



DIN EN ISO 9001:2000
certified



ADDI-DATA GmbH
Dieselstraße 3
D-77833 OTTERSWEIER



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

Technical description

ADDICOUNT APCI-/CPCI-1710

Pulse Width Modulation (PWM)

Edition: 02.02 – 07/2006

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

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 WIDTH MODULATION (PWM).....	9
2.1	Function description	9
2.1.1	Block diagram of the PWM function.....	9
2.1.2	Typical applications	9
2.2	Used signals	10
2.3	Pin assignment of the PWM.....	11
2.4	Connection example	11
2.5	I/O mapping of the PWM function	12
2.6	Description of the I/O functions.....	12
2.6.1	Timer Register	13
	Low level	13
	High level	13
2.6.2	PWM commando register	13
2.6.3	PWM status register.....	13
2.6.4	PWM synchro/gate/interrupt status register	14
2.6.5	Digital output	14
2.6.6	Digital inputs.....	14
2.6.7	Version register	14
2.7	Limit values.....	15
2.8	Working with PWM	16
3	STANDARDSOFTWARE.....	17
3.1	Define values.....	17
3.2	Interrupt mask	17
3.3	PWM functions.....	18
3.3.1	Initialisation.....	18
	1) i_APCI1710_InitPWM (...)	18
	2) i_APCI1710_EnablePWM (...).....	22
	3) i_APCI1710_SetNewPWMTiming (...)	25
	4) i_APCI1710_DisablePWM (...)	27
	5) i_APCI1710_GetPWMInitialisation (...).....	29
3.3.2	PWM status	32
	6) i_APCI1710_GetPWMStatus (...)	32

3.3.3	Windows NT/95 interrupt kernel functions	34
1)	i_APCI1710_KRNL_GetPWMStatus (...)	34
2)	i_APCI1710_KRNL_SetNewPWMTiming (...)	36
3.3.4	Digital output	39
1)	i_APCI1710_EnableDisablePWMDigitalOutputManualSetting (...)	39
2)	i_APCI1710_SetPWMDigitalOutputOn (...)	40
3)	i_APCI1710_SetPWMDigitalOutputOff (...)	41
3.3.5	Digital inputs	42
1)	i_APCI1710_ReadPWM1DigitalInput (...)	42
2)	i_APCI1710_ReadPWMAllDigitalInput (...)	43

Figures

Fig. 2-1: Block diagram of the PWM function.....	9
Fig. 2-2: Pin assignment of the front connector	11
Fig. 2-3: Connection example	11

Tables

Table 1-1: Delivered manuals.....	8
Table 2-1: Used signals	10
Table 2-2: I/O mapping of the PWM function	12
Table 3-1: Define table	17
Table 3-2: Interrupt mask of the function PWM.....	17
Table 3-3: Time value table	19

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 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** 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 standardsoftware are also identical.

1.4 Function description

In addition to the 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 WIDTH MODULATION (PWM)

2.1 Function description

The function „PWM“ is a pulse width modulation interface which allows to generate an output signal with defined low and high level timing.

With this function are available:

- 32-bit timer to set the LOW and HIGH level
- 2 digital inputs as start or stop trigger
- 2 digital outputs for the frequency output

Properties:

Complete isolation through optical couplers for the input and output channels to avoid earth circuit

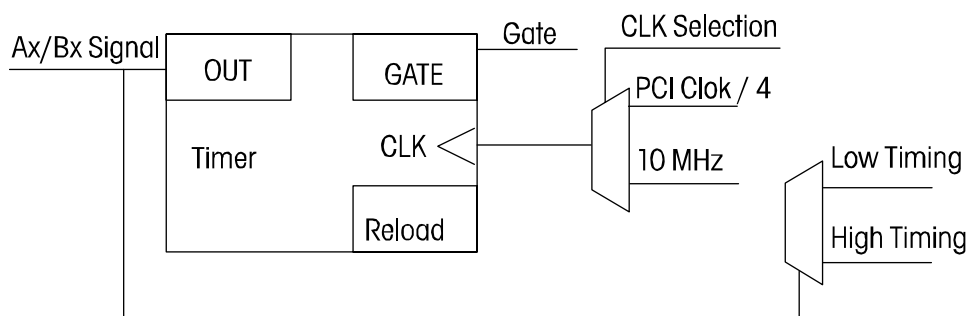
- Interrupt status at the end of period
- Signals up to 2.5 MHz can be handled
- Start level selection
- Stop level selection
- Hardware gate
- Software gate

2.1.1 Block diagram of the PWM function

The interface contains:

- Digital inputs
- Digital outputs

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



2.1.2 Typical applications

- Frequency generation
- Pulse Width Modulation (PWM)
- Drives

2.2 Used signals

The function "PWM" occupies **5 inputs and 3 outputs** on a function module.

Table 2-1: Used signals

Signals at the connector	Polarity	Function
Ax + / -	Diff. / TTL	Digital output (PWM0)
Bx + / -	Diff. / TTL	Digital output (PWM1)
Cx + / -	Diff. / TTL / (24V)	External gate (PWM0)
Dx +/-	Diff. / TTL / (24V)	External gate (PWM0)
E	24 V (5 V)	Digital input
F	24 V (5 V)	Digital input
G	24 V (5 V)	Digital input
H	24 V (5 V)	Digital output PWM10 or freely controllable

x: Number of the function module.

(..) Option

2.3 Pin assignment of the PWM

i

IMPORTANT!

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

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

The figure below shows the **max. use of the PWM on the board**. The function is implemented on all function modules.

Fig. 2-2: Pin assignment of the front connector

Pin		Pin		Pin							
Digital output1	34	+UREF/24 V ext.	Function module 3	18	A3 +	34	18	1	Ext. GND	1	Function module 1
	35	H1		19	A3 -			2	A1 +	2	
	36	H2		20	B3 +			3	A1 -	3	
	37	H3		21	B3 -			4	B1 +	4	
	38	H4		22	C3 +			5	B1 -	5	
Digital input1	39	E1	23	C3 -	39	6	C1 +	6	Function module 2		
	40	E2	24	D3 +	40	7	C1 -	7			
	41	E3	25	D3 -	41	8	D1 +	8			
	42	E4	26	A4 +	42	9	D1 -	9			
Digital input 2	43	F1	Function module 4	27	A4 -	43	10	A2 +	10		
	44	F2		28	B4 +	44	11	A2 -	11		
	45	F3		29	B4 -	45	12	B2 +	12		
	46	F4		30	C4 +	46	13	B2 -	13		
Digital input 3	47	G1	31	C4 -	47	14	C2 +	14	Function module 2		
	48	G2	32	D4 +	48	15	C2 -	15			
	49	G3	33	D4 -	49	16	D2 +	16			
	50	G4	50	33	50	17	D2 -	17			

2.4 Connection example

Fig. 2-3: Connection example

2 Ax+

3 Ax-

electr. motor

11

2.5 I/O mapping of the PWM function

Table 2-2: I/O mapping of the PWM function

	IOWR		IORD	
	D31.....D8	D7.....D0	D31.....D8	D7.....D0
BASEx + 0	PWM0 Timer (Low-Reload value)		PWM0 Low Base Timing	
BASEx + 4	PWM0 Timer (High-Reload value)		PWM0 High Base Timing	
BASEx + 8	-	PWM0 commando		PWM0 commando
BASEx + 12	-	PWM0/PWM1 synchro-gate		PWM0 status
BASEx + 16	-	Status register	-	PWM0 interrupt status
BASEx + 20	PWM1 Timer (Low-Reload value)		PWM1 Low Base Timing	
BASEx + 24	PWM1 Timer (High-Reload value)		PWM1 High Base Timing	
BASEx + 28	-	PWM1 commando		PWM1 commando
BASEx + 32	-	PWM1 gate		PWM1 status
BASEx + 36	-	PWM0/PWM1 synchro gate		PWM1 interrupt status
BASEx + 40		Digital output		Digital output
BASEx + 44				Digital input status
BASEx + 60	-		VERSION register	

-: No function

x: Number of the function module

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

2.6 Description of the I/O functions

The function „PWM“ is a scaled-down version of the Counter/Timer function for the APCI-/CPCI-1710 function. As function the PCI clock/4 generates rectangular signals. The output pulse signals from the timer generate the pulse width modulation.

PWM generator

The low/high timing division factor is written into the timer and determines the output frequency. The PCI clock/4 or the 40 MHz quartz of the board can be set as input frequency.

2.6.1 Timer Register

Low level

The "Low" reload value of the timer is written. The duration of the low level for the timer is determined by the division factor.

High level

The "High" reload value of the timer is written. The duration of the high level for the timer is determined by the division factor.

2.6.2 PWM commando register

DQ0:	0:	The current period is stopped directly after the gate reset.
	1:	The current period is stopped after the gate reset and the end of period
DQ1:	0:	The output signal keeps the level condition after the stop signal
	1:	The output is set on Low or High after the stop signal
DQ2:	0:	The output is set on Low after the stop signal
	1:	The output is set on High after the stop signal
DQ3:	0:	Interrupt generation disabled
	1:	Interrupt generation enabled
DQ4:	0:	External input rate not used
	1:	External input rate started PWM
DQ5:	0:	The period starts with a low level
	1:	The period starts with a high level
DQ6:		Not used
DQ7:	0:	PCI bus clock/4 is used as time base
	1:	10 MHz clock is used as time base

2.6.3 PWM status register

Write:

DQ0:	0:	Disables PWM
	1:	Enables PWM
DQ4:	0:	No function
	1:	Resets the PWM initialisation

Read:

DQ0:	0:	PWM Software gate not set
	1:	PWM Software gate set
DQ4:	0:	PWM not initialised
	1:	PWM initialised
DQ5:	0:	PWM not started
	1:	PWM is running
DQ6:	0:	External input rate is Low
	1:	External input rate is High

DQ7:	0:	Output signal is Low
	1:	Output signal is High

2.6.4 PWM synchro/gate/interrupt status register

Write:

DQ0:	0:	Disables PWM0 and PWM1
	1:	Enables PWM0 and PWM1
DQ4:	0:	No function
	1:	Resets the initialisation of PWM0 and PWM1

Read:

DQ0:	1:	Interrupt at the end of period
		The interrupt request is reset through reading of the register
DQ31..1		Is always set on 0

2.6.5 Digital output

Read/Write:

DQ0:	0:	Digital output H is controlled parallel to the signal PWM
	1:	Digital output H can be controlled manually through DQ1
DQ1:	0:	Sets output H to 0 if DQ0 was set
	1:	Sets output H to 1 if DQ0 was set
DQ31..1		Is always set on 0

2.6.6 Digital inputs

Read:

DQ0:	0:	Digital E is low (0)
	1:	Digital E is high (1)
DQ1:	0:	Digital input F is low (0)
	1:	Digital input F is high (1)
DQ2:	0:	Digital input G is low (0)
	1:	Digital input G is high (1)
DQ31..1		Is always set on 0

2.6.7 Version register

Base address + 60

Contains the function description and the revision. (reading command, ASCII format)

Example

BASE + 60 "P" "W" "1" "0"

Meaning: PWM; Revision 1.0

2.7 Limit values

Max. output frequency: 5 MHz

Division factor: 4 to $2^{32} - 1$

2.8 Working with PWM

1. Select a function module
2. Select a channel.
3. Select the input frequency.
4. Define the Start/Stop level.
5. Define the Low time
6. Define the High time.
7. Select the start condition (external gate or software start)
8. Start the PWM output.

3 STANDARDSOFTWARE

3.1 Define values

Table 3-1: Define table

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_30MHZ	30	1E
APCI1710_33MHZ	33	21
APCI1710_40MHZ	40	28

3.2 Interrupt mask

Each PWM timer can generate an interrupt after the period cycle had run out.

In order to get this interrupt:

- You shall release the interrupt with "i_APCI1710_EnablePWM" and
- Set the interrupt routine with the function "i_APCI1710_SetBoardIntRoutineX"

Table 3-2: Interrupt mask of the function PWM

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

3.3 PWM functions

3.3.1 Initialisation

1) i_APCI1710_InitPWM (...)

Syntax:

```
<Return Wert> = i_APCI1710_InitPWM
                                (BYTE      b_BoardHandle,
                                BYTE      b_ModulNbr,
                                BYTE      b_PWM,
                                BYTE      b_ClockSelection,
                                BYTE      b_TimingUnit,
                                ULONG      ul_LowTiming,
                                ULONG      ul_HighTiming,
                                PULONG     pul_RealLowTiming,
                                PULONG     pul_RealHighTiming)
```

Parameter

-Input :

BYTE	b_BoardHandle	Handle of the board xPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_PWM	Selected PWM generator (0 or 1).
BYTE	b_ClockSelection	Selection of the clock signal - APCI1710_30 MHZ: The board uses a PCI bus of 30 MHz - APCI1710_33MHZ: The board uses a PCI bus clock of 33 MHz - APCI1710_40MHZ: The board uses the 40 MHz quartz clock
BYTE	b_TimingUnit	Selection of the time unit (0 to 4) 0: ns 1: µs 2: ms 3: s 4: mn
ULONG	ul_LowTiming	Time of the low level. See Table 3-1: Define table
ULONG	ul_HighTiming	Time of the high level. See Table 3-1.

-Output:

PULONG	pul_RealLowTiming	Actual duration of the Low level.
PULONG	pul_RealHighTiming	Actual duration of the High level.

Table 3-3: Time value table

Clock selection	<i>b_TimingUnit</i>	<i>ul_TimingInterval</i> minimal value	<i>ul_TimingInterval</i> maximal value
APCI1710_30MHZ	ns (0)	266	4294967295
	μs (1)	1	571230650
	ms (2)	1	571230
	s (3)	1	571
	mn (4)	1	9
APCI1710_33MHZ	ns (0)	242	4294967295
	μs (1)	1	519691043
	ms (2)	1	519691
	s (3)	1	520
	mn (4)	1	8
APCI1710_40MHZ	ns (0)	200	4294967295
	μs (1)	1	429496729
	ms (2)	1	429496
	s (3)	1	429
	mn (4)	1	7

Task:

Configures the selected PWM generator (*b_PWM*) of the selected module (*b_ModulNbr*).

ul_LowTiming, *ul_HighTiming* and *ul_TimingUnit* determine the time of the Low/High level

pul_RealLowTiming, *pul_RealHighTiming* return the real time value.

Call this function before you access other PWM functions.

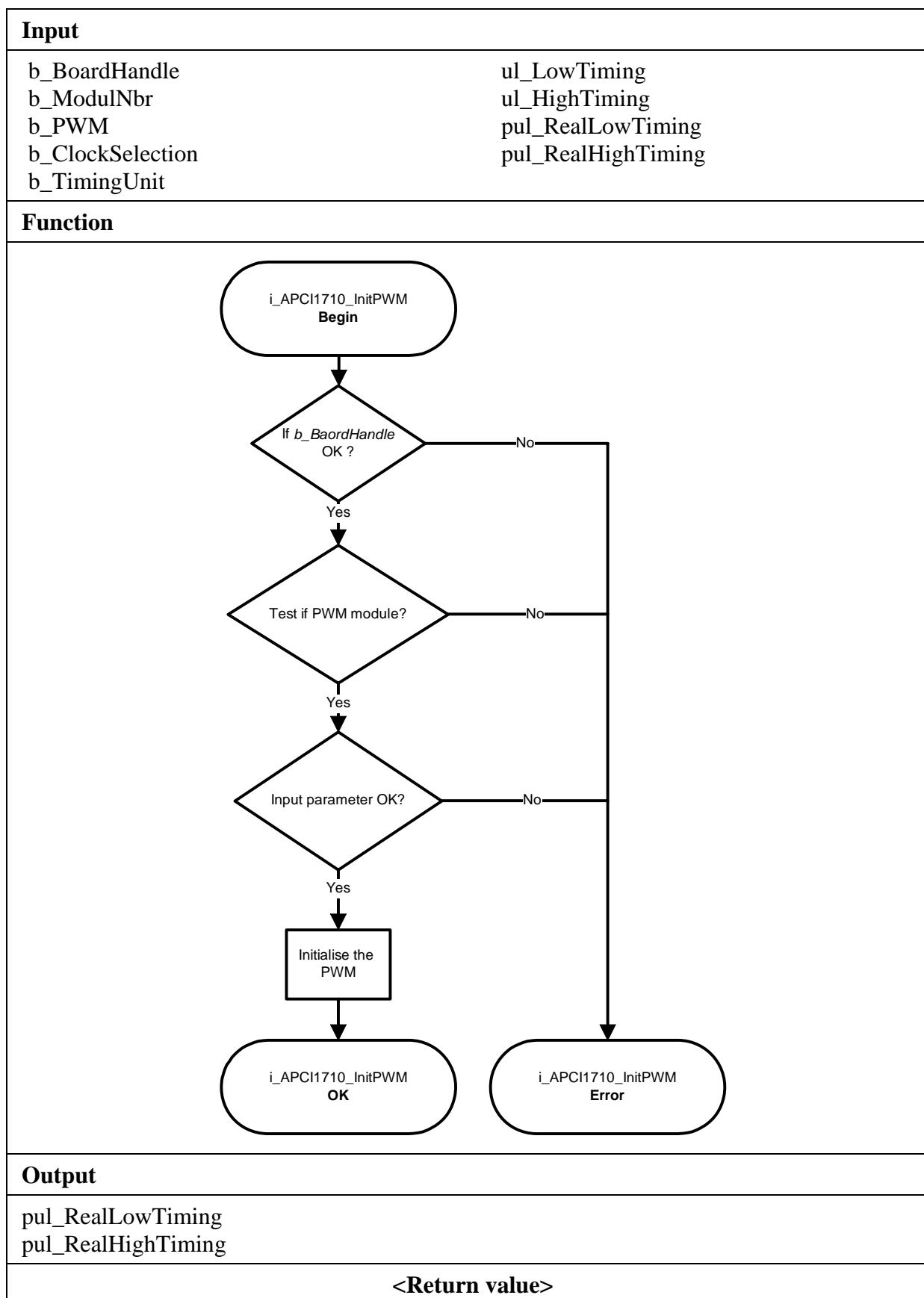
Calling convention:ANSI C:

```
int          i_ReturnValue;  
unsigned char b_BoardHandle;  
unsigned long ul_RealLowTiming;  
unsigned long ul_RealHighTiming;
```

```
i_ReturnValue = i_APCI1710_InitPWM  
                (b_BoardHandle,  
                 0,  
                 0,  
                 APCI1710_40MHZ,  
                 1,  
                 100,  
                 500,  
                 &ul_RealLowTiming,  
                 &ul_RealHighTiming);
```

Return value:

- 0: No error
- 1: Handle parameter of the board is wrong
- 2: The selected module number is wrong
- 3: The selected module is no "PWM" module
- 4: PWM selection is wrong
- 5: The selected input clock is wrong
- 6: The selected time unit is wrong
- 7: Selected "Low" base time is wrong
- 8: Selected "High" base time is wrong
- 9: The 40 MHz clock cannot be used with this board



2) i_APCI1710_EnablePWM (...)

Syntax:

```
<Return Wert> = i_APCI1710_EnablePWM
                                (BYTE    b_BoardHandle,
                                BYTE    b_ModulNbr,
                                BYTE    b_PWM,
                                BYTE    b_StartLevel,
                                BYTE    b_StopMode,
                                BYTE    b_StopLevel,
                                BYTE    b_ExternGate,
                                BYTE    b_InterruptEnable)
```

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_PWM	Selected PWM generator (0 or 1)
BYTE	b_StartLevel	Selection of the level before the start 0: The level starts on the low level 1: The period start on the high level
BYTE	b_StopMode	Selection of the Stop mode 0: PWM is stopped directly after the function "i_APCI1710_DisablePWM" and the current period is stopped. 1: After the function "i_APCI1710_DisablePWM", PWM is stopped at the end of period.
BYTE	b_StopLevel	Stops the selection of the PWM level 0: The output signal keeps its level condition after calling of the function "i_APCI1710_DisablePWM" 1: The output signal is set on Low after calling of the function "i_APCI1710_DisablePWM" 2: The output signal is set on High after calling the function "i_APCI1710_DisablePWM"
BYTE	b_ExternGate	Selection of the external gate action 0: External gate signal not used 1: External gate signal used
BYTE	b_InterruptEnable	Enables or disables the PWM interrupt. APCI1710_ENABLE: Enables the PWM interrupt. An interrupt occurs after each period APCI1710_DISABLE: Disables the PW interrupt

-Output:

There is no output.

Task:

Enables the selected PWM generator (*b_PWM*) of the selected module (*b_ModulNbr*).

Call the function "i_APCI1710_InitPWM" before accessing an other function.

If the PWM interrupt is enabled, the PWM generates an interrupt after each period. See function. "i_APCI1710_SetBoardIntRoutineX" and chapter 3.2.

Calling convention:ANSI C:

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

```
i_ReturnValue = i_APCI1710_EnablePWM
                                   (b_BoardHandle,
                                   0,
                                   0,
                                   0,
                                   1,
                                   0,
                                   0,
                                   APCI1710_DISABLE);
```

Return value:

0: No error

-1: Handle parameter of the board is wrong

-2: The selected module number is wrong.

-3: The selected module is no "PWM" module.

-4: PWM selection is wrong.

-5: PWM not initialised, see function "i_APCI1710_InitPWM"

-6: Selection of the PWM start level is wrong

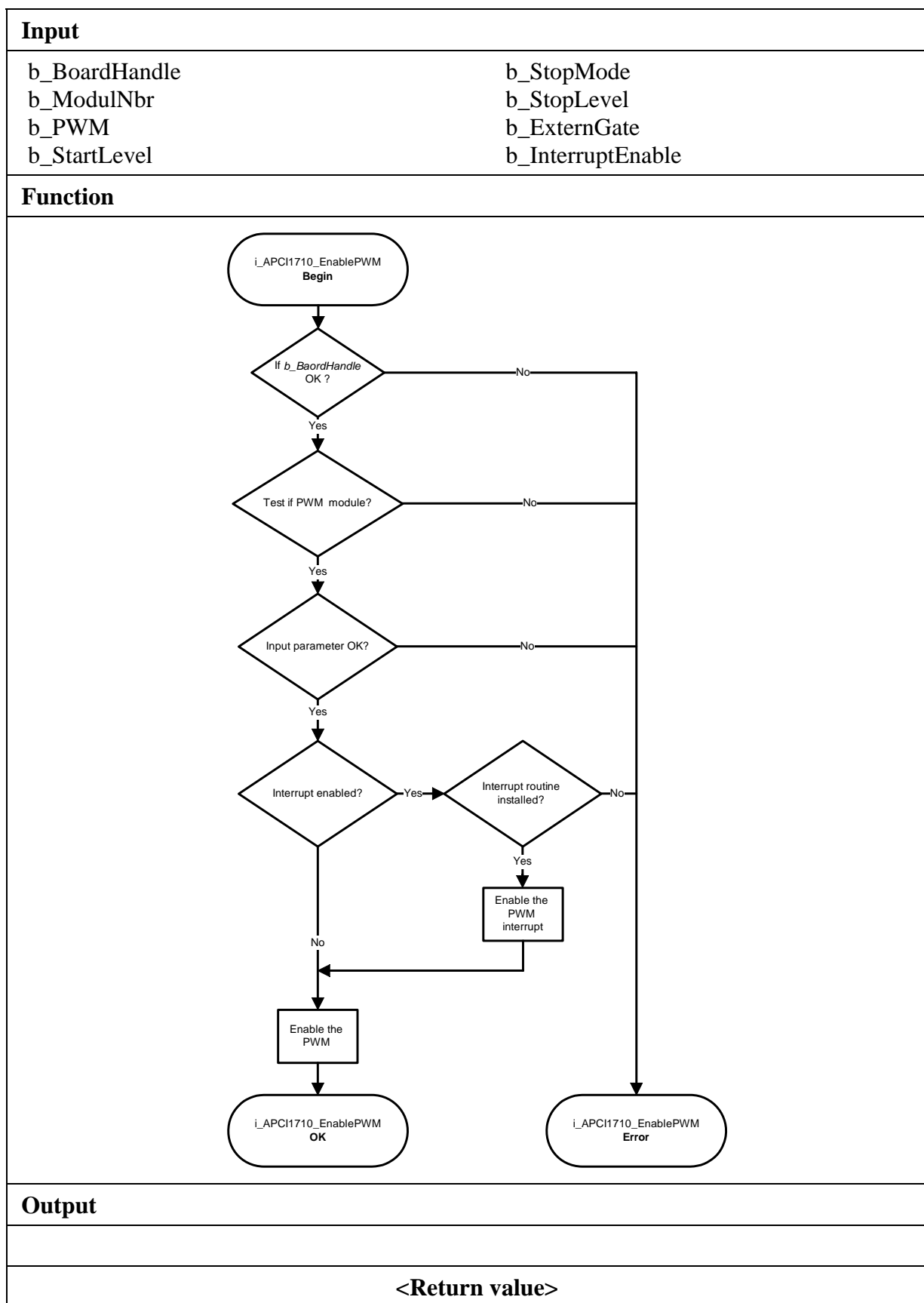
-7: Selection of the PWM Stopp mode is wrong

-8: Selection of the PWM Stopp level is wrong.

-9: Selection of the external gate signal is wrong

-10: Interrupt parameter is wrong

-11: Interrupt function is not initialised. See function "i_APCI1710_SetBoardIntRoutineX"



3) i_APCI1710_SetNewPWMTiming (...)

Syntax:

```
<Return Wert> = i_APCI1710_InitPWM
                                (BYTE      b_BoardHandle,
                                BYTE      b_ModulNbr,
                                BYTE      b_PWM,
                                BYTE      b_TimingUnit,
                                ULONG     ul_LowTiming,
                                ULONG     ul_HighTiming)
```

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_PWM	Selected PWM generator (0 or 1).
BYTE	b_TimingUnit	Selection of the time unit (0 to 4) 0: ns 1: µs 2: ms 3: s 4: mn
ULONG	ul_LowTiming	Time of the low level. See Table 3-1
ULONG	ul_HighTiming	Time of the high level. See Table 3-1

-Output:

There is no output.

Task:

Sets a new time period. *ul_LowTiming*, *ul_HighTiming* and *ul_TimingUnit* determine the LOW/High time base for the period.

Calling convention:

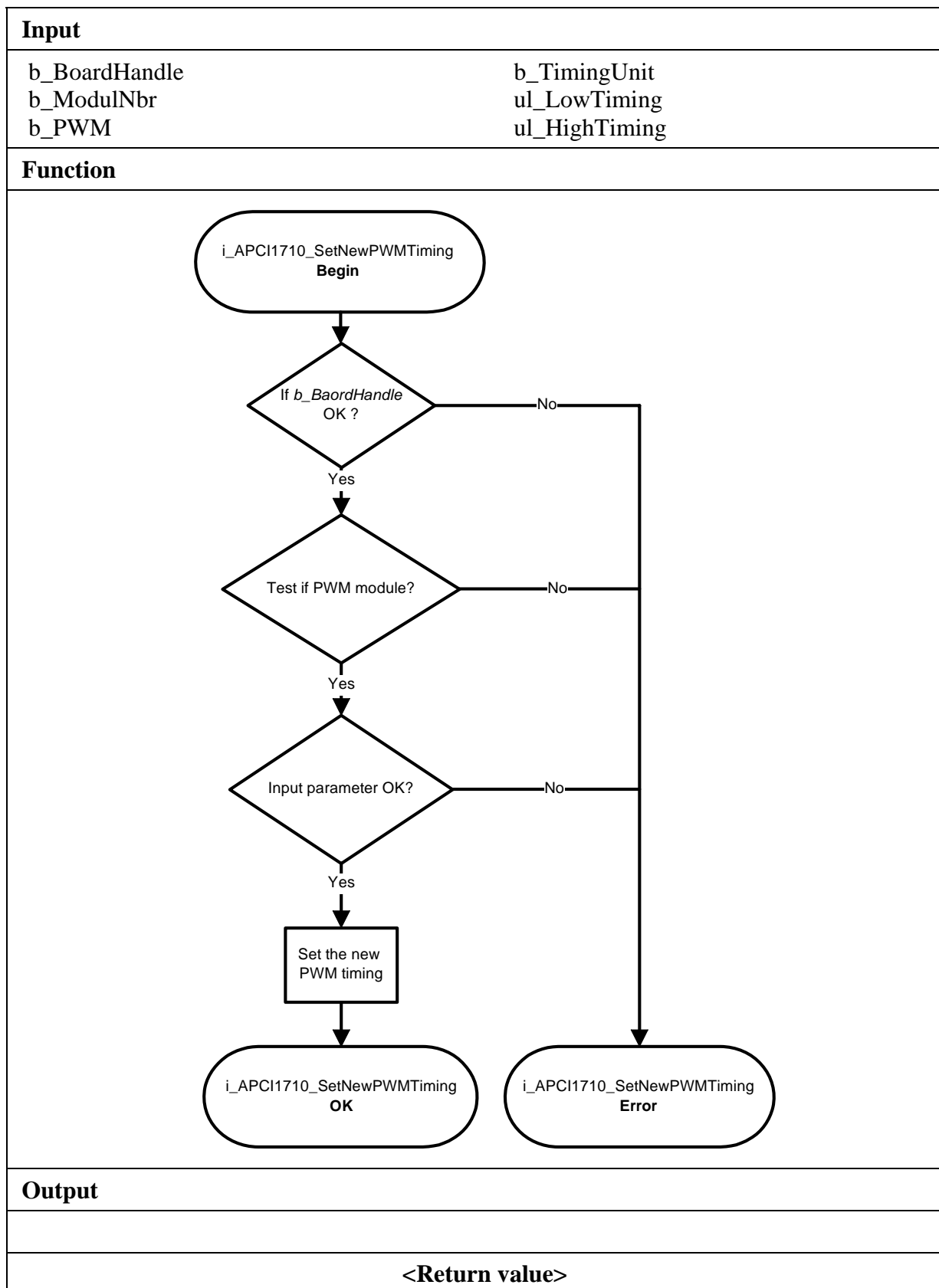
ANSI C:

```
int      i_ReturnValue;
unsigned char b_BoardHandle;
i_ReturnValue = i_APCI1710_InitPWM
                                (b_BoardHandle,
                                0,
                                0,
                                1,
                                100,
                                500);
```

Return value:

0: No error
 -1: Handle parameter of the board is wrong
 -2: The selected module number is wrong.
 -3: The selected module is no "PWM" module.
 -4: PWM selection is wrong

- 5: PWM is not initialised. See function "i_APCI1710_InitPWM"
- 6: The selected time unit is wrong
- 7: Selected "Low" base time is wrong
- 8: Selected "High" base time is wrong



4) i_APCI1710_DisablePWM (...)**Syntax:**

```
<Return Wert> = i_APCI1710_DisablePWM
                                (BYTE    b_BoardHandle,
                                BYTE    b_ModulNbr,
                                BYTE    b_PWM)
```

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_PWM	Selected PWM generator (0 or 1)

-Output:

There is no output

Task:

Disables the selected PWM (*b_PWM*) of the selected module (*b_ModulNbr*).

The output signal level depends on the initialisation with

"i_APCI1710_EnablePWM".

See the parameters *b_StartLevel*, *b_StopMode* and *b_StopLevel* of this function.

Calling convention:ANSI C:

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

```
i_ReturnValue = i_APCI1710_DisablePWM
                                (b_BoardHandle,
                                0,
                                0);
```

Return value:

0: No error

-1: Handle parameter of the board is wrong

-2: The selected module number is wrong

-3: The selected module is no "PWM" module

-4: PWM selection is wrong

-5: PWM not initialised. See function "i_APCI1710_InitPWM"

-6: Selection of the PWM start level is wrong

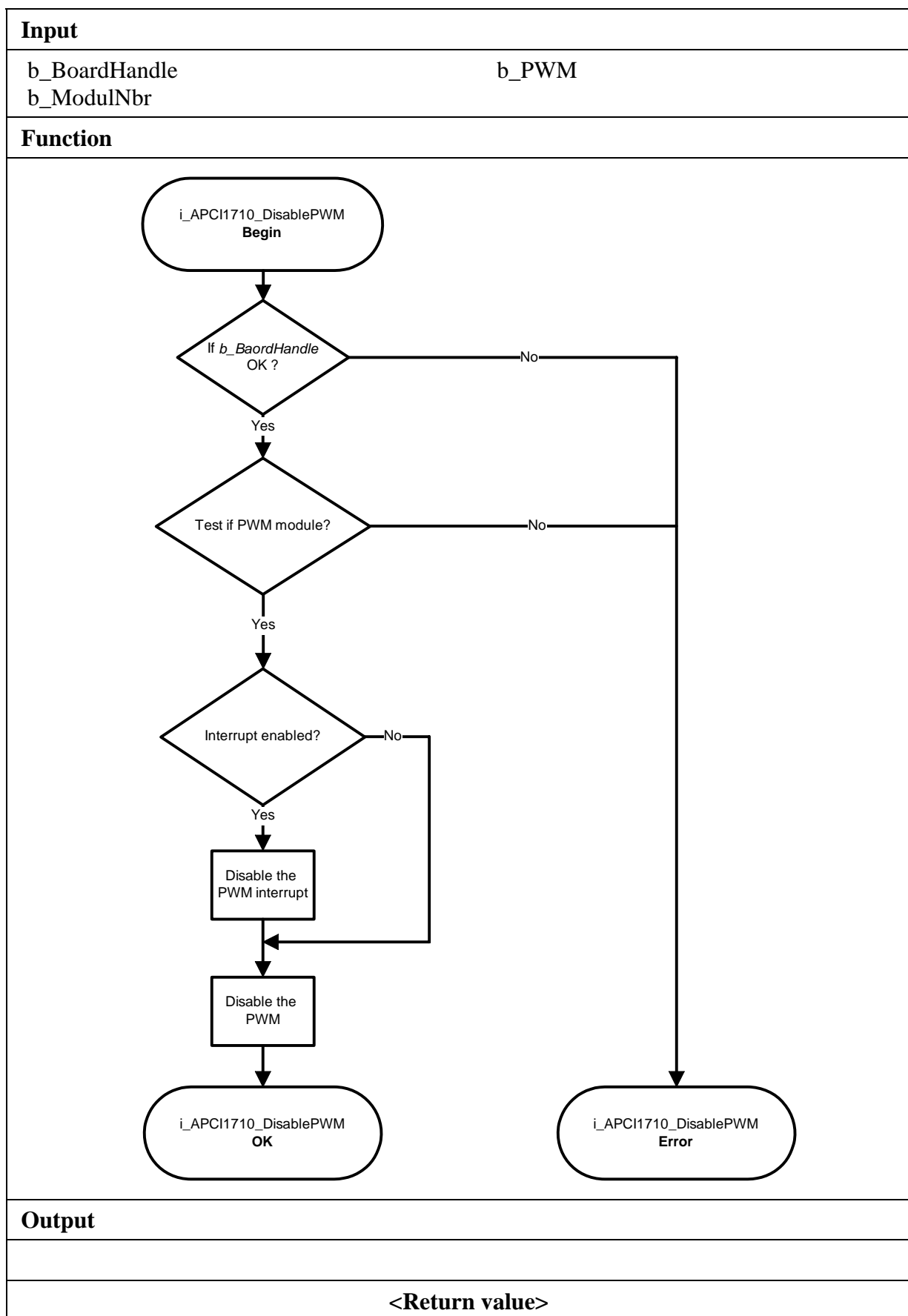
-7: Selection of the PWM Stopp mode is wrong

-8: Selection of the PWM Stopp level is wrong

-9: Selection of the external gate signal is wrong

-10: Interrupt parameter is wrong

-11: Interrupt function is not initialised



5) i_APCI1710_GetPWMInitialisation (...)

Syntax:

<Return Wert> = i_APCI1710_GetPWMInitialisation
 (BYTE b_BoardHandle,
 BYTE b_ModulNbr,
 BYTE b_PWM,
 PBYTE pb_TimingUnit,
 PULONG pul_LowTiming,
 PULONG pul_HighTiming,
 PBYTE pb_StartLevel,
 PBYTE pb_StopMode,
 PBYTE pb_StopLevel,
 PBYTE pb_ExternGate,
 PBYTE pb_InterruptEnable,
 PBYTE pb_Enable)

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_PWM	Selected PWM generator (0 or 1)

-Output:

PBYTE	pb_TimingUnit	Base timing Unit (0 to 4) 0: ns 1: µs 2: ms 3: s 4: mn
PULONG	pul_LowTiming	Low base timing value. See base timing value description table
PULONG	pul_HighTiming	High base timing value. See base timing value description table
PBYTE	pb_StartLevel	Start period level selection 0: The period starts with a low level 1: The period starts with a high level
PBYTE	pb_StopMode	Stop mode selection 0: The PWM is stopped directly after the function "i_APCI1710_DisablePWM" and stops the current period 1: After the function "i_APCI1710_DisablePWM" the PWM is stopped at the end of the current period
PBYTE	pb_StopLevel	Stop PWM level selection 0: The output signal keeps the level after the function "i_APCI1710_DisablePWM" 1: The output signal is set to low after the function "i_APCI1710_DisablePWM" 2: The output signal is set to high after the function "i_APCI1710_DisablePWM"

PBYTE	pb_ExternGate	External gate action selection 0: External gate signal not used. 1: External gate signal used.
PBYTE	pb_InterruptEnable	Aktiviert or Deaktiviert the PWM interrupt. APCI1710_ENABLE: Aktiviert the PWM interrupt. An interrupt occurs after each period APCI1710_DISABLE: Deaktiviert the PWM interrupt
PBYTE	pb_Enable	Indicate if the PWM is enabled or not 0: PWM not enabled 1: PWM enabled

Task:

Returns the PWM (*b_PWM*) initialisation of the selected module (*b_ModulNbr*).
First call the function " i_APCI1710_InitPWM" before you call this function.

Calling convention:ANSI C:

```

unsigned char    b_TimingUnit;
unsigned long    ul_LowTiming;
unsigned long    ul_HighTiming;
unsigned char    b_StartLevel;
unsigned char    b_StopMode;
unsigned char    b_StopLevel;
unsigned char    b_ExternGate;
unsigned char    b_InterruptEnable;
unsigned char    b_Enable;

```

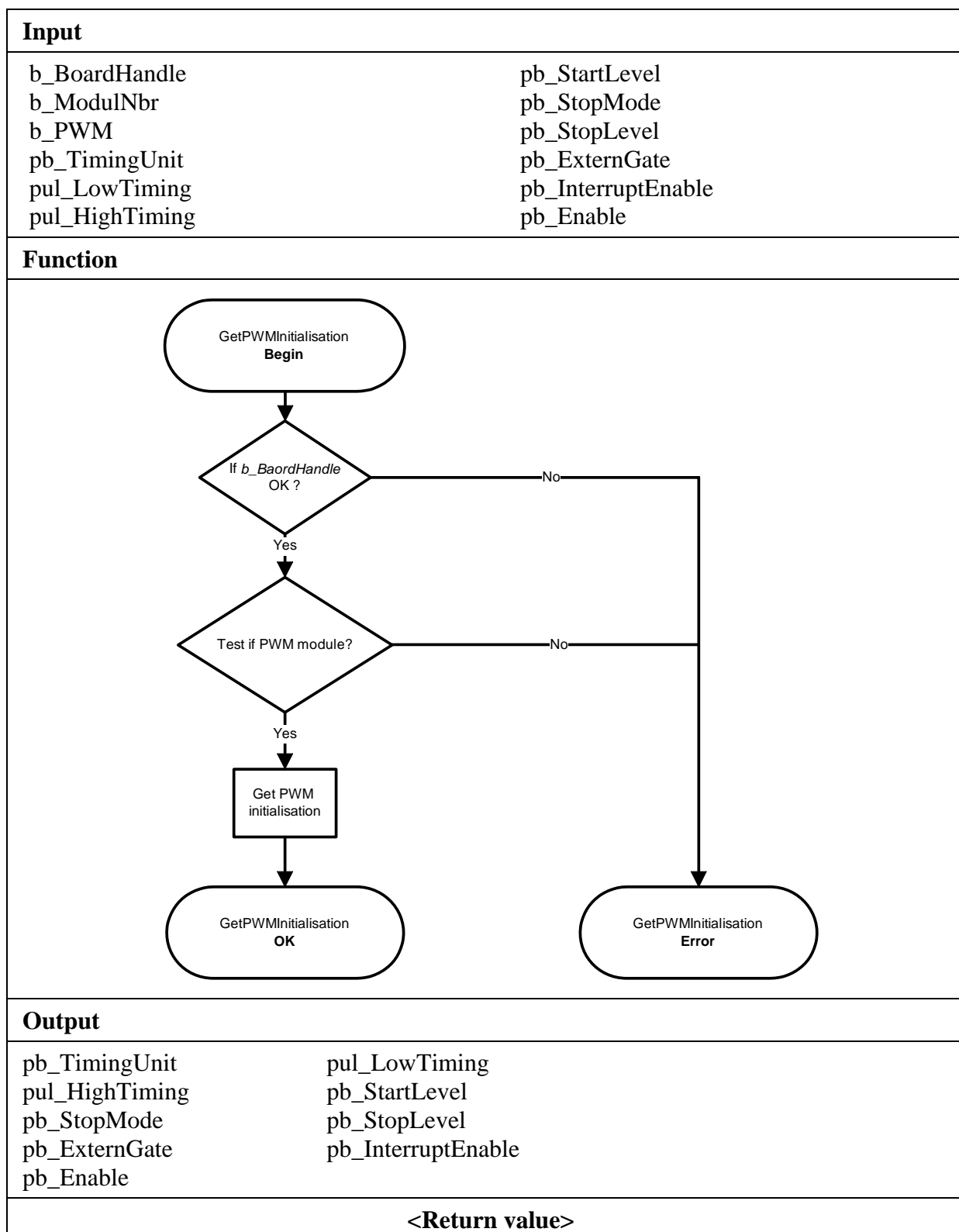
```

i_ReturnValue = i_APCI1710_GetPWMInitialisation
                (b_BoardHandle,
                 0,
                 0,
                 &b_TimingUnit,
                 &ul_LowTiming,
                 &ul_HighTiming,
                 &b_StartLevel,
                 &b_StopMode,
                 &b_StopLevel,
                 &b_ExternGate,
                 &b_InterruptEnable,
                 &b_Enable);

```

Return value:

- 0: No error
- 1: Handle parameter of the board is wrong
- 2: The selected module number is wrong
- 3: The selected module is no "PWM" module
- 4: PWM selection is wrong
- 5: PWM is not initialised. See function "i_APCI1710_InitPWM"



3.3.2 PWM status

6) i_APCI1710_GetPWMStatus (...)

Syntax:

```
<Return Wert> = i_APCI1710_GetPWMStatus
                                (BYTE      b_BoardHandle,
                                 BYTE      b_ModulNbr,
                                 BYTE      b_PWM,
                                 PBYTE     pb_PWMOutputStatus,
                                 PBYTE     pb_ExternGateStatus)
```

Parameter:
-Input:

BYTE	b_BoardHandle	Handle of the board APCI-/CPCI_1710
BYTE	b_PWM	Selected PWM generator (0 or 1)
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)

-Output:

PBYTE	pb_PWMOutputStatus	Status of the PWM output. 0: PWM Output level is low 1: PWM Output level is high.
PBYTE	pb_ExternGateStatus	Status of the external gate. 0: External gate is low. 1: External gate is high.

Task:

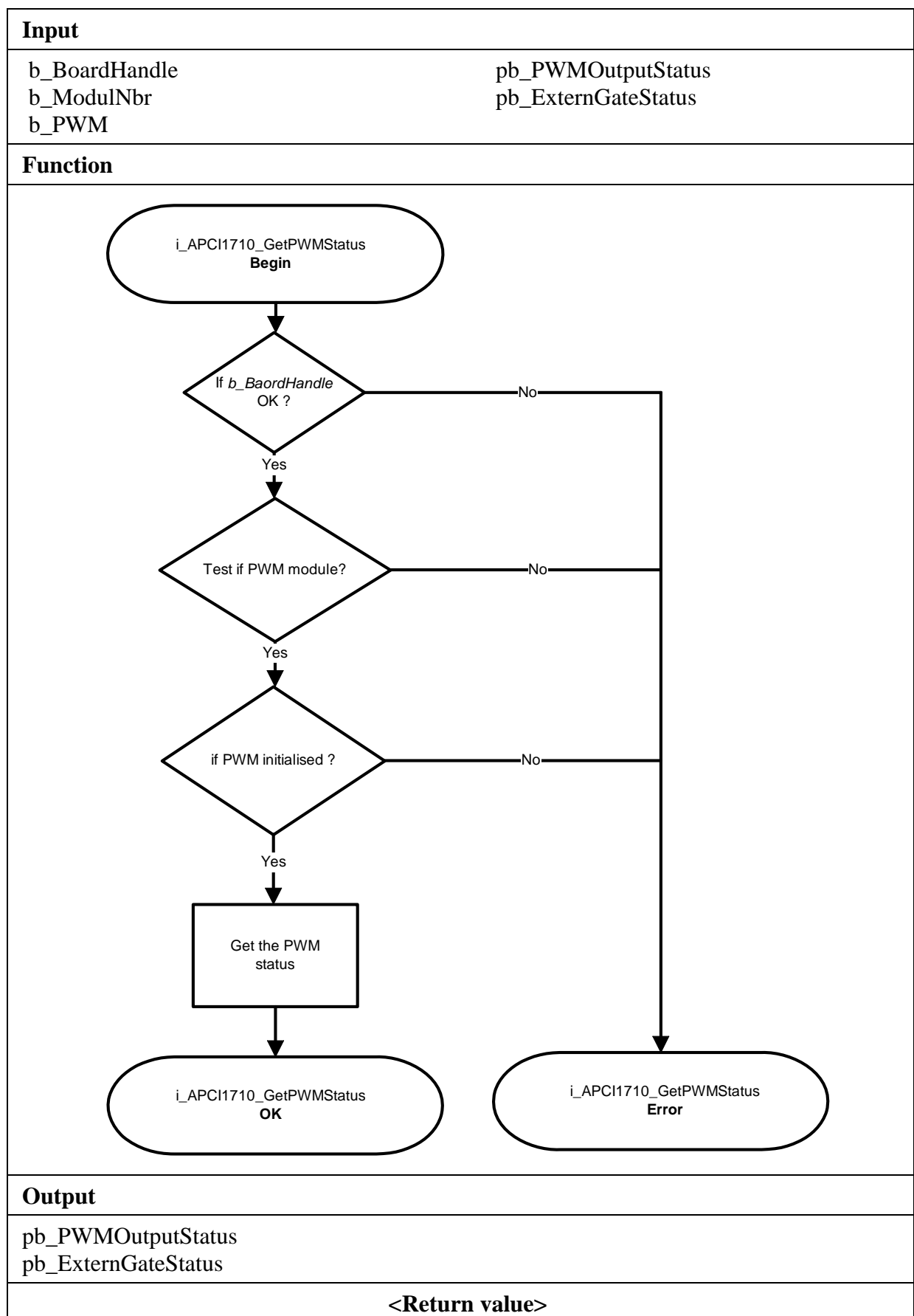
Returns the status of the PWM (*b_PWM*) of the module (*b_ModulNbr*).

Calling convention:
ANSI C:

```
int      i_ReturnValue;
unsigned char b_BoardHandle;
unsigned char b_PWMOutputStatus;
unsigned char b_ExternGateStatus;
i_ReturnValue = i_APCI1710_GetPWMStatus
                (b_BoardHandle,
                 0,
                 0,
                 &b_PWMOutputStatus,
                 &b_ExternGateStatus);
```

Return value:

0: No error
-1: Handle parameter of the board is wrong
-2: The selected module number is wrong
-3: The selected module is no "PWM" module.
-4: PWM selection is wrong
-5: PWM is not initialised. See function "i_APCI1710_InitPWM"
-6: PWM is not enabled. See function "i_APCI1710_EnablePWM"



3.3.3 Windows NT/95 interrupt kernel functions



IMPORTANT!

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

1) i_APCI1710_KRNL_GetPWMStatus (...)

Syntax:

```
<Return Wert> = i_APCI1710_KRNL_GetPWMStatus
                                (UINT    ui_BaseAddress,
                                 BYTE     b_ModulNbr,
                                 BYTE     b_PWM,
                                 PBYTE    pb_PWMOutputStatus,
                                 PBYTE    pb_ExternGateStatus)
```

Parameter:

-Input:

UINT	ui_BaseAddress	Base address of the APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_PWM	Selectd PWM generator (0 or 1)

-Output:

PBYTE	pb_PWMOutputStatus	Status of the PWM output. 0: PWM output level is Low 1: PWM output level is High.
PBYTE	pb_ExternGateStatus	Status of the external gate. 0: External gate is low. 1: External gate is high.

Task:

Returns the status of the selected PWM (*b_PWM*) of the selected module (*b_ModulNbr*).

Calling convention:

ANSI C:

```
int          i_ReturnValue;
unsigned int  ui_BaseAddress;
unsigned char b_PWMOutputStatus;
unsigned char b_ExternGateStatus;
i_ReturnValue = i_APCI1710_KRNL_GetPWMStatus
                (ui_BaseAddress,
                 0,
                 0,
                 &b_PWMOutputStatus,
                 &b_ExternGateStatus);
```

Return value:

0: No error

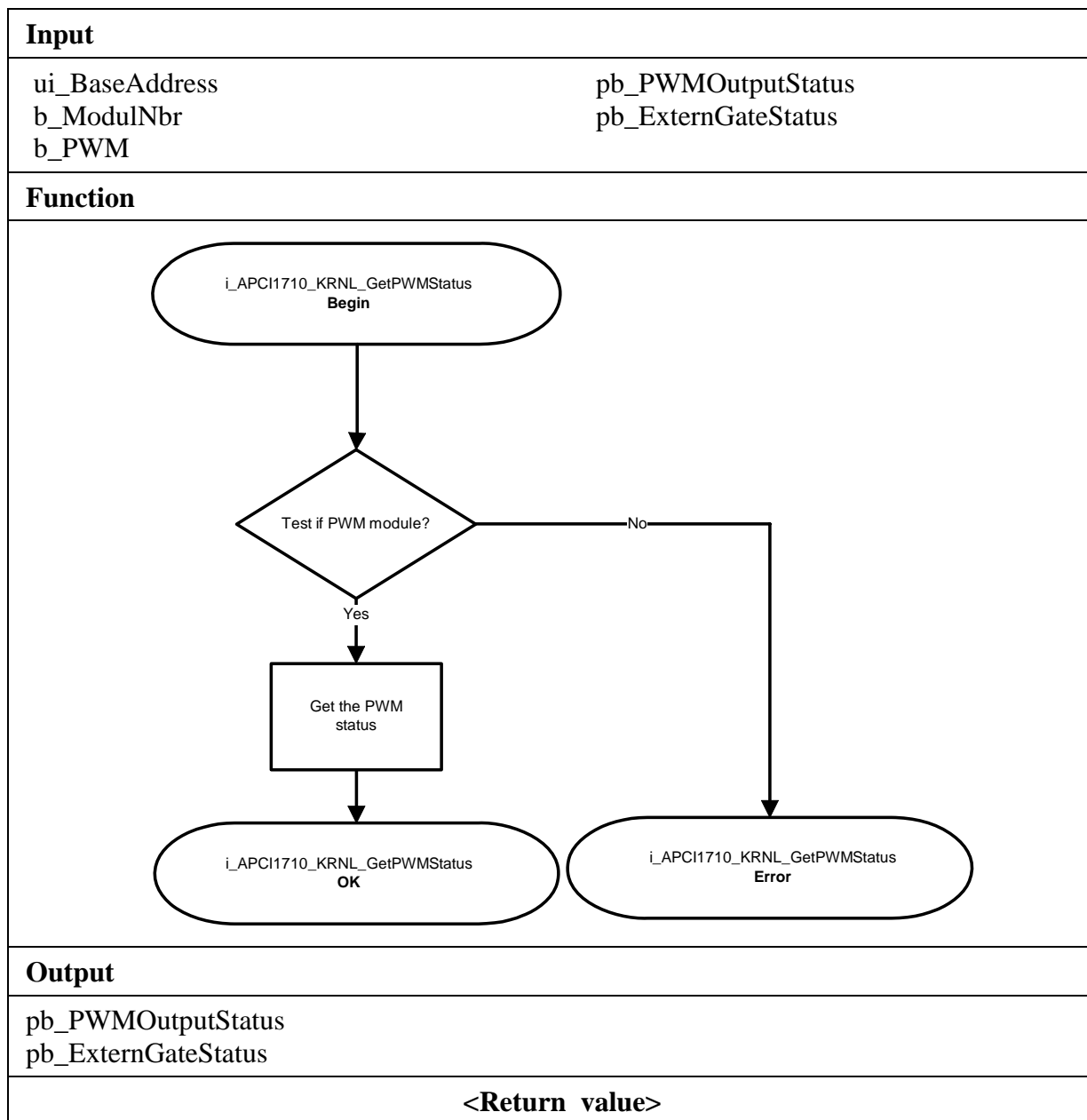
-1: Module selection wrong

-2: The module is not a PWM module

-3: PWM selection is wrong

-4: PWM not initialised. See function _APCI1710_InitPWM"

-5: PWM not enabled. See function "i_APCI1710_EnablePWM"



2) i_APCI1710_KRNL_SetNewPWMTiming (...)

Syntax:

```
<Return Wert> = i_APCI1710_KRBL_InitPWM
                (UINT      ui_BaseAddress,
                 BYTE      b_ModulNbr,
                 BYTE      b_PWM,
                 BYTE      b_ClockSelection,
                 BYTE      b_TimingUnit,
                 ULONG      ul_LowTiming,
                 ULONG      ul_HighTiming)
```

Parameter:

-Input:

UINT	ui_BaseAddress	Base address of the APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_PWM	Selected PWM generator (0 or 1).
BYTE	b_ClockSelection	Selection of the clock signal - APCI1710_30MHZ: The board uses a PCI bus clock of 30 MHz - APCI1710_33MHZ: The board uses a PCI bus clock of 33 MHz - APCI1710_40 MHz: The board uses a 40 MHz quartz clock.
BYTE	b_TimingUnit	Selection of the base unit (0 to 4) 0: ns 1: µs 2: ms 3: s 4: mn
ULONG	ul_LowTiming	Time of the Low level. See Table 3-1
ULONG	ul_HighTiming	Time of the High level. See Table 3-1

-Output:

There is no output

Task:

Sets a new time period: *ul_LowTiming*, *ul_HighTiming* and *ul_TimingUnit* determine the LOW/High time base for the period.

Calling convention:ANSI C:

```
int          i_ReturnValue;  
unsigned int  ui_BaseAddress;  
i_ReturnValue = i_APCI1710_InitPWM  
                (ui_BaseAddress,  
                0,  
                0,  
                APCI1710_40MHZ,  
                1,  
                100,  
                500);
```

Return value:

0: No error

-1: Handle parameter of the board is wrong

-2: The selected module number is wrong

-3: The selected module is no "PWM" module

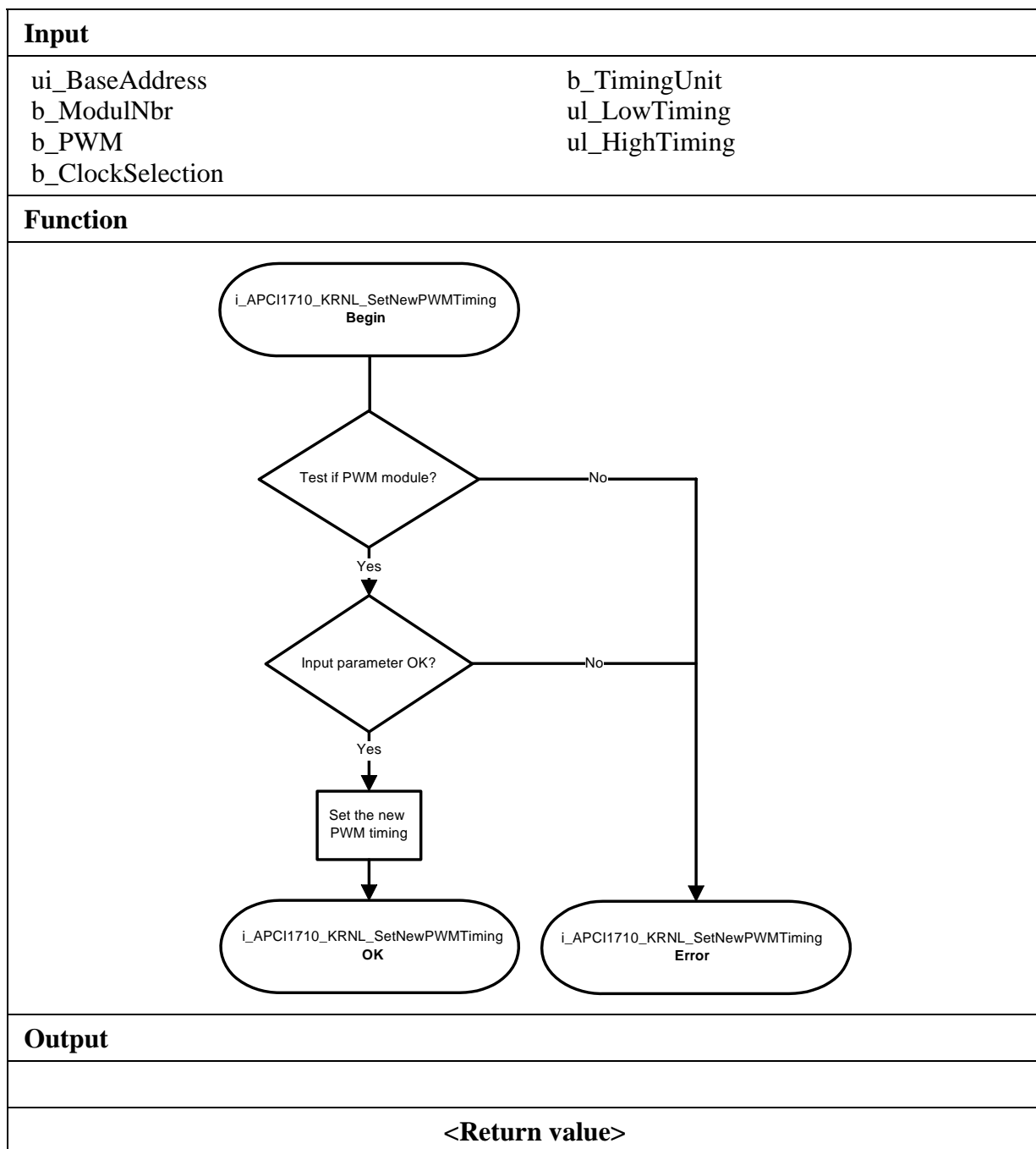
-4: PWM selection is wrong

-5: PWM not initialised. See function_APCI1710_InitPWM"

-6: The selected time unit is wrong

-7: Selected base time "Low" is wrong

-8: Selected base time "High" is wrong



3.3.4 Digital output

1) i_APCI1710_EnableDisablePWMDigitalOutputManualSetting (...)

Syntax:

<Return value> = i_APCI1710_EnableDisablePWMDigitalOutputManualSetting
 (BYTE b_BoardHandle,
 BYTE b_ModulNbr,
 BYTE b_Flag)

Parameter:

-Input:

BYTE	b_BoardHandle	Handle of the board APCI-/CPCI-1710
BYTE	b_PWM	Selected PWM generator (0 or 1)
BYTE	b_Flag	APCI1710_DISABLE : Deactivate the manual management of output H. This is the default state. The output is controlled parallel to output PWM0. APCI1710_ENABLE : The output H is controlled manually through the functions i_APCI1710_SetPWMDigitalOutputOn and i_APCI1710_SetPWMDigitalOutputOff

-Output:

There is no output

Task:

Defines the working method of digital output H.

Callin convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_EnableDisablePWMDigitalOutputManualSetting
                (b_BoardHandle,
                 0,
                 APCI1710_ENABLE);
```

Return value:

0: No error

- 1: Handle parameter of the board is wrong
- 2: The selected module number is wrong
- 3: The selected module is no "PWM" module.
- 4: Firmware Revision error. Version is below 1.2
- 5: Working method selection parameter is wrong

2) i_APCI1710_SetPWMDigitalOutputOn (...)

Syntax:

```
<Return value> = i_APCI1710_SetPWMDigitalOutputOn
                    (BYTE      b_BoardHandle,
                     BYTE      b_ModulNbr)
```

Parameter:

- Input:

BYTE	b_BoardHandle	Handle of the board
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)

- Output

There is no output.

Task:

Switches on the digital output H of the selected PWM module (b_ModulNbr).

Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_SetPWMDigitalOutputOn
                (b_BoardHandle,
                 0);
```

Return value:

0: No error

- 1: Handle parameter of the board is wrong
- 2: The selected module number is wrong
- 3: The selected module is no "PWM" module.
- 4: Firmware revision error Version is below 1.2
- 5: Manual control of output H was not activated. See function
i_APCI1710_EnableDisablePWMDigitalOutputManualSetting

3) i_APCI1710_SetPWMDigitalOutputOff (...)

Syntax:

```
<Return value> = i_APCI1710_SetPWMDigitalOutputOff
                    (BYTE      b_BoardHandle,
                     BYTE      b_ModulNbr)
```

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)

- Output

There is no output

Task:

Switches off the digital output of the selected PWM module (b_ModulNbr).

Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_SetPWMDigitalOutputOff
                    (b_BoardHandle,
                     0);
```

Return value:

0: No error

-1: Handle parameter of the board is wrong

-2: The selected module number is wrong

-3: The selected module is no "PWM" module

-4: Firmware revision error. Version is below 1.2

-5: Manual control of output H was not activated. See function
i_APCI1710_EnableDisablePWMDigitalOutputManualSetting

3.3.5 Digital inputs

1) i_APCI1710_ReadPWM1DigitalInput (...)

Syntax:

```
<Return value> = i_APCI1710_ReadPWM1DigitalInput
                    (BYTE      b_BoardHandle,
                     BYTE      b_ModulNbr,
                     BYTE      b_InputChannel,
                     PBYTE     pb_ChannelStatus)
```

Parameter:
- Input

BYTE	b_BoardHandle	Handle of the APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_InputChannel	Selection of the digital inputs (0 to 2). 0: input E 1: input F 2: input G

-Output:

PBYTE	pb_ChannelStatus	Status of the digital input. 0: not active 1: input is active
-------	------------------	---

Task:

Returns the status of the selected digitalen input to the PWM module.
Variable *b_InputChannel* and module *b_ModulNbr*.

Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_ReadPWM1DigitalInput
                (b_BoardHandle,
                 0,
                 0,
                 &b_ChannelStatus);
```

Return value:

0: No error
-1: Handle parameter of the board is wrong
-2: The selected module number is wrong
-3: The selected module is no "PWM" module.
-4: Firmware revision error. Version is below 1.2
-5: Selection of the digital PWM input is wrong

2) i_APCI1710_ReadPWMAIldigitalInput (...)

Syntax:

```
<Return value> = i_APCI1710_ReadPWMAIldigitalInput
                    (BYTE      b_BoardHandle,
                     BYTE      b_ModulNbr,
                     PBYTE     pb_InputStatus)
```

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)

- Output

PBYTE	pb_InputStatus	Status of the digital inputs.
-------	----------------	-------------------------------

Task:

Returns the status of all digital PWM inputs. (*b_ModulNbr*).

D2	D1	D0
INPUT 2 (G)	INPUT 1 (F)	INPUT 0 (E)

1: Input is active

0: Input is not active

Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_ReadPWMAIldigitalInput
                (b_BoardHandle,
                 0,
                 &b_InputStatus);
```

Return value:

0: No error

-1: Handle parameter of the board is wrong

-2: The selected module number is wrong

-3: The selected module is no "PWM" module.

-4: Firmware revision error. Version is below 1.2