI C :

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned char b_Status;
```

```
i_ReturnValue = i_APCI1710_ReadPulseEncoderStatus
                (b_BoardHandle, 0, 0,
                 &b_Status);
```

Return value:

- 0: No error
- 1: Handle parameter of the board is wrong.
- 2: Selected module number is wrong.
- 3: Selected pulse counter is wrong
- 4: Pulse counter not initialised. See function "i_APCI1710_InitPulseEncoder"

5) i_APCI1710_ReadPulseEncoderValue (...)

Syntax:

```
<Return Wert> = i_APCI1710_ReadPulseEncoderValue
                                     (BYTE      b_BoardHandle,
                                     BYTE      b_ModulNbr,
                                     BYTE      b_PulseEncoderNbr,
                                     PULONG    pul_ReadValue)
```

Parameter:

- Input

BYTE	b_BoardHandle	Handle of the xPCI-1710 board
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_PulseEncoderNbr	Selection of the pulse counter (0 to 3)

- Output

PULONG	pul_ReadValue	Pulse encoder value
--------	---------------	---------------------

Task:

Reads the pulse counter value from the selected pulse counter (*b_PulseEncoderNbr*) from the selected module (*b_ModulNbr*).

Calling convention:

ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned long ul_ReadValue;
```

```
i_ReturnValue = i_APCI1710_ReadPulseEncoderValue
               (b_BoardHandle,
                0,
                0,
                &ul_ReadValue);
```

Return value:

- 0: No error
- 1: Handle parameter of the board is wrong.
- 2: Selected module number is wrong.
- 3: Selected pulse counter is wrong
- 4: Pulse counter not initialised. See function "i_APCI1710_InitPulseEncoder"

3.3.3 Writing into the pulse counter

6) i_APCI1710_WritePulseEncoderValue (...)

Syntax:

```
< Return Wert > = i_APCI1710_WritePulseEncoderValue
                                (BYTE      b_BoardHandle
                                BYTE      b_ModulNbr,
                                BYTE      b_PulseEncoderNbr,
                                ULONG     ul_WriteValue)
```

Parameter:

- Input

BYTE	b_BoardHandle	Handle of the xPCI-1710 board
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_PulseEncoderNbr	Selection of the pulse counter (0 to 3)
ULONG	ul_WriteValue	Write the 32-bit value

- Output

There is no output.

Task:

Writes the 32-bit value (*ul_WriteValue*) into the selected pulse counter (*b_PulseEncoderNbr*) of the indicated module (*b_ModulNbr*). This process sets the pulse counter value new.

Calling convention:

ANSI C:

```
int      i_ReturnValue;
unsigned char b_BoardHandle;
i_ReturnValue = i_APCI1710_WritePulseEncoderValue
                (b_BoardHandle,
                0, 0, 2000);
```

Return value:

0: No error
 -1: Handle parameter of the board is wrong.
 -2: Selected module number is wrong.
 -3: Selected pulse counter is wrong.
 -4: Pulse counter not initialised. See function "i_APCI1710_InitPulseEncoder"

3.4 Interrupt kernel routine for Windows NT/9x

i

IMPORTANT!

These functions are only available for the user interrupt routine under Windows NT and Windows 95/98 in the synchronous mode. See function "i_APCI1710_SetBoardIntRoutineWin32"

3.4.1 Reading the pulse counter

1) i_APCI1710_KRNL_ReadPulseEncoderValue (...)

Syntax:

```
<Return Wert> = i_APCI1710_KRNL_ReadPulseEncoderValue
                (UINT          ui_BaseAddress,
                 BYTE          b_ModulNbr,
                 BYTE          b_PulseEncoderNbr,
                 PULONG       pul_ReadValue)
```

Parameter:

- Input

UINT	ui_BaseAddress	Base address of the xPCI-1710 . See "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_PulseEncoderNbr	Selection of the pulse counter (0 to 3)

- Output

PULONG	pul_ReadValue	Value of the pulse counter
--------	---------------	----------------------------

Task:

Reads the counter value of the selected pulse counter (*b_PulseEncoderNbr*) of the indicated module (*b_ModulNbr*).

Calling convention:

ANSI C :

```
int          i_ReturnValue;
unsigned int ui_BaseAddress;
unsigned long ul_ReadValue;
```

```
i_ReturnValue = i_APCI1710_KRNL_ReadPulseEncoderValue
                (ui_BaseAddress,
                 0, 0,
                 &ul_ReadValue);
```

Return value:

0: No error
-1: Selected module number is wrong
-2: Selected module is no "pulse counter" module
-3: Selected pulse counter is wrong.

3.4.2 Reading into the pulse counter

2) `i_APCI1710_KRNL_WritePulseEncoderValue (...)`

Syntax:

```
< Return Wert > = i_APCI1710_KRNL_WritePulseEncoderValue
                    (UINT    ui_BaseAddress,
                     BYTE    b_ModulNbr,
                     BYTE    b_PulseEncoderNbr,
                     ULONG   ul_WriteValue)
```

Parameter:

- Input

UINT	ui_BaseAddress	Base address of the xPCI-1710 . See "i_APCI1710_GetHardwareInformation"
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_PulseEncoderNbr	Selection of the pulse counter (0 to 3)
ULONG	ul_WriteValue	32-bit value to write

- Output

There is no output.

Task:

Writes the 32-bit value (*ul_WriteValue*) into the selected pulse counter (*b_PulseEncoderNbr*) of the indicated module (*b_ModulNbr*). This process sets the pulse counter value new.

Calling convention:

ANSI C :

```
int          i_ReturnValue;
unsigned int ui_BaseAddress;
```

```
i_ReturnValue = i_APCI1710_KRNL_WritePulseEncoderValue
                (ui_BaseAddress,
                 0, 0, 2000);
```

Return value:

0: No error
 -1: Selected module number is wrong.
 -2: The selected module is no "pulse counter" module.
 -3: The selected pulse counter is wrong.