



DIN EN ISO 9001:2000
certified



ADDI-DATA GmbH
Dieselstraße 3
D-77833 OTTERSWEIER



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

Function description

ADDICOUNT APCI-/CPCI-1710

TOR

3rd edition 03/2005

Product information

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

Warranty and liability

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

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

Copyright

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

ADDI-DATA software product licence

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

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

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

Trademarks

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

WARNING

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



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

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

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

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

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

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

◆ **Used symbols:**



IMPORTANT!

designates hints and other useful information.



WARNING!

It designates a possibly dangerous situation.

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

1	DEFINITION OF APPLICATION	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	TOR	9
2.1	Function description	9
2.1.1	Block diagram	9
2.1.2	Typical applications	9
2.2	Used signals	10
2.3	Pin assignment of all modules with TOR function	11
2.4	Connection example	12
2.5	I/O mapping	12
2.5.1	Write register.....	12
2.5.2	Read register.....	13
2.6	Description of the I/O functions.....	13
2.6.1	Function description	13
2.6.2	Timer1 REGISTER	14
2.6.3	Timer0 REGISTER	14
2.6.4	TOR COMMANDO Register	14
2.6.5	TOR Gate Register	14
2.6.6	TOR Synchronisation GATE Register	15
2.6.7	TOR Status Register.....	15
2.6.8	TOR Interrupt Status Register	16
2.6.9	Version Register (Base + 60)	16
2.7	Working with the TOR function	16
3	SOFTWARE FUNCTIONS	17
3.1	Introduction.....	17
3.2	Interrupt mask	17
3.3	Initialisation.....	19
	1) i_APCI1710_InitTorCounter (...)	19
	2) i_APCI1710_EnableTorCounter (...)	22
	3) i_APCI1710_DisableTorCounter (...)	24
	4) i_APCI1710_GetTorCounterInitialisation (...).....	25
3.4	Read TOR counter	27
	1) i_APCI1710_GetTorCounterProgressStatus (...)	27

	2) i_APCI1710_ReadTorCounterValue (...)	29
3.5	Interrupt kernel functions for Windows NT/95.....	31
	1) i_APCI1710_KRNL_GetTorCounterProgressStatus (...).....	31
	2) i_APCI1710_KRNL_ReadTorCounterValue (...).....	33

Figures

Fig. 2-1: Block diagram of the TOR function	9
Fig. 2-2: Pin assignment of the 50-pin SUB-D connector.....	11
Fig. 2-3: Connection example	12

Tables

Table 1-1: Delivered manuals.....	8
Table 2-1: Used signals	10
Table 2-2: I/O mapping (write register).....	12
Table 3-1: Define Value	17
Table 3-2: Interrupt mask of the function TOR	17
Table 3-3: Return table for counter value	18
Table 3-4: Base time value	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 **APCI-/CPCI-1710** board must not be used as safety related part for securing emergency stop functions.

The **APCI-/CPCI-1710** board 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 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-1710/1711** is compatible with the board **APCI-1710** regarding the software installation. The programs ADDIREG and SET1710 do not make a difference between PCI boards and CompactPCI boards.

The API functions of the standard software are also identical.

1.4 Function description

Besides a global description of the software functions this manual contains:

- The pin assignment of the front connector
- A list of the signals used
- The I/O mapping
- A chapter on the API software functions of the supplied standard software.

1.5 Used abbreviations

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

Please observe the following abbreviations:

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

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

2 TOR

2.1 Function description

The function "TOR" is a counter interface which allows counting input signal within a defined time.

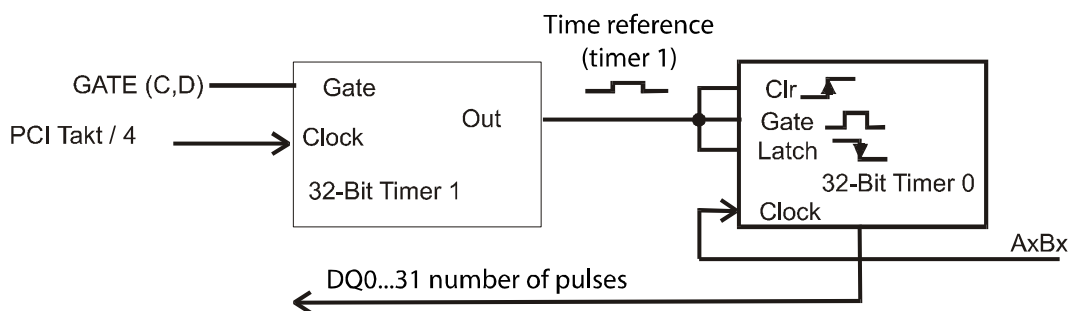
On one module 2 TOR counters are available. Each TOR counter contains 2 x 32-bit timer.

Properties:

- Isolation through optical couplers for the input and output channels to avoid earth circuit
- Interrupt status at the end of the measuring period
- Inputs and outputs can be inverted by software.
- Software Gate

2.1.1 Block diagram

Fig. 2-1: Block diagram of the TOR function



2.1.2 Typical applications

- Frequency measurement
- Pulse counting per time interval

2.2 Used signals

The function "TOR" occupies **4 inputs (A to D)** of the respecting function module of the **APCI-/CPCI-1710**.

Table 2-1: Used signals

AT THE CONNECTOR	POLARITY	FUNCTION
A x +/-	Diff. / TTL	Digital input 1 (TOR 1)
B x +/-	Diff. / TTL	Digital input 2, (TOR2)
C x +/-	Diff. / TTL / Opt. 24V	External gate (TOR1)
D x +/-	Diff. / TTL / Opt. 24V	External gate (TOR2)

x: Number of the function module.

2.3 Pin assignment of all modules with TOR function

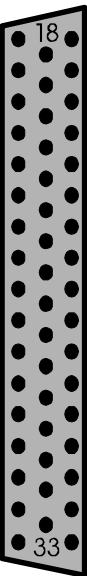
i

IMPORTANT!

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

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

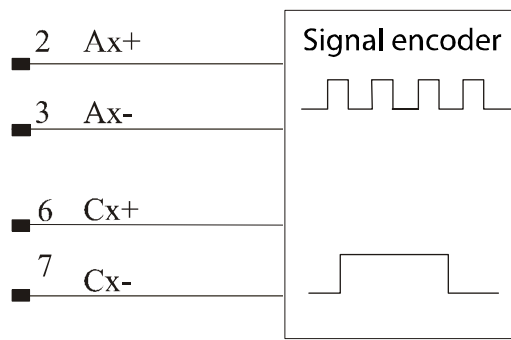
Fig. 2-2: Pin assignment of the 50-pin SUB-D connector

Pin		Pin		Pin		Pin		
Digital output 1	34 -	Function module 3	18 A3 +		1 Ext. GND	1	Function module 1	
	35 -		19 A3 -		2 A1 +	2		
	36 -		20 B3 +		3 A1 -	3		
	37 -		21 B3 -		4 B1 +	4		
	38 -		22 C3 +		5 B1 -	5		
Digital input 1	39 -	Function module 4	23 C3 -		6 C1 +	6	Function module 2	
	40 -		24 D3 +		7 C1 -	7		
	41 -		25 D3 -		8 D1 +	8		
	42 -		26 A4 +		9 D1 -	9		
	43 -		27 A4 -		10 A2 +	10		
Digital input 2	44 -	Function module 3	28 B4 +		11 A2 -	11	Function module 1	
	45 -		29 B4 -		12 B2 +	12		
	46 -		30 C4 +		13 B2 -	13		
	47 -		31 C4 -		14 C2 +	14		
	48 -		32 D4 +		15 C2 -	15		
Digital input 3	49 -		33 D4 -		16 D2 +	16		Function module 2
	50 -				17 D2 -	17		

- Not connected

2.4 Connection example

Fig. 2-3: Connection example



2.5 I/O mapping

2.5.1 Write register

Table 2-2: I/O mapping (write register)

		D31...D24	D23...D16	D15.....D8	D7.....D0
BYTES	Wr	HIGHBYTE	MIDHIGHBYTE	MIDLOWBYTE	LOWBYTE
BASE _x + 0	☑	TOR1 BASE_TIME			
BASE _x + 4	☑	-	-	-	TOR1 COMMANDO
BASE _x + 8	☑	-	-	-	TOR1 GATE
BASE _x + 12	☑	-	-	-	TOR1/TOR2 SYNCHRO_GATE
BASE _x + 16		TOR2 BASE_TIME			
BASE _x + 20	☑	-	-	-	TOR2 COMMANDO
BASE _x + 24	☑	-	-	-	TOR2 GATE
BASE _x + 28	☑	-	-	-	TOR1/TOR2 SYNCHRO_GATE
BASE _x + 60		-	-	-	-

-: No function; **x**: Number of the function module.

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

2.5.2 Read register

		D31...D24	D23...D16	D15.....D8	D7.....D0
BYTES	Rd	HIGHBYTE	MIDHIGHBYTE	MIDLOWBYTE	LOWBYTE
BASE _x + 0	<input checked="" type="checkbox"/>	TOR1 COUNTER_VALUE			
BASE _x + 4	<input checked="" type="checkbox"/>	-	-	-	TOR1 STATUS
BASE _x + 8	<input checked="" type="checkbox"/>	-	-	-	TOR1 GATE
BASE _x + 12	<input checked="" type="checkbox"/>	-	-	-	TOR1 INT_STATUS
BASE _x + 16		TOR2 COUNTER_VALUE			
BASE _x + 20	<input checked="" type="checkbox"/>	-	-	-	TOR2 STATUS
BASE _x + 24	<input checked="" type="checkbox"/>	-	-	-	TOR2 GATE
BASE _x + 28	<input checked="" type="checkbox"/>	-	-	-	TOR2 INT_STATUS
BASE _x + 60	<input checked="" type="checkbox"/>	FUNKNBR2	FUNKNBR1	REVBYTE2	REVBYTE1

2.6 Description of the I/O functions

2.6.1 Function description

The function "TOR" is a scaled-down version of the module function "Timer/Counter". The pulse signals from Timer 1 indicates the start and stop pulse signal for Timer 0. Timer 0 counts the input signals. After the stop signal from Timer 0 the number of pulses is stored and can be read through I/O read commands.

The timer 1 is used as time reference generator.

The division factor is set in timer 1 and determines the output frequency. The input frequency is set according to the PCI clock pulse. Timer 0 is synchronised with the start event.

Pulse measurement

As soon as a **start** signal occurs from Timer 1, Timer 0 is reset and counts the pulse signals of signals from channel A_x (B_x).

During the process the status bit "Counter in Progress" is set in the status register.

As soon as a **stop** signal is generated from Timer 1, the Timer 0 is stopped and the status bit "Counter in Progress" is reset.

An interrupt can also be generated. The value can be read. The latest measured value is read in the "counter measurement" register.

2.6.2 Timer1 REGISTER

Base address + 0 for Tor1 and base address + 16 for Tor2

32-bit register: The "Reload" value for TIMER 1 is written. The output frequency from Timer 1 is determined by the division factor.

2.6.3 Timer0 REGISTER

Base address + 0 for Tor1 and base address + 16 for Tor2

32-bit register for reading the current value of the pulse measurement (i.e. latest measurement value)

2.6.4 TOR COMMANDO Register

Base address + 4: TOR1

Base address + 20: TOR2

DQ0	0	Simple mode
	1	Continuous mode
DQ1	0	Interrupt generation disabled
	1	Interrupt generation enabled
DQ2:	0	Counting to "High" level
	1	Inverted image of the digital input, counting to „Low" level
DQ3	0	External input gate not used
	1	External gate starts the TOR counter

2.6.5 TOR Gate Register

Base address + 8: TOR1

Base address + 24: TOR2.

Writing:

DQ0	0	Disables the counter
	1	Enables the counter
DQ4	0	No write error possible
	1	Resets the initialisation of the TOR counter

Reading:

DQ0	0	Counter disabled
	1	Counter enabled
DQ4	0	TOR counter not initialised.
	1	TOR counter initialised

2.6.6 TOR Synchronisation GATE Register**Base address +12:** TOR1**Base address +28:** TOR2.

DQ0	0	Disables counter from Tor1 and Tor2
	1	TOR counter not initialised from TOR1 and TOR2
DQ4	0	No commands possible
	1	Resets the initialisation from TOR1 and TOR2

2.6.7 TOR Status Register**Base address + 4:** TOR1**Base address + 20:** TOR2.

32-bit register: Returns the status information from "TOR".

DQ0	0	Measurement is not running
	1	Measurement is running. Reset at reading the address Base+0 for TOR1 and the address Base+16 for TOR2
DQ1	0	Measurement not ended.
	1	Measurement ended. Reset at reading the address Base+0 for TOR1 and the address Base+16 for TOR2
DQ2	0	Measurement time not over.
	1	Measurement time is over. Reset at reading the address Base+0 for TOR1 and the address Base+16 for TOR2
DQ4	0	Simple mode selected
	1	Continuous mode selected
DQ5	0	Interrupt mode disabled
	1	Interrupt mode enabled
DQ6	0	Input signal not inverted

	1	Input signal not inverted
DQ7	0	Gate input signal not used
	1	Gate input signal used
Other bits		Set on 0 at reading

2.6.8 TOR Interrupt Status Register

Base address + 12: TOR1

Base address + 28: TOR2.

32-bit register: Returns the interrupt status information of the TR function.

DQ0	=1	Interrupt is generated at the end of measurement. The interrupt request is reset after reading the register.
DQ31..1		Is always set on 0.

2.6.9 Version Register (Base + 60)

The function and the revision are identified (read command, ASCII format)

BASE + 60 "T" "O" "1" "3"

Meaning: TOR Revision 1.3

2.7 Working with the TOR function

1. Signal connection at the connector
2. Initialising the TOR function (Selection of the TOR counter, input frequency etc.)
3. Releasing the measurement (single or continuous measurement with/without interrupt)
4. Reading and evaluating the status of the TOR counter
5. Reading the measurement value of the TOR counter.

3 SOFTWARE FUNCTIONS

3.1 Introduction



IMPORTANT!

Observe 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	1	1
DLL_COMPILER_VB	2	2
DLL_COMPILER_PASCAL	3	3
DLL_LABVIEW	4	4
DLL_COMPILER_VB5	5	5
APCI1710_SINGLE	0	0
APCI1710_CONTINUOUS	1	1
APCI1710_30MHZ	30	1E
APCI1710_33MHZ	33	21
APCI1710_40MHZ	40	28

3.2 Interrupt mask

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

Table 3-2: Interrupt mask of the function TOR

b_ModuleMask	ul_InterruptMask	Meaning
0000 0001	0001 0000 0000 0000	Interrupt occurred on TOR counter 0 from module 0
0000 0001	0010 0000 0000 0000	Interrupt occurred on TOR counter 1 from module 0
0000 0010	0001 0000 0000 0000	Interrupt occurred on TOR counter 0 from module 1
0000 0010	0010 0000 0000 0000	Interrupt occurred on TOR counter 1 from module 1
0000 0100	0001 0000 0000 0000	Interrupt occurred on TOR counter 0 from module 2
0000 0100	0010 0000 0000 0000	Interrupt occurred on TOR counter 1 from module 2
0000 1000	0001 0000 0000 0000	Interrupt occurred on TOR counter 0 from module 3
0000 1000	0010 0000 0000 0000	Interrupt occurred on TOR counter 1 from module 3

Table 3-3: Return table for counter value

b_ModuleMask	ul_InterruptMask	Source	ul_CounterLatchValue
b_ModuleMask = 1	ul_InterruptMask = 4096	Counter cycle of module 0, TOR counter 0 is stopped. The measurement is latched.	Counter value of TOR 0
b_ModuleMask = 1	ul_InterruptMask = 8192	Counter cycle of module 0, TOR counter 1 is stopped. The measurement is latched.	Counter value of TOR 1
b_ModuleMask = 2	ul_InterruptMask = 4096	Counter cycle of module 1, TOR counter 0 is stopped. The measurement is latched.	Counter value of TOR 0
b_ModuleMask = 2	ul_InterruptMask = 8192	Counter cycle of module 1, TOR counter 1 is stopped. The measurement is latched.	Counter value of TOR 1
b_ModuleMask = 4	ul_InterruptMask = 4096	Counter cycle of module 2, TOR counter 0 is stopped. The measurement is latched.	Counter value of TOR 0
b_ModuleMask = 4	ul_InterruptMask = 8192	Counter cycle of module 2, TOR counter 1 is stopped. The measurement is latched.	Counter value of TOR 1
b_ModuleMask = 8	ul_InterruptMask = 4096	Counter cycle of module 3, TOR counter 0 is stopped. The measurement is latched.	Counter value of TOR 0
b_ModuleMask = 8	ul_InterruptMask = 8192	Counter cycle of module 3, TOR counter 1 is stopped. The measurement is latched.	Counter value of TOR 1

3.3 Initialisation

1) i_APCI1710_InitTorCounter (...)

Syntax:

```
<Return Wert> = i_APCI1710_InitTorCounter
                                (BYTE      b_BoardHandle,
                                 BYTE      b_ModulNbr,
                                 BYTE      b_TorCounter,
                                 BYTE      b_PCInputClock,
                                 BYTE      b_TimingUnit,
                                 ULONG      ul_TimingInterval,
                                 PULONG     pul_RealTimingInterval)
```

Parameter:
-Input:

BYTE	b_BoardHandle	Handle of the board APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TorCounter	Selected TOR counter (0 or 1).
BYTE	b_PCInputClock	Selection of the PCI bus clock - APCI1710_30MHZ: The PC has a PCI bus clock of 30 MHz - APCI1710_33MHZ: The PC has a PCI bus clock of 33 MHz - APCI1710_40MHZ: The XPCI-1710 board has an integrated clock of 40 MHz
BYTE	b_TimingUnit	Base time unit (0 to 4) 0: ns 1: µs 2: ms 3: s 4: mn
ULONG	ul_TimingInterval	Base time value. See table 3-6: Base time value

-Output:

PULONG pul_RealTimingInterval Real time value.

Table 3-4: Base time value

PCI-Bus clock	b_TimingUnit	ul_TimingInterval Mindestwert	ul_TimingInterval Höchstwert
APCI1710_30MHz	ns (0)	133	4294967295
	µs (1)	1	571230650
	ms (2)	1	571230
	s (3)	1	571
	mn (4)	1	9
APCI1710_33MHz	ns (0)	121	4294967295
	µs (1)	1	519691043
	ms (2)	1	51969
	s (3)	1	520
	mn (4)	1	8
APCI1710_40MHZ	ns (0)	100	4294967295
	µs (1)	1	429496729
	ms (2)	1	429496
	s (3)	1	429
	mn (4)	1	7

Task:

Configures the selected TOR counter (*b_TorCounter*) of the selected module (*b_ModulNbr*). *ul_TimingInterval* and *ul_TimingUnit*, determines the time base for the measurement. *pul_RealTimingInterval* returns the real time value. This function must be called before any other function is called, which accesses to the TOR counter.

Calling convention

ANSI C :

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

```
i_ReturnValue = i_APCI1710_InitTorCounter
                (b_BoardHandle,
                 0,
                 0,
                 APCI1710_33MHZ,
                 1,
                 100,
                 &ul_RealTimingInterval);
```

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: The selected module is wrong

-3: The module is no TOR module

-4: Selected TOR counter is wrong

-5: The selected PCI input clock is wrong

-6: Selected time unit is wrong.

-7: Selected base time is wrong

-8: 40 MHz clock cannot be used with this board.

-9: 40 MHz clock cannot be used with this TOR version.

2) i_APCI1710_EnableTorCounter (...)

Syntax:

```
<Return Wert> = i_APCI1710_EnableTorCounter
                (BYTE b_BoardHandle,
                 BYTE b_ModulNbr,
                 BYTE b_TorCounter,
                 BYTE b_InputMode,
                 BYTE b_ExternGate,
                 BYTE b_CycleMode,
                 BYTE b_InterruptEnable)
```

Parameter:**-Input:**

BYTE	b_BoardHandle	Handle of the board APCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TorCounter	Selected TOR counter (0 or 1).
BYTE	b_InputMode	Selection of the input signal level 0: TOR counts at each „Low" level 1: TOR counts at each "High" level 2: APCI1710_TOR_SIMPLE_MODE: Function like 4 fold MODE, but only one of the four edges per period is evaluated. 3: APCI1710_TOR_DOUBLE_MODE Function like 4fold-MODE, but only two of the four edges are evaluated. 4: APCI1710_TOR_QUADRUPLE _MODE: The edge evaluation switch generates a counting pulse in the 4 fold MODE from two phase shifted signals.
BYTE	b_ExternGate	Selected action for the external gate 0: External gate signal not used 1: External gate signal used. If the single mode is selected, each High level signal will start the counter. If the continuous mode is selected, the first High level signal will start the TOR counter. The gate input is used as signal B.
BYTE	b_CycleMode	Acquisition mode of the TOR counter
BYTE	b_InterruptEnable	TOR counter interrupt APCI1710_ENABLE: Enables the TOR counter Interrupt APCI1710_DISABLE: Disables the TOR counter Interrupt

-Output:

There is no output

Task:

Releases the TOR counter (*b_TorCounter*) of the indicated module (*b_ModulNbr*). The function "i_APCI1710_InitTorCounter" shall be called before this function. If the TOR counter interrupt is released, the TOR counter generates an interrupt after the time cycle is run out. See function "i_APCI1710_SetBoardIntRoutineX" and the description of the interrupt mask in chapter 3.2. The *b_CycleMode* parameter determines if the measurement runs for one or several cycles.

Calling convention:

ANSI C :

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

```
i_ReturnValue = i_APCI1710_EnableTorCounter
                (b_BoardHandle,
                 0,
                 0,
                 1,
                 0
                 APCI1710_SINGLE,
                 APCI1710_DISABLE);
```

Return value:

- 0: No error
 - 1: The handle parameter of the board is wrong.
 - 2: The selected module is wrong.
 - 3: The module is no TOR module
 - 4: The selected TOR counter is wrong
 - 5: TOR counter not initialised. See function "i_APCI1710_InitTorCounter"
 - 6: The selected TOR input signal is wrong
 - 7: The selected mode for the external gate signal is wrong
 - 8: The selected acquisition mode of the TOR counter is wrong.
 - 9: Interrupt parameter is wrong
 - 10: Interrupt function not initialised.
- See function "i_APCI1710_SetBoardIntRoutineX"

3) i_APCI1710_DisableTorCounter (...)

Syntax:

```
<Return Wert> = i_APCI1710_DisableTorCounter
                    (BYTE b_BoardHandle,
                     BYTE b_ModulNbr,
                     BYTE b_TorCounter)
```

Parameters:
-Input:

BYTE	b_BoardHandle	Handle of the board APCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TorCounter	Selected TOR counter (0 or 1).

- Output:

There is no output..

Task:

Disables the TOR counter (*b_TorCounter*) of the selected module (*b_ModulNbr*). The status register will be deleted, if the TOR counter is deactivated after a cycle has started and if the TOR counter starts again with the function "i_APCI1710_EnableTorCounter"

Calling convention:

ANSI C :

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

```
i_ReturnValue = i_APCI1710_DisableTorCounter
                (b_BoardHandle,
                 0,
                 0);
```

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: The selected module is wrong

-3: The module is no TOR module

-4: The selected TOR counter is wrong

-5: TOR counter is not initialised. See function: "i_APCI1710_InitTorCounter"

-6: TOR counter is not released.

See function "i_APCI1710_EnableTorCounter"

4) i_APCI1710_GetTorCounterInitialisation (...)

Syntax:

```
<Return Wert> = i_APCI1710_GetTorCounterInitialisation
                (BYTE    b_BoardHandle,
                 BYTE    b_ModulNbr,
                 BYTE    b_TorCounter,
                 PBYTE   pb_TimingUnit,
                 PULONG  pul_TimingInterval,
                 PBYTE   pb_InputMode,
                 PBYTE   pb_ExternGate,
                 PBYTE   pb_CycleMode,
                 PBYTE   pb_Enable,
                 PBYTE   pb_InterruptEnable)
```

Parameter:**-Input:**

BYTE	b_BoardHandle	Handle of the board APCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TorCounter	Selected TOR counter (0 or 1)

- Output:

PBYTE	pb_TimingUnit	Base time unit (0 to 4) 0: ns 1: µs 2: ms 3: s 4: mn
PULONG	pul_TimingInterval	Base time value. See table.
PBYTE	pb_InputMode	Selection of the input signal level 0: TOR counts at each "Low" level 1: TOR counts at each "High" level
PBYTE	pb_ExternGate	Selected action for the external gate 0: External gate signal not used 1: External gate signal is used..
PBYTE	pb_CycleMode	Cycle type of the TOR acquisition APCI1710_SINGLE: Single cycle-MODE. The TOR counter counts only one cycle The function is to be called at each cycle. APCI1710_CONTINUOUS: Each time cycle starts a new counting process.
PBYTE	pb_Enable	Indicates if the TOR counter is released or not. 0: TOR counter disabled 1: TOR counter enabled
PBYTE	pb_InterruptEnable	Selection of the TOR nterrupt. APCI1710_ENABLE: TOR Interrupt enabled. APCI1710_DISABLE: TOR Interrupt disabled

Task:

Returns the TOR (*b_TorCounter*) initialisation of the selected module (*b_ModulNbr*). You must call up the "i_APCI1710_InitTorCounter" function before calling up this function

Calling convention:

ANSI C :

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned char b_TimingUnit;
unsigned char b_InputMode;
unsigned char b_CycleMode;
unsigned char b_InterruptEnable;
unsigned long ul_TimingInterval;
```

```
i_ReturnValue = i_APCI1710_GetTorCounterInitialisation
                (b_BoardHandle,
                 0,
                 0,
                 &b_TimingUnit,
                 &ul_TimingInterval,
                 &b_InputMode,
                 &b_CycleMode,
                 &b_InterruptEnable);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The selected module is wrong
- 3: The module is no TOR module
- 4: The selected TOR counter is wrong
- 5: TOR is not initialised. See function: "i_APCI1710_InitTorCounter"

3.4 Read TOR counter

1) i_APCI1710_GetTorCounterProgressStatus (...)

Syntax:

```
<Return Wert> = i_APCI1710_GetTorCounterProgressStatus
                                (BYTE    b_BoardHandle,
                                 BYTE    b_ModulNbr,
                                 BYTE    b_TorCounter,
                                 PBYTE   pb_TorCounterStatus)
```

Parameter:

-Input:

BYTE	b_BoardHandle	Handle of the board APCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TorCounter	Selected TOR counter (0 or 1)

-Output:

PBYTE	pb_TorCounterStatus	Returns the TOR status
		0: Counting cycle not started. Software gate not set.
		1: Counting cycle is started. Software gate is set
		2: Counting cycle stopped and ended.
		3: Time is run out. Base time must be changed with the function "i_APCI1710_InitTorCounter"

Task:

Returns the status of the selected TOR counter (*b_TorCounter*) in the selected module (*b_ModulNbr*).

Calling convention:

ANSI C :

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

```
i_ReturnValue = i_APCI1710_GetTorCounterProgressStatus
                (b_BoardHandle,
                 0,
                 0,
                 &pb_TorCounterStatus);
```

Return value:

0: No error
 -1: The handle parameter of the board is wrong
 -2: The selected module is wrong
 -3: The module is no TOR module

- 4: The selected TOR counter is wrong
- 5: TOR is not initialised. See function "i_APCI1710_InitTorCounter"
- 6: TOR is not released. See function: "i_APCI1710_EnableTorCounter"

2) **i_APCI1710_ReadTorCounterValue (...)****Syntax:**

<Return Wert> = i_APCI1710_ReadTorCounterValue
 (BYTE b_BoardHandle,
 BYTE b_ModulNbr,
 BYTE b_TorCounter,
 UINT ui_TimeOut,
 PBYTE pb_TorCounterStatus,
 PULONG pul_TorCounterValue)

Parameter:**-Input:**

BYTE	b_BoardHandle	Handle of the board APCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TorCounter	Selected TOR counter (0 or 1)
UINT	ui_TimeOut	Selection of the sequence time (0 to 65535) 0: No time sequence used. The function returns the TOR status and checks if the counter cycle has stopped the measured counting value. 1 to 65535: Determines the time sequence in ms. The function will be returned after Timeout or counting cycle stop.

- Output:

PBYTE	pb_TorCounterStatus	Returns the TOR status 0: Counting cycle not started. Software gate is not set. 1: Counting cycle is started Software gate I set. 2: Counting cycle is stopped. The counting cycle will be ended 3: An overflow has occurred. The time base must be changed with the function "i_APCI1710_InitTorCounter" 4: Timeout is occurred.
PULONG	pul_TorCounterValue	TOR counting value.

Task:

Returns the status (*pb_TorCounterStatus*) of the TOR counter (*b_TorCounter*) and the counting value (*pul_TorCounterValue*) of the selected module (*b_ModulNbr*) after a counting cycle is stopped.

Calling convention:

ANSI C :

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned char b_TorCounterStatus;
unsigned long ul_TorCounterValue;

i_ReturnValue = i_APCI1710_ReadTorCounterValue
                (b_BoardHandle,
                 0,
                 0,
                 0,
                 &pb_TorCounterStatus,
                 &ul_TorCounterValue);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The selected module is wrong
- 3: The module is no TOR module
- 4: The selected TOR counter is wrong
- 5: TOR is not initialised. See function "i_APCI1710_InitTorCounter"
- 6: TOR is not released. See function "i_APCI1710_EnableTorCounter"
- 7: Timeout parameter is wrong (0 to 65535)

3.5 Interrupt kernel functions for Windows NT/95



IMPORTANT!

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

1) i_APCI1710_KRNL_GetTorCounterProgressStatus (...)

Syntax:

```
<Return Wert> = i_APCI1710_KRNL_GetTorCounterProgressStatus
                                (UINT          ui_BaseAddress,
                                 BYTE           b_ModulNbr,
                                 BYTE           b_TorCounter,
                                 PBYTE         pb_TorCounterStatus)
```

Parameter:

-Input:

UINT	ui_BaseAddress	Base address of the board APCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TorCounter	Selektierter TOR Zähler (0 oder 1)

- Ausgabe:

PBYTE	pb_TorCounterStatus	Returns the TOR status
		0: Counting cycle is not started. Software gate is not set.
		1: Counting cycle is started. Software gate is set
		2: Counting cycle is stopped. The counting cycle is stopped.
		3: an overflow occurred. The time base must be changed with the function "i_APCI1710_InitTorCounter"
		4: Timeout occurred

Task:

Returns the TOR status (*pb_TorCounterStatus*) of the selected TOR module (*b_ModulNbr*).

Calling convention:

ANSI C :

```
int          i_ReturnValue;
unsigned int  ui_BaseAddress;
unsigned char b_TorCounterStatus;
i_ReturnValue = i_APCI1710_KRNL_GetTorCounterProgressStatus
                (ui_BaseAddress,
                 0,
                 0,
                 &pb_TorCounterStatus);
```

Return value:

0: No error

-1: The selected module is wrong

-2: The module is no TOR module

-3: The selected TOR counter is wrong

-4: TOR is not initialised. See function "i_APCI1710_InitTorCounter"

-5: TOR is not released. See function "i_APCI1710_EnableTorCounter"

2) **i_APCI1710_KRNL_ReadTorCounterValue (...)****Syntax:**

```
<Return Wert> = i_APCI1710_KRNL_ReadTorCounterValue
                (UINT      ui_BaseAddress,
                 BYTE      b_ModulNbr,
                 BYTE      b_TorCounter,
                 PBYTE     pb_TorCounterStatus,
                 PULONG    pul_TorCounterValue)
```

Parameter:**-Input:**

UINT	ui_BaseAddress	Base address of the board APCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TorCounter	Selected TOR counter (0 or 1)

-Output:

PBYTE	pb_TorCounterStatus	Returns the TOR status 0: Counting cycle is not started. Software gate is not set. 1: Counting cycle is started. Software gate is set. 2: Counting cycle is stopped. The counting cycle will be ended 3: An overflow occurred. The time base must be changed with the function "i_APCI1710_InitTorCounter" 4: Timeout occurred.
PULONG	pul_TorCounterValue	TOR counter value.

Task:

Returns the TOR status (*pb_TorCounterStatus*) and the TOR counter value (*pul_TorCounterValue*) of the selected TOR module (*b_ModulNbr*) after a counting cycle stopped.

Calling convention:

ANSI C :

```
int          i_ReturnValue;
unsigned int  ui_BaseAddress;
unsigned char b_TorCounterStatus;
unsigned long ul_TorCounterValue;
i_ReturnValue = i_APCI1710_KRNL_ReadTorCounterValue
                (ui_BaseAddress,
                 0,
                 0,
                 &pb_TorCounterStatus,
                 &ul_TorCounterValue);
```

Return value:

0: No error

-1: The selected module is wrong

-2: The module is no TOR module

-3: The selected TOR counter is wrong

-4: TOR is not initialised. See function "i_APCI1710_InitTorCounter"

-5: TOR is not released. See function: "i_APCI1710_EnableTorCounter"