

Function Description

Counter/Timer

APCLe-1711 and APCI-/CPCI-1710
Multifunction counter board



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 product by the user or improper use, for example, if the product 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 product 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 products.

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 product 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, APCI-1500, MSX-Box and MSX-E are registered trademarks of ADDI-DATA GmbH.
- Turbo Pascal, Delphi, Borland C, Borland C++ are registered trademarks of Borland Software Corporation.
- Microsoft .NET, Microsoft C, Visual C++, MS-DOS, Windows 95, Windows 98, Windows 2000, Windows NT, Windows EmbeddedNT, Windows XP, Windows Vista, Windows 7, Windows Server 2000, Windows Server 2003, Windows Embedded and Internet Explorer are registered trademarks of Microsoft Corporation.
- LabVIEW, LabWindows/CVI, DASyLab, DIAdem are registered trademarks of National Instruments Corporation.
- CompactPCI is a registered trademark of PCI Industrial Computer Manufacturers Group.
- VxWorks is a registered trademark of Wind River Systems, Inc.
- RTX is a registered trademark of IntervalZero.

Warning

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



Personal injury



Damage to the board, the PC and peripherals



Pollution of the environment

- Protect yourself, others and the environment!

- Read the safety precautions (yellow leaflet) carefully!

If this leaflet is not enclosed 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.

- Pay attention to the following symbols:



IMPORTANT!

Designates hints and other useful information.



WARNING!

Designates a possibly dangerous situation.

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



WARNING!

Designates a possibly dangerous situation.

If the instructions are ignored, the board, the PC and/or peripherals may be **destroyed** and persons may be **endangered**.

Contents

Warning	3
Chapter overview	5
1 Function description	6
1.1 Block diagram	7
1.2 Used signals.....	8
1.3 Pin assignment: Function module.....	9
1.4 Connection example	11
1.5 I/O address assignment.....	12
1.6 I/O functions.....	13
1.6.1 Mode description	13
1.6.2 Timer	14
1.7 Procedure for using the counter/timer.....	16
2 Software functions	17
2.1 Interrupt mask.....	18
2.2 Initialisation.....	19
2.2.2 Reading the timer.....	28
2.2.3 Writing to the timer	40
2.3 Functions in kernel mode.....	43
2.3.1 Reading the timer.....	43
2.3.2 Writing to the timer	48
3 Appendix	51
3.1 Index	51
4 Contact and support	52

Figures

Fig. 1-1: Block diagram: "Counter/Timer" functionality	7
Fig. 1-2: Pin assignment: 50-pin D-Sub male connector	9
Fig. 1-3: Pin assignment: 78-pin D-Sub female connector (only APCIe-1711).....	10
Fig. 1-4: Connection example: Function modules.....	11

Tables

Table 1-1: Used signals	8
Table 1-2: I/O address assignment.....	12
Table 2-1: Define value	17
Table 2-2: Interrupt mask	18
Table 2-3: Counter/timer modes.....	20
Table 2-4: Selection of the input clock	20

Chapter overview

In this manual, you will find the following information:

Chapter	Content
1	Function description including a block diagram and a connection example
2	Software functions for the "Counter/Timer" functionality
3	Appendix with index
4	Contact and support address

This document solely describes the functionality "Counter/Timer".

For general information on the **APCIe-1711** or **APCI-/CPCI-1710**, please read the "Technical Description" of the board. It contains, for example, the chapter "Inserting and installing the board" that supports you in commissioning.

1 Function description

The counter/timer functionality is a programmable interval counter/timer, which is comparable to the Intel 82C54.

Each function module supports three individual 32-bit counters/timers, which can be read out or written on via the data bus, as well as a function and control logic.

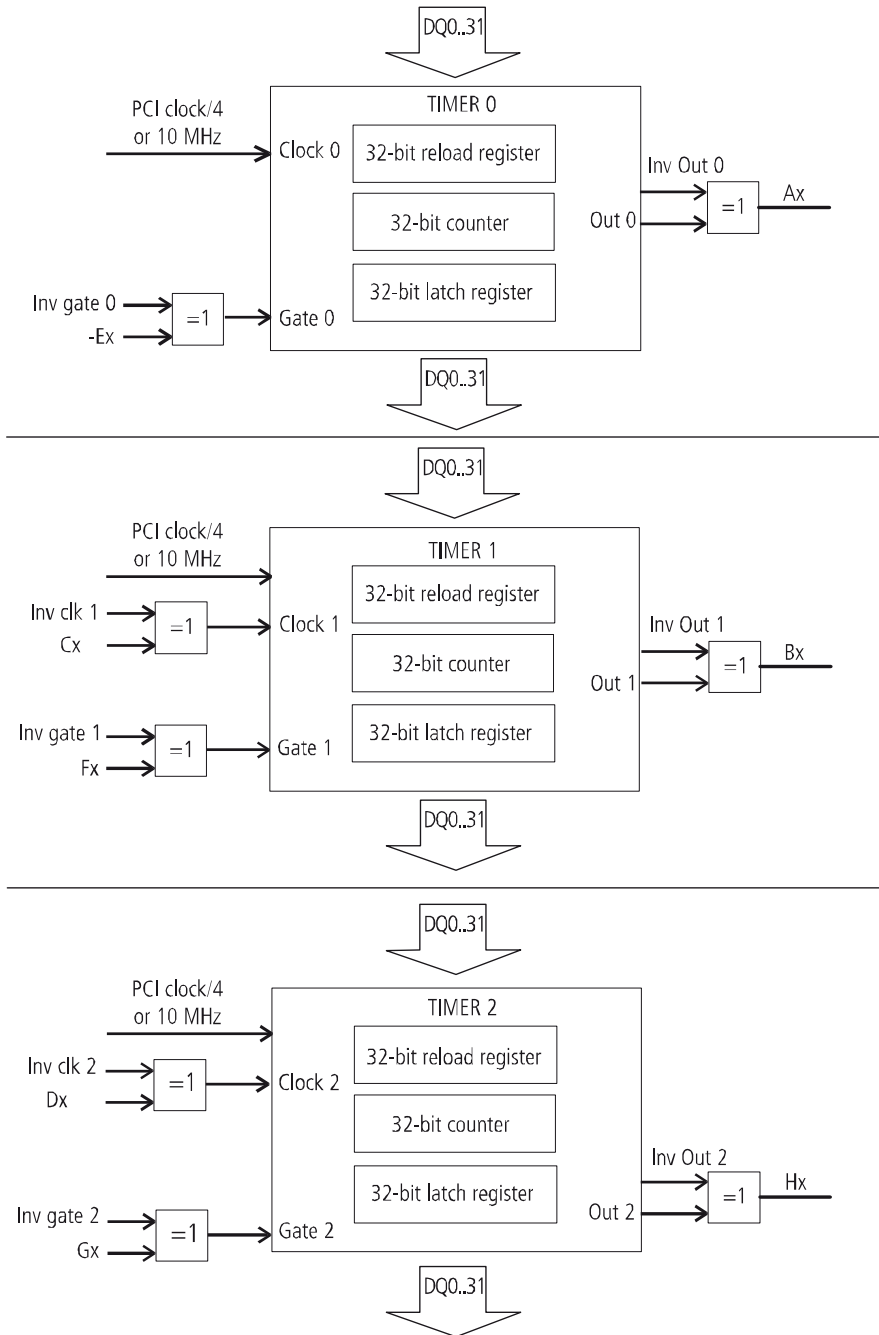
Delay times are generated by the software control. It is possible for the user to program the desired delay time for the function module instead of setting time loops in the software. After the delay time has elapsed, an interrupt can be generated.

Features:

- 4 function modules with three 32-bit counters/timers each (only binary numbers)
- Optical isolation of the inputs and outputs through opto-couplers to prevent ground loops
- Processing of up to 5 MHz signals
- 6 programmable modes
- Status read-back und latch command
- Inputs and outputs invertible through software
- Hardware and software gates possible, can be read back
- Single interface: no multiple assignment of addresses
- Interrupt triggered by an individual release bit per counter/timer and an interrupt status register
- Available clock: PCI bus clock divided by 4 (only **PCI-1710**) or 10 MHz from the quartz oscillator on the board, can be selected through software

1.1 Block diagram

Fig. 1-1: Block diagram: “Counter/Timer” functionality



1.2 Used signals

The counter/timer functionality uses five inputs (channels C to G) and three outputs (channels A, B, H) of the corresponding function module of the board.

Table 1-1: Used signals

Signal	At the connector	Polarity	Function
OUT 1	Ax +/-	differential/TTL	Output of counter/timer 0
OUT 2	Bx +/-	differential/TTL	Output of counter/timer 1
OUT 3	Hx	24 V / optional 5 V	Output of counter/timer 2
GATE 1	Ex	24 V / optional 5 V	Gate input of counter/timer 0
GATE 2	Fx	24 V / optional 5 V	Gate input of counter/timer 1
GATE 3	Gx	24 V / optional 5 V	Gate input of counter/timer 2
CLK 1	-	-	used by internal clock
CLK 2	Cx +/-	differential/TTL/ optional 24 V	Clock/counter input of counter/timer 1
CLK 3	Dx +/-	differential/TTL/ optional 24 V	Clock/counter input of counter/timer 2

x = Number of the function module

1.3 Pin assignment: Function module



IMPORTANT!

Depending on hardware or software, the function modules are numbered differently.

As to the pin assignment, the modules are numbered from 1 to 4. In the program **SET1711** or **SET1710** and in the software functions, module numbering starts with 0.

Fig. 1-2: Pin assignment: 50-pin D-Sub male connector

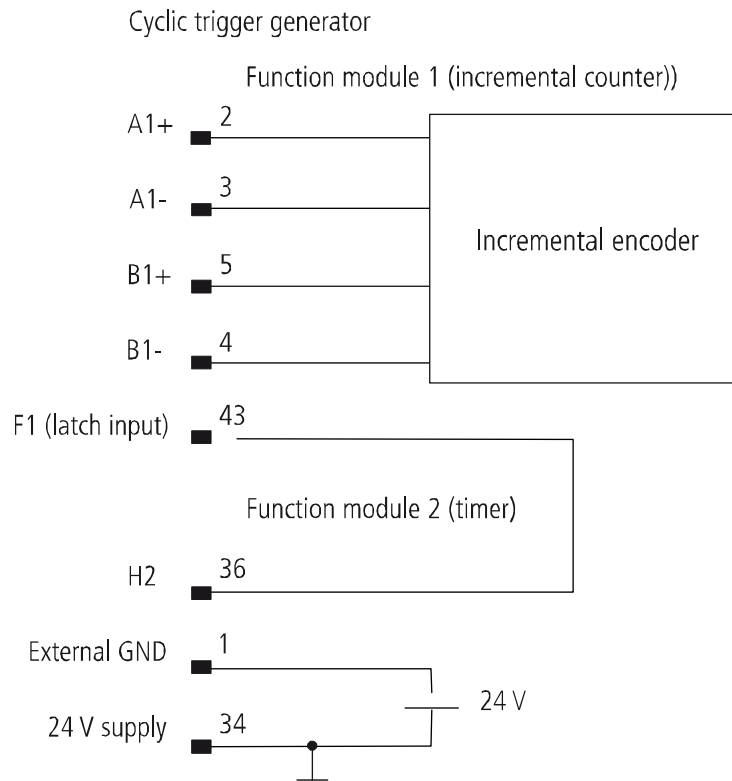
Pin		Pin		Pin		Pin	
34	U _{Ref} /+24 V supply			34	18	1	GND
35	FM1: OUT 3	Function module 3 (FM3)	18	FM3: OUT 1+	35	2	FM1: OUT 1+
36	FM2: OUT 3		19	FM3: OUT 1-	36	3	FM1: OUT 1-
37	FM3: OUT 3		20	FM3: OUT 2+	37	4	FM1: OUT 2+
38	FM4: OUT 3		21	FM3: OUT 2-	38	5	FM1: OUT 2-
39	FM1: GATE 1		22	FM3: CLK 2+	39	6	FM1: CLK 2+
40	FM2: GATE 1		23	FM3: CLK 2-	40	7	FM1: CLK 2-
41	FM3: GATE 1		24	FM3: CLK 3+	41	8	FM1: CLK 3+
42	FM4: GATE 1		25	FM3: CLK 3-	42	9	FM1: CLK 3-
43	FM1: GATE 2		26	FM4: OUT 1+	43	10	FM2: OUT 1+
44	FM2: GATE 2	Function module 4 (FM4)	27	FM4: OUT 1-	44	11	FM2: OUT 1-
45	FM3: GATE 2		28	FM4: OUT 2+	45	12	FM2: OUT 2+
46	FM4: GATE 2		29	FM4: OUT 2-	46	13	FM2: OUT 2-
47	FM1: GATE 3		30	FM4: CLK 2+	47	14	FM2: CLK 2+
48	FM2: GATE 3		31	FM4: CLK 2-	48	15	FM2: CLK 2-
49	FM3: GATE 3		32	FM4: CLK 3+	49	16	FM2: CLK 3+
50	FM4: GATE 3		33	FM4: CLK 3-	50	17	FM2: CLK 3-

This pin assignment also applies to the **APCIe-1711** if the cable **ST1711-50** is connected to the 78-pin D-Sub female connector of the board.
For further information on this, please refer to the "Technical Description" of the board **APCIe-1711**.

1.4 Connection example

The counter/timer functionality is implemented on all function modules.

Fig. 1-4: Connection example: Function modules



1.5 I/O address assignment

Table 1-2: I/O address assignment

			D31...D24	D23...D16	D15...D8	D7...D0
Bytes	Rd	Wr	HIGHBYTE	MIDHIGHBYTE	MIDLOWBYTE	LOWBYTE
BASEx +0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	TIMER0	TIMER0	TIMER0	TIMER0
BASEx +4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	TIMER1	TIMER1	TIMER1	TIMER1
BASEx +8	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	TIMER2	TIMER2	TIMER2	TIMER2
BASEx +12		<input checked="" type="checkbox"/>	GLOBAL CONTROL REGISTER			
BASEx +16	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0x00h	0x00h	TIMER0 CONTROL REG	TIMER0 CONTROL REG
BASEx +20	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0x00h	0x00h	TIMER1 CONTROL REG	TIMER1 CONTROL REG
BASEx +24	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0x00h	0x00h	TIMER2 CONTROL REG	TIMER2 CONTROL REG
BASEx +28						
BASEx +32		<input checked="" type="checkbox"/>				SET TIMER 0 REGISTER
BASEx +36		<input checked="" type="checkbox"/>				SET TIMER 1 REGISTER
BASEx +40		<input checked="" type="checkbox"/>				SET TIMER 2 REGISTER
BASEx +44		<input checked="" type="checkbox"/>				SET TIMER 0 SOFTGATE
BASEx +48		<input checked="" type="checkbox"/>				SET TIMER 1 SOFTGATE
BASEx +52		<input checked="" type="checkbox"/>				SET TIMER 2 SOFTGATE
BASEx +60			FUNKNBR2	FUNKNBR1	REVBYTE2	REVBYTE1

x = Number of the function module

The counter/timer functionality uses 15 DWORDS in the I/O range of the function module x. Access is always read or written in 32-bit width.

1.6 I/O functions

To program the individual counters/timers, six modes are available. The counter/timer can be loaded with a new value and be read at any time. Before it is read though, the value has to be latched.

1.6.1 Mode description

a) Mode 0: Interrupt at end of sequence

Mode 0 is used for counting events.

After the initialisation, the output is set to "Low". When the counter has reached the value 0, the output is set to "High". It keeps this position until a new counting sequence starts or a new counter value is written.

b) Mode 1: Hardware-retriggerable monoflop

In this mode, the GATE input is used to trigger the timer instead of activating or deactivating it. Except for this, this mode corresponds to Mode 0.

c) Mode 2: Pulse generator

In this mode, the counter divides the selected input clock by the start value "ul_ReloadValue".

Mode 2 is suited to generate a real-time clock interrupt.

After the initialisation, the output is set to "High". When the counter has reached the value 1, the output is set to "Low". After just one clock pulse, it is reset to "High". The counter reloads the start value ("ul_ReloadValue") and the counting sequence is rerun. The number of sequences is infinite. An interrupt can always be generated at the end of a sequence.

Time computation: $(ul_ReloadValue + 2) \times input\ clock$

d) Mode 3: Square wave signal generator

In Mode 3, the baud rate is generated. This mode differs from Mode 2 only as far as the output sequence is concerned.

After the initialisation, the output is set to "High". When the counter has run down to the half of the start value, the output is set to "Low". It keeps this position until a new counting sequence starts. The number of sequences is infinite.

Time computation: $(ul_ReloadValue + 2) \times input\ clock$

e) Mode 4: Software-triggered strobe

After the initialisation, the output is set to "High". When the counter has run down, the output is set to "Low". After just one clock pulse, it is reset to "High".

The counting sequence is triggered by writing the start value. If a new value is written during the counting sequence, this value is loaded as the new start value at the next clock pulse.

f) Mode 5: Hardware-triggered strobe (retriggerable)

In this mode, the GATE input is used to trigger the timer instead of activating or deactivating it. Except for this, this mode corresponds to Mode 4.

1.6.2 Timer

a) TIMER 0 register (base +0)

The TIMER0 register is a 32-bit register to which the reload value of counter/timer 0 is written. When reading this address, the current latched value of counter/timer 0 is read.

b) TIMER 1 register (base +4)

The TIMER0 register is a 32-bit register to which the reload value of counter/timer 1 is written. When reading this address, the current latched value of counter/timer 1 is read.

c) TIMER 2 register (base +8)

The TIMER0 register is a 32-bit register to which the reload value of counter/timer 2 is written. When reading this address, the current latched value of counter/timer 2 is read.

d) Control register (base +12)

The control register is a 32-bit register of the three counters.

D0: = 0

D1: Timer 0 selection; if "1", the counter value and the status are latched.

D2: Timer 1 selection; if "1", the counter value and the status are latched.

D3: Timer 2 selection; if "1", the counter value and the status are latched.

D4: Write "0" = Latch the selected timer state

D5: Write "0" = Latch the selected timer state

D6: = 1

D7: = 1

D8 ... 31: No function

The LATCH function enables all of the three counters to be latched simultaneously through software. To do so, all of the three counters have to be selected.

D1= D2= D3 = D6 = D7 =1, & D4 = 0 = Latching the selected counters

e) TIMER 0/1/2 control register (base +16/20/24)

D0: Mode bit 1 (only writing possible); counter mode selection: 0, 1, 2, 3, 4, 5

D1: Mode bit 2 (only writing possible)

D2: Mode bit 3 (only writing possible)

D3: =0

D4: =0

D5: =0

D6: NULL_COUNT (only reading possible); if "1", the timer is stopped, if "0", the timer runs down (has been loaded before)

D7: OUT (only reading possible), output of the current timer output state

D8: GATE (only reading possible), output of the current timer GATE state

D9 ... D31: When reading back, remaining bits are set to 0.

f) SET TIMER 0 register (base +32)

D0: 1: Inversion of the external signal GATE 0 0: No inversion (reset)
 D1: 1: Inversion of the internal signal CLK 0 0: No inversion (reset)
 D2: 1: Inversion of the external signal OUT 0 0: No inversion (reset)
 D3: 1: Interrupt release for TIMER 0 0: No inversion (reset)
 D4 ... D31: No function

g) SET TIMER 1 register (base + 36)

D0: 1: Inversion of the external signal GATE 1 0: No inversion (reset)
 D1: 1: Inversion of the external signal CLK 1 0: No inversion (reset)
 D2: 1: Inversion of the external signal OUT 1 0: No inversion (reset)
 D3: 1: Interrupt release for TIMER 1 0: No interrupt (reset)
 D4: 1: Use of the external signal CLK 1 0: Internal clock
 D5: 0: The PCI bus clock divided by 4 is used as the time base (only **PCI-1710**).
 1: The internal clock (10 MHz) is used as the time base.
 D6 ... D31: No function

h) SET TIMER 2 register (base +40)

D0: 1: Inversion of the external signal GATE 2 0: No inversion (reset)
 D1: 1: Inversion of the external signal CLK 2 0: No inversion (reset)
 D2: 1: Inversion of the external signal OUT 2 0: No inversion (reset)
 D3: 1: Interrupt release for TIMER 2 0: No interrupt (reset)
 D4: 1: Use of the external signal CLK 2 0: Internal clock
 D5: 0: The PCI bus clock divided by 4 is used as the time base (only **PCI-1710**).
 1: The internal clock (10 MHz) is used as the time base.
 D6 ... D31: No function

i) SET TIMER 0/1/2 softgate register (base +44/48/52)

D0: 1: Counting is possible according to the external GATE signal.
 0: Counting is disabled (reset).
 D1 ... D31: No function

j) Version register (base +60)

The function and the revision are detected (read command, ASCII format).
 BASE +60 „I“ „C“ „1“ „3“ = Counter/Timer (Intel Counter) 1.3

1.7 Procedure for using the counter/timer

In order to use the counter/timer, you have to proceed as follows:

- Program the counter/timer in the desired mode.
- Adapt the signals GATE or OUT to the desired level.
- Write the reload value to the timer.

The counter/timer is now ready for use.

If you want to use the interrupt function, you first have to set the release bits for the interrupt to "1".

2 Software functions



IMPORTANT!

Please pay attention to the following notations in the text:

Function: "i_APCI1710_SetBoardInformation"

Variable: "ui_Address".

Table 2-1: Define value

Name of the constant	Decimal value	Hexadecimal value
DLL_COMPILER_C	1	1
DLL_COMPILER_VB	2	2
DLL_COMPILER_PASCAL	3	3
DLL_LABVIEW	4	4
APCI1710_PCI_BUS_CLOCK (only PCI-1710)	0	0
APCI1710_FRONT_CONNECTOR_INPUT (only PCI-1710)	1	1
APCI1710_30MHZ (only PCI-1710)	30	1E
APCI1710_33MHZ (only PCI-1710)	33	21
APCI1710_40MHZ	40	28
APCI1710_10MHZ	10	A

On the board **APCIe-1711**, only the 40 MHz and the 10 MHz clocks are available. If you select 30 MHz or 33 MHz, an error message will be displayed.

2.1 Interrupt mask

Each counter/timer can generate an interrupt. The interrupt and the interrupt routine are activated by the function "i_APCI1710_SetBoardIntRoutineX".

Table 2-2: Interrupt mask

b_ModuleMask	ul_InterruptMask	Meaning
0000 0001	0000 0000 0001 0000	TIMER 0 interrupt triggered by function module 0
0000 0001	0000 0000 0010 0000	TIMER 1 interrupt triggered by function module 0
0000 0001	0000 0000 0100 0000	TIMER 2 interrupt triggered by function module 0
0000 0010	0000 0000 0001 0000	TIMER 0 interrupt triggered by function module 1
0000 0010	0000 0000 0010 0000	TIMER 1 interrupt triggered by function module 1
0000 0010	0000 0000 0100 0000	TIMER 2 interrupt triggered by function module 1
0000 0100	0000 0000 0001 0000	TIMER 0 interrupt triggered by function module 2
0000 0100	0000 0000 0010 0000	TIMER 1 interrupt triggered by function module 2
0000 0100	0000 0000 0100 0000	TIMER 2 interrupt triggered by function module 2
0000 1000	0000 0000 0001 0000	TIMER 0 interrupt triggered by function module 3
0000 1000	0000 0000 0010 0000	TIMER 1 interrupt triggered by function module 3
0000 1000	0000 0000 0100 0000	TIMER 2 interrupt triggered by function module 3

2.2 Initialisation

1) i_APCI1710_InitTimer (...)

Syntax:

<Return value> = i_APCI1710_InitTimer

(BYTE b_BoardHandle,
 BYTE b_ModulNbr,
 BYTE b_TimerNbr,
 BYTE b_TimerMode,
 ULONG ul_ReloadValue,
 BYTE b_InputClockSelection,
 BYTE b_InputClockLevel,
 BYTE b_OutputLevel,
 BYTE b_HardwareGateLevel)

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of the board APCIe-1711 or APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TimerNbr	Number of the timer to be configured (0 to 2)
BYTE	b_TimerMode	Selection of the counter/timer mode (0 to 5) 0: Interrupt at end of sequence 1: Hardware-retriggerable monoflop 2: Pulse generator 3: Square wave signal generator 4: Software-triggered strobe 5: Hardware-triggered strobe (retriggerable)
ULONG	ul_ReloadValue	Counter start value or division factor
BYTE	b_InputClockSelection	Selection of the timer input clock - APCI1710_30MHZ (only PCI-1710) - APCI1710_33MHZ (only PCI-1710) - APCI1710_40MHz - APCI1711_10MHz
BYTE	b_InputClockLevel	Selection of the input clock level 0: Active if "Low" 1: Active if "High" (input inverted)
BYTE	b_OutputLevel	Selection of the output clock level 0: Active if "Low" 1: Active if "High" (output inverted)

BYTE b_HardwareGateLevel Selection of the hardware gate level
 0: Active if "Low" (input inverted)
 1: Active if "High" (Set the parameter to "0" if the external gate is not used!)

- Output:

There is no output.

Table 2-3: Counter/timer modes

Mode No.	Mode	u_ReloadValue	Hardware gate/ Hardware trigger
0	Interrupt at end of sequence	Start value	Hardware gate
1	Hardware-retriggerable monoflop	Start value	Hardware trigger
2	Pulse generator	Division factor	Hardware gate
3	Square wave signal generator	Division factor	Hardware gate
4	Software-triggered strobe	Start value	Hardware gate
5	Hardware-triggered strobe (retriggerable)	Start value	Hardware trigger

Table 2-4: Selection of the input clock

b_InputClockSelection	Description
APCI1710_PCI_BUS_CLOCK (only PCI-1710)	The PCI bus clock divided by 4 is used as the timer input clock. The PCI bus clock may be 30 MHz or 33 MHz. For timer 0, only these clocks and "APCI1710_10MHZ" can be selected.
APCI1710_10MHZ	The 40 MHz clock divided by 4 is used as the timer input clock. PCI-1710: For timer 0, only this clock and "APCI1710_PCI_BUS_CLOCK" can be selected.
APCI1710_FRONT_CONNECTOR_INPUT (only PCI-1710)	At the front connector, an input clock for timers 1 and 2 can be applied. This clock source can overwrite the output clock of timer 0 or all other clock sources.

Task:

Configuration of the timer (b_TimerNbr) and the operating mode (b_TimerMode) of the selected module (b_ModulNbr). This function has to be called before a function that accesses the timer.

Function call:

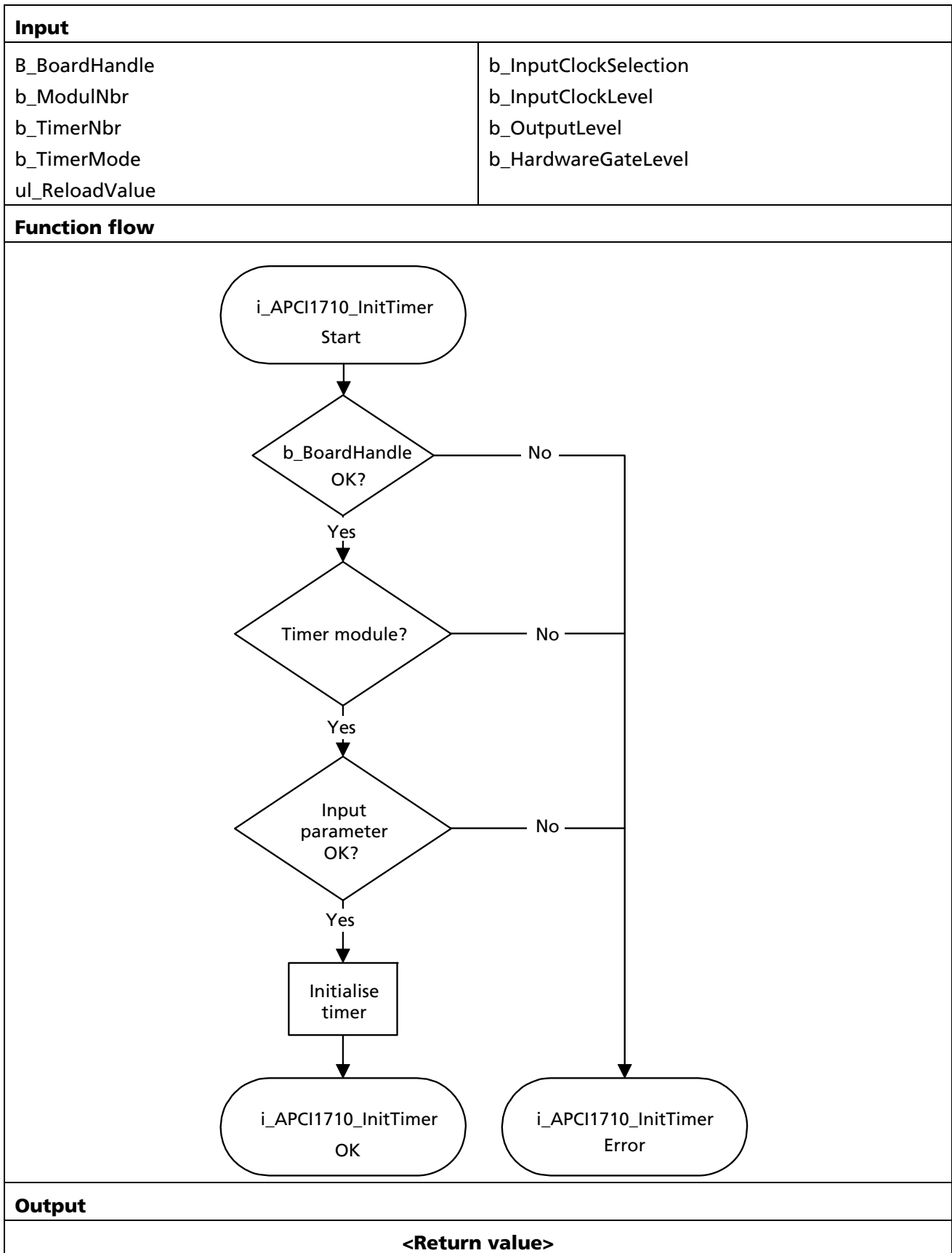
ANSI C:

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

```
i_ReturnValue = i_APCI1710_InitTimer  
                (b_BoardHandle,  
                 0,  
                 0,  
                 2,  
                 0xFF00,  
                 APCI1710_40MHZ (APCIe-1711),  
                 APCI1710_PCI_BUS_CLOCK (PCI-1710),  
                 1,  
                 1,  
                 1);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong.
- 2: The module number is wrong.
- 3: The timer is wrong.
- 4: The module is not a timer module.
- 5: The operating mode is wrong.
- 6: The timer input clock is wrong.
- 7: The input clock level is wrong.
- 8: The output clock level is wrong.
- 9: The hardware gate level is wrong.



2) i_APCI1710_EnableTimer (...)**Syntax:**

```
<Return value> = i_APCI1710_EnableTimer
                                (BYTE  b_BoardHandle,
                                BYTE  b_ModulNbr,
                                BYTE  b_TimerNbr,
                                BYTE  b_InterruptEnable)
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of the board APCIe-1711 or APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TimerNbr	Number of the timer to be released (0 to 2)
BYTE	b_InterruptEnable	Activates or deactivates the timer interrupt. APCI1710_ENABLE: Interrupt activated APCI1710_DISABLE: Interrupt deactivated

- Output:

There is no output.

Task:

Activates the timer (b_TimerNbr) of the selected module (b_ModulNbr). First call the function "i_APCI1710_InitTimer" and then this function.

When the timer interrupt is released, the timer generates an interrupt once the counter value is "0" (see function "i_APCI1710_SetBoardIntRoutineX").

Function call:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_EnableTimer
                                (b_BoardHandle,
                                0,
                                0,
                                APCI1710_DISABLE);
```

Return value:

0: No error

-1: The handle parameter of the board is wrong.

-2: The module number is wrong.

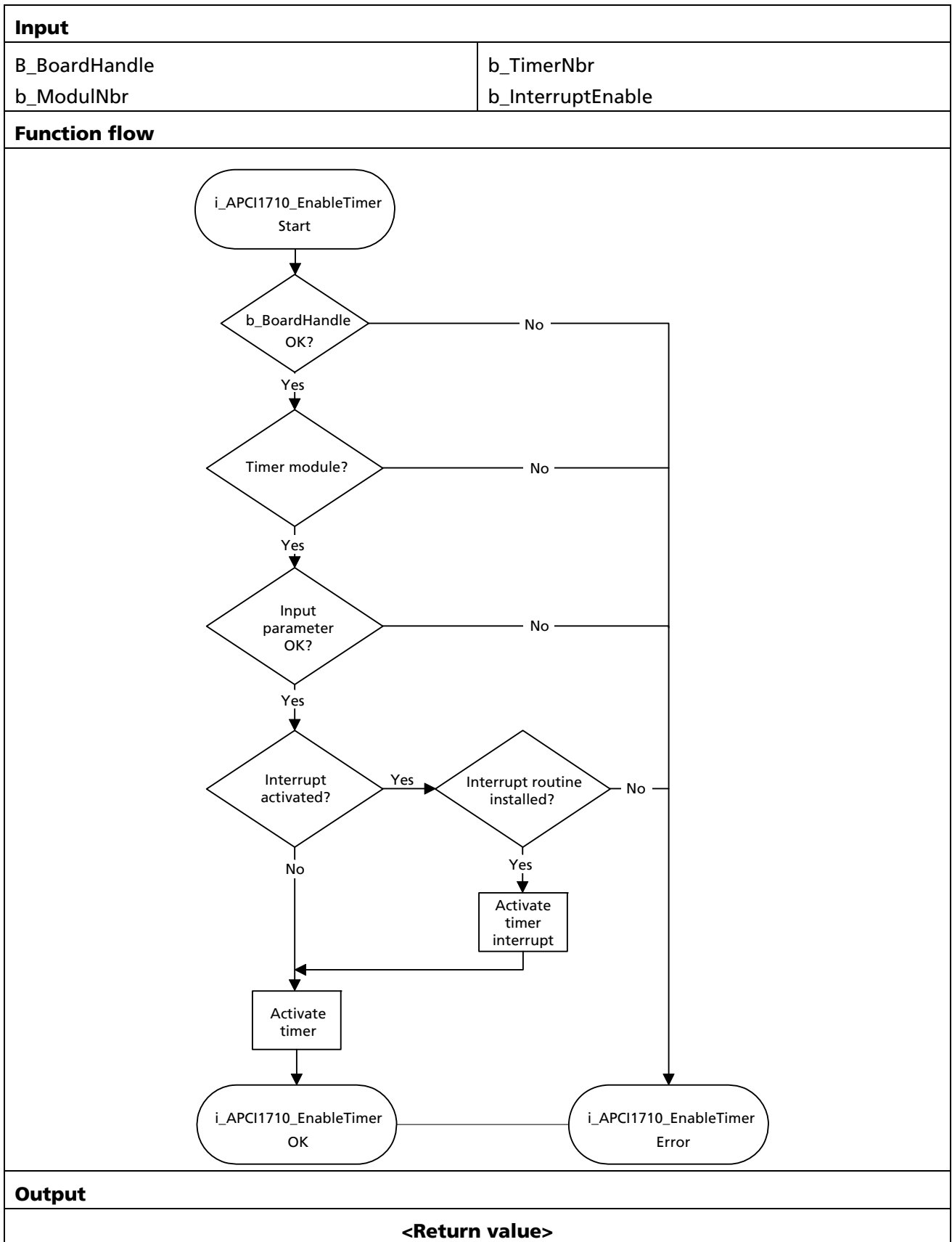
-3: The timer is wrong.

-4: The module is not a timer module.

-5: The timer is not initialised (see function "i_APCI1710_InitTimer").

-6: The interrupt parameter is wrong.

-7: The interrupt function is not initialised (see function "i_APCI1710_SetBoardIntRoutineX").



3) i_APCI1710_DisableTimer (...)**Syntax:**

```
<Return value> = i_APCI1710_DisableTimer
                                (BYTE  b_BoardHandle,
                                BYTE  b_ModulNbr,
                                BYTE  b_TimerNbr)
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of the board APCIe-1711 or APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TimerNbr	Number of the timer to be disabled (0 to 2)

- Output:

There is no output.

Task:

Deactivates the timer (b_TimerNbr) of the selected module (b_ModulNbr).

Function call:

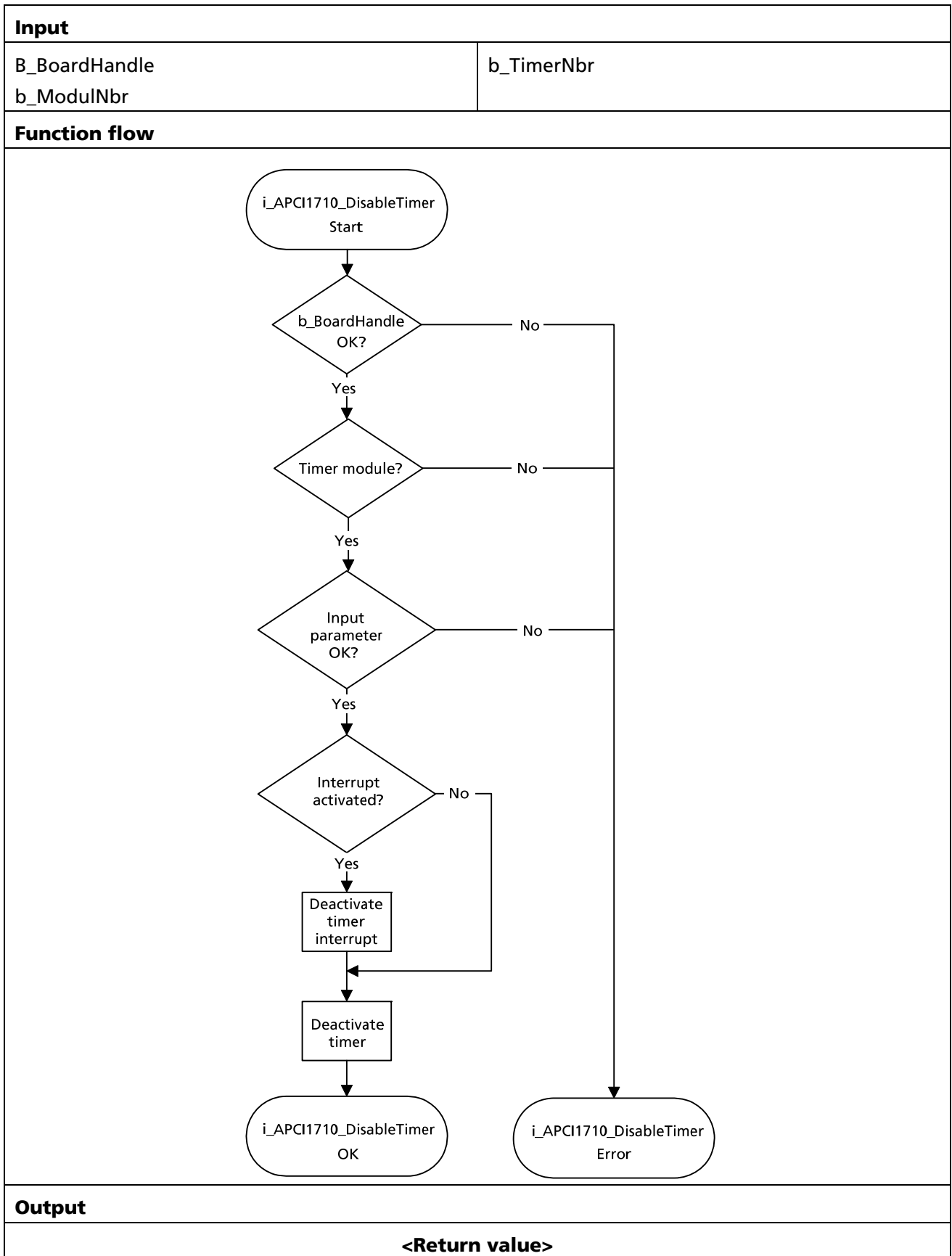
ANSI C:

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

```
i_ReturnValue = i_APCI1710_DisableTimer
                                (b_BoardHandle,
                                0,
                                0);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong.
- 2: The module number is wrong.
- 3: The timer is wrong.
- 4: The module is not a timer module.
- 5: The timer is not initialised (see function "i_APCI1710_InitTimer").



2.2.2 Reading the timer

1) i_APCI1710_ReadTimerValue (...)

Syntax:

```
<Return value> = i_APCI1710_ReadTimerValue
                                     (BYTE    b_BoardHandle,
                                     BYTE    b_ModulNbr,
                                     BYTE    b_TimerNbr,
                                     PULONG  pul_TimerValue)
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of the board APCIe-1711 or APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TimerNbr	Number of the timer to be read (0 to 2)

- Output:

PULONG	pul_TimerValue	Timer value
--------	----------------	-------------

Task:

Returns the value of the timer (b_TimerNbr) of the selected module (b_ModulNbr).

Function call:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_ReadTimerValue
                                     (b_BoardHandle,
                                     0,
                                     0,
                                     & ul_TimerValue);
```

Return value:

0: No error

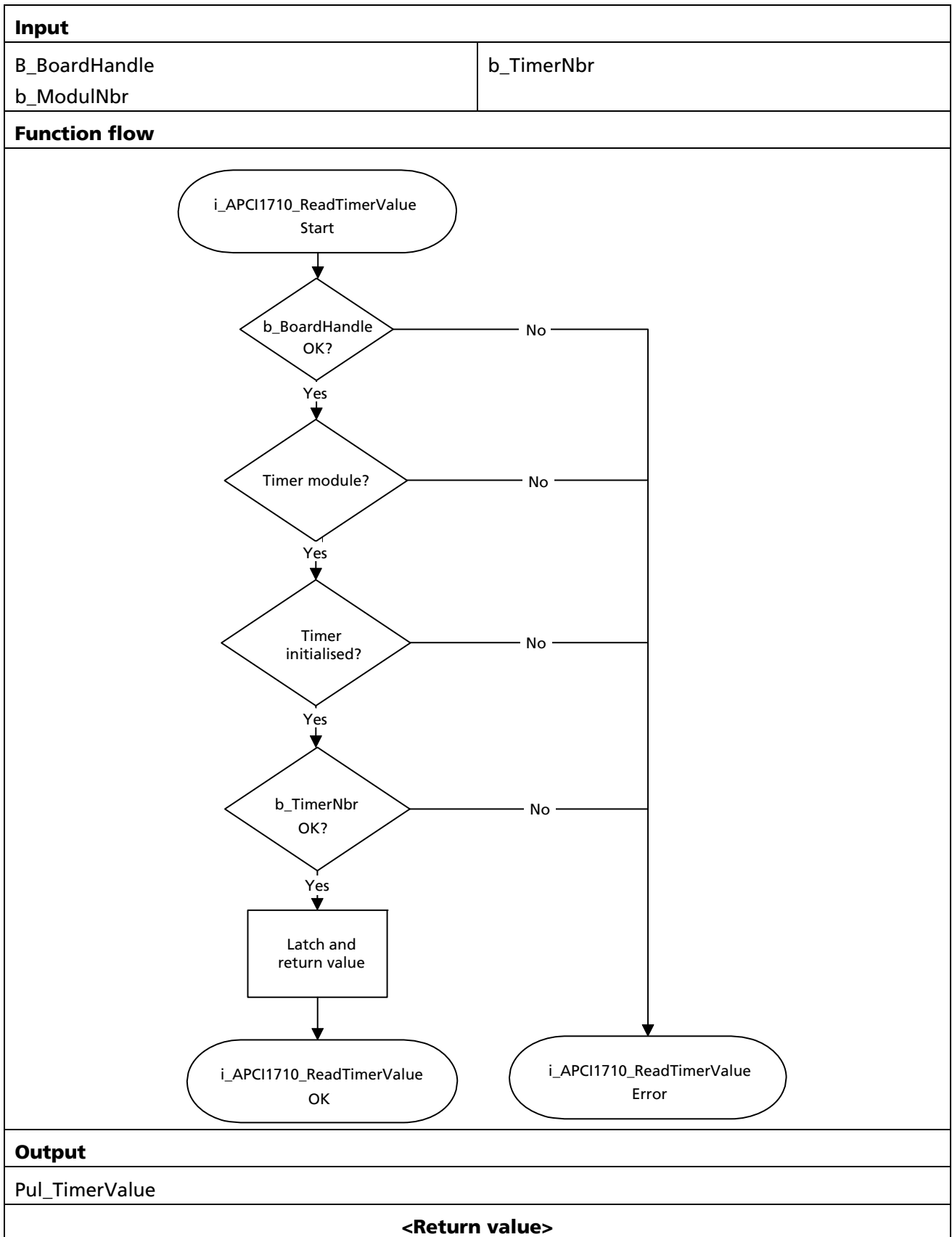
-1: The handle parameter of the board is wrong.

-2: The module number is wrong.

-3: The timer is wrong.

-4: The module is not a timer module.

-5: The timer is not initialised (see function "i_APCI1710_InitTimer").



2) i_APCI1710_ReadAllTimerValue (...)**Syntax:**

```
<Return value> = i_APCI1710_ReadAllTimerValue
                                     (BYTE      b_BoardHandle,
                                     BYTE      b_ModulNbr,
                                     PULONG    pul_TimerValueArray)
```

Parameters:**- Input:**

```
BYTE      b_BoardHandle   Handle of the board APCIe-1711 or APCI-/CPCI-1710
BYTE      b_ModulNbr     Number of the module to be configured
```

- Output:

```
PULONG    pul_TimerValueArray   Timer value sequence
                                     Element 0 contains the value of timer 0.
                                     Element 1 contains the value of timer 1.
                                     Element 2 contains the value of timer 2.
```

Task:

Returns all timer values of the selected module (b_ModulNbr).

Function call:

ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned long ul_TimerValueArray [3];

i_ReturnValue = i_APCI1710_ReadAllTimerValue
               (b_BoardHandle,
                0,
                ul_TimerValueArray);
```

Return value:

0: No error

-1: The handle parameter of the board is wrong.

-2: The module number is wrong.

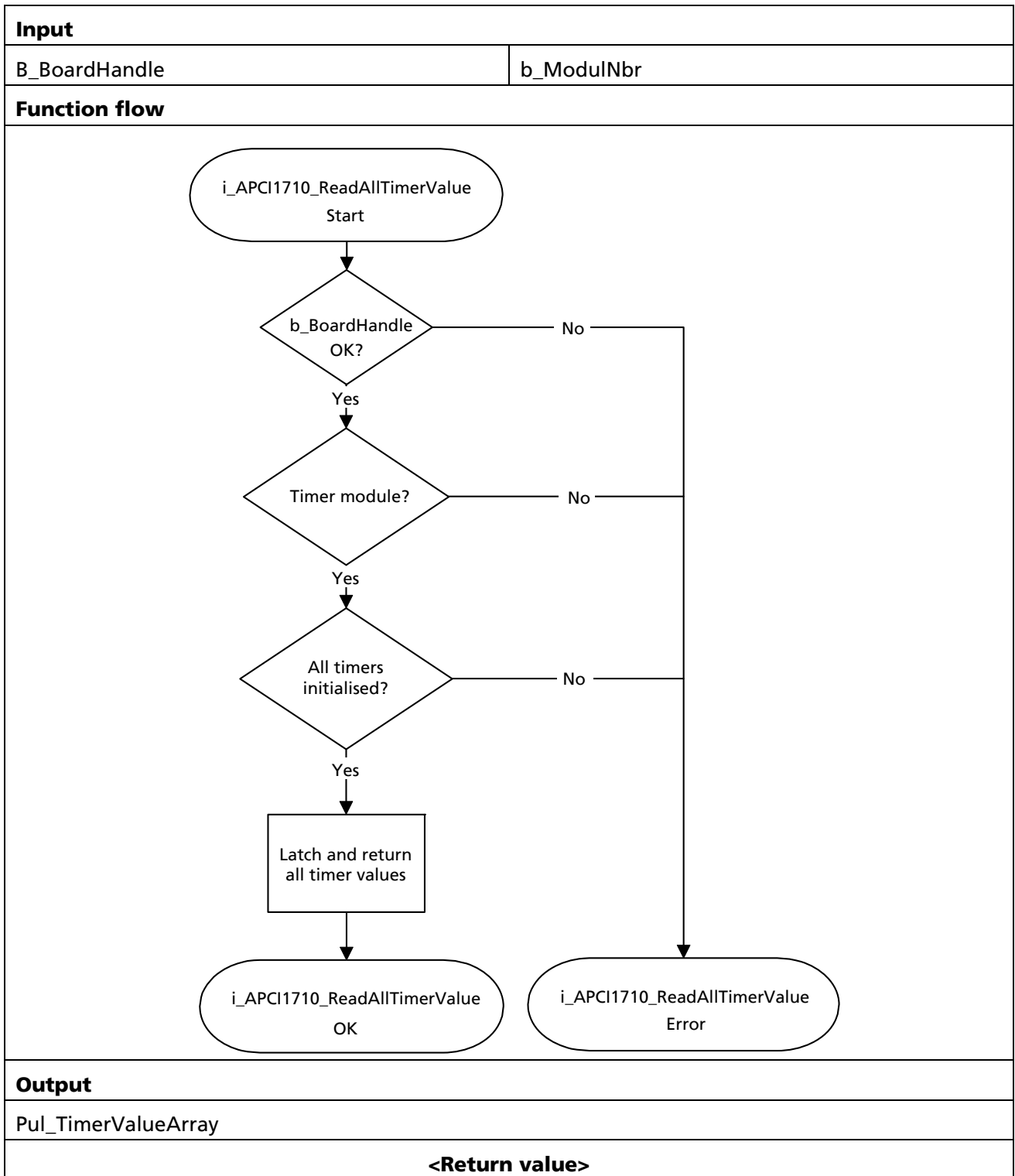
-3: The timer is wrong.

-4: The module is not a timer module.

-5: Timer 0 is not initialised (see function "i_APCI1710_InitTimer").

-6: Timer 1 is not initialised (see function "i_APCI1710_InitTimer").

-7: Timer 2 is not initialised (see function "i_APCI1710_InitTimer").



3) i_APCI1710_GetTimerOutputLevel (...)**Syntax:**

```
<Return value> = i_APCI1710_GetTimerOutputLevel
                                     (BYTE      b_BoardHandle,
                                     BYTE      b_ModulNbr,
                                     BYTE      b_TimerNbr,
                                     PBYTE     pb_OutputLevel)
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of the board APCIe-1711 or APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TimerNbr	Number of the timer to be checked (0 to 2)

- Output:

PBYTE	pb_OutputLevel	Level of the output signal 0: The output is set to "Low". 1: The output is set to "High".
-------	----------------	---

Task:

Returns the level of the output signal (pb_OutputLevel) of the timer (b_TimerNbr) of the selected module (b_ModulNbr).

Function call:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_GetTimerOutputLevel
               (b_BoardHandle,
                0,
                0,
                &b_OutputLevel);
```

Return value:

0: No error

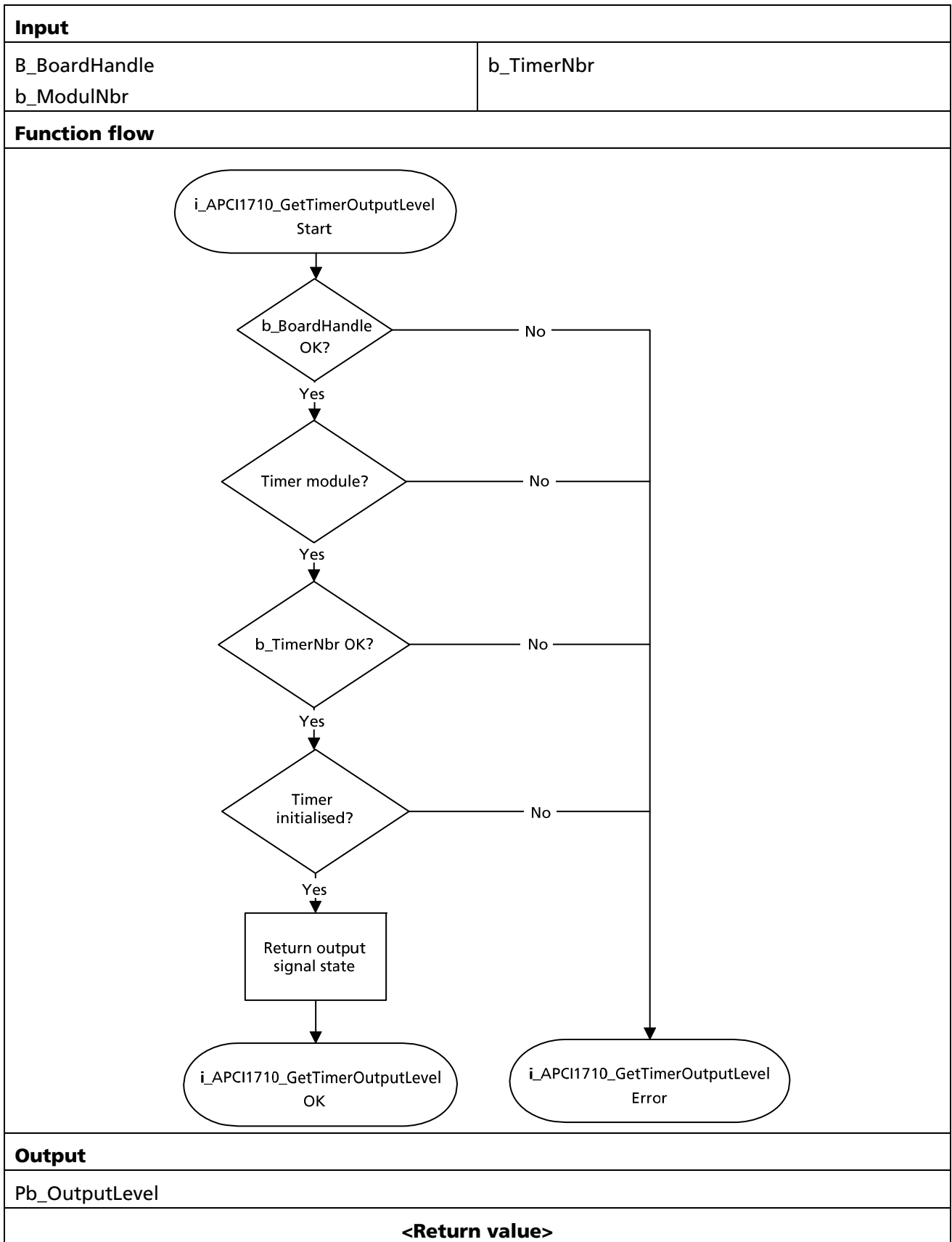
-1: The handle parameter of the board is wrong.

-2: The module number is wrong.

-3: The timer is wrong.

-4: The module is not a timer module.

-5: The timer is not initialised (see function "i_APCI1710_InitTimer").



4) i_APCI1710_GetTimerProgressStatus (...)**Syntax:**

```
<Return value> = i_APCI1710_GetTimerProgressStatus
                                     (BYTE    b_BoardHandle,
                                     BYTE    b_ModulNbr,
                                     BYTE    b_TimerNbr,
                                     PBYTE   pb_TimerStatus)
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of the board APCIe-1711 or APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TimerNbr	Number of the timer to be checked (0 to 2)

- Output:

PBYTE	pb_TimerStatus	Timer state
		0: Timer is stopped.
		1: Timer runs down.

Task:

Returns the state (pb_TimerStatus) of the timer (b_TimerNbr) of the selected module (b_ModulNbr).

Function call:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_GetTimerProgress
                                     (b_BoardHandle,
                                     0,
                                     0
                                     &b_TimerStatus);
```

Return value:

0: No error

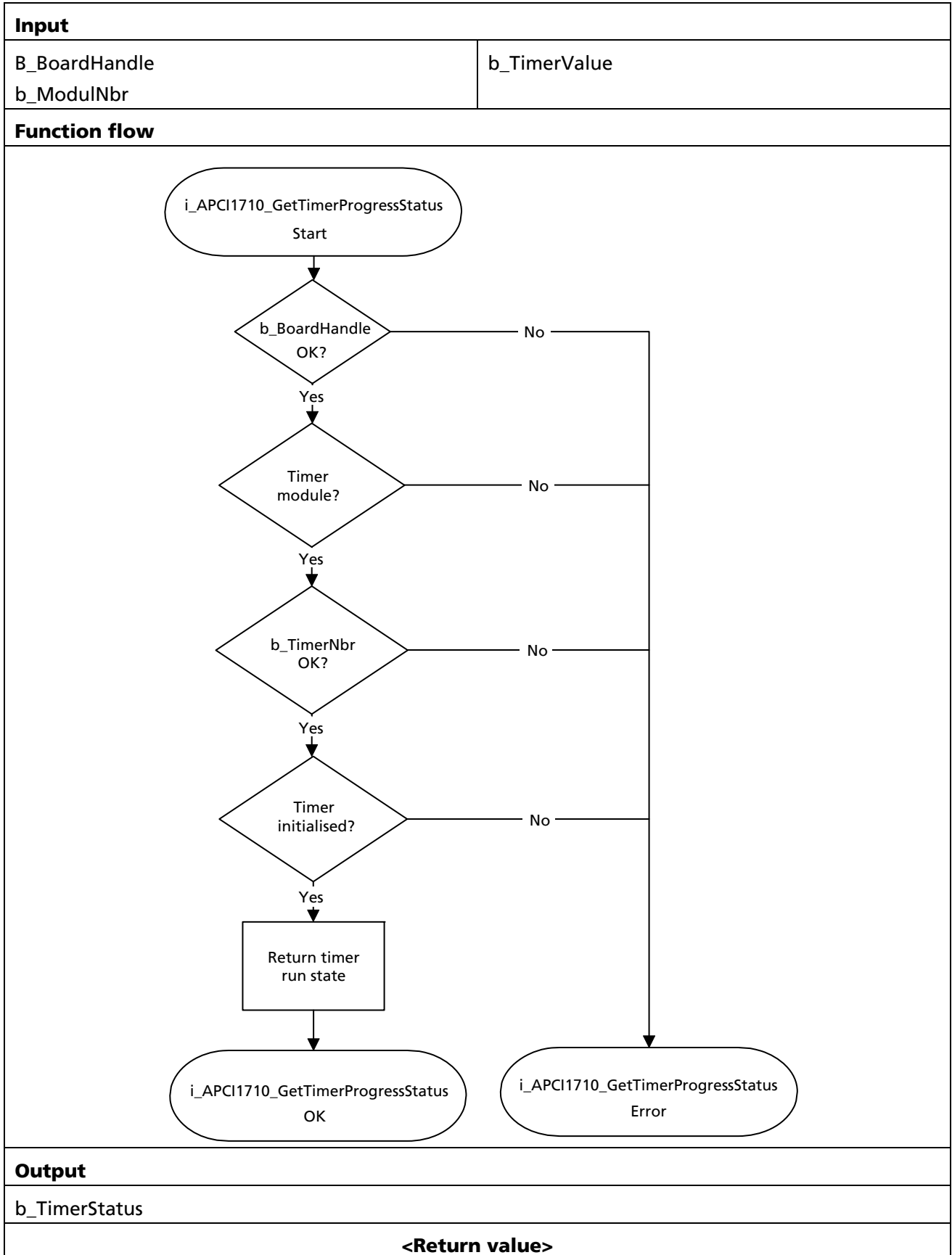
-1: The handle parameter of the board is wrong.

-2: The module number is wrong.

-3: The timer is wrong.

-4: The module is not a timer module.

-5: The timer is not initialised (see function "i_APCI1710_InitTimer").



2.2.3 Writing to the timer

1) i_APCI1710_WriteTimerValue (...)

Syntax:

```
<Return value> = i_APCI1710_WriteTimerValue
                                     (BYTE    b_BoardHandle
                                     BYTE    b_ModulNbr,
                                     BYTE    b_TimerNbr,
                                     ULONG   ul_WriteValue)
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of the board APCIe-1711 or APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TimerNbr	Number of the timer to be checked (0 to 2)
ULONG	ul_WriteValue	Value to be written

- Output:

There is no output.

Task:

Writes the value (ul_WriteValue) to the selected timer (b_TimerNbr) of the selected module (b_ModulNbr). This function depends on the operating mode used.

Function call:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_WriteTimerValue
                                     (b_BoardHandle,
                                     0,
                                     0,
                                     0xFF00FF00);
```

Return value:

0: No error

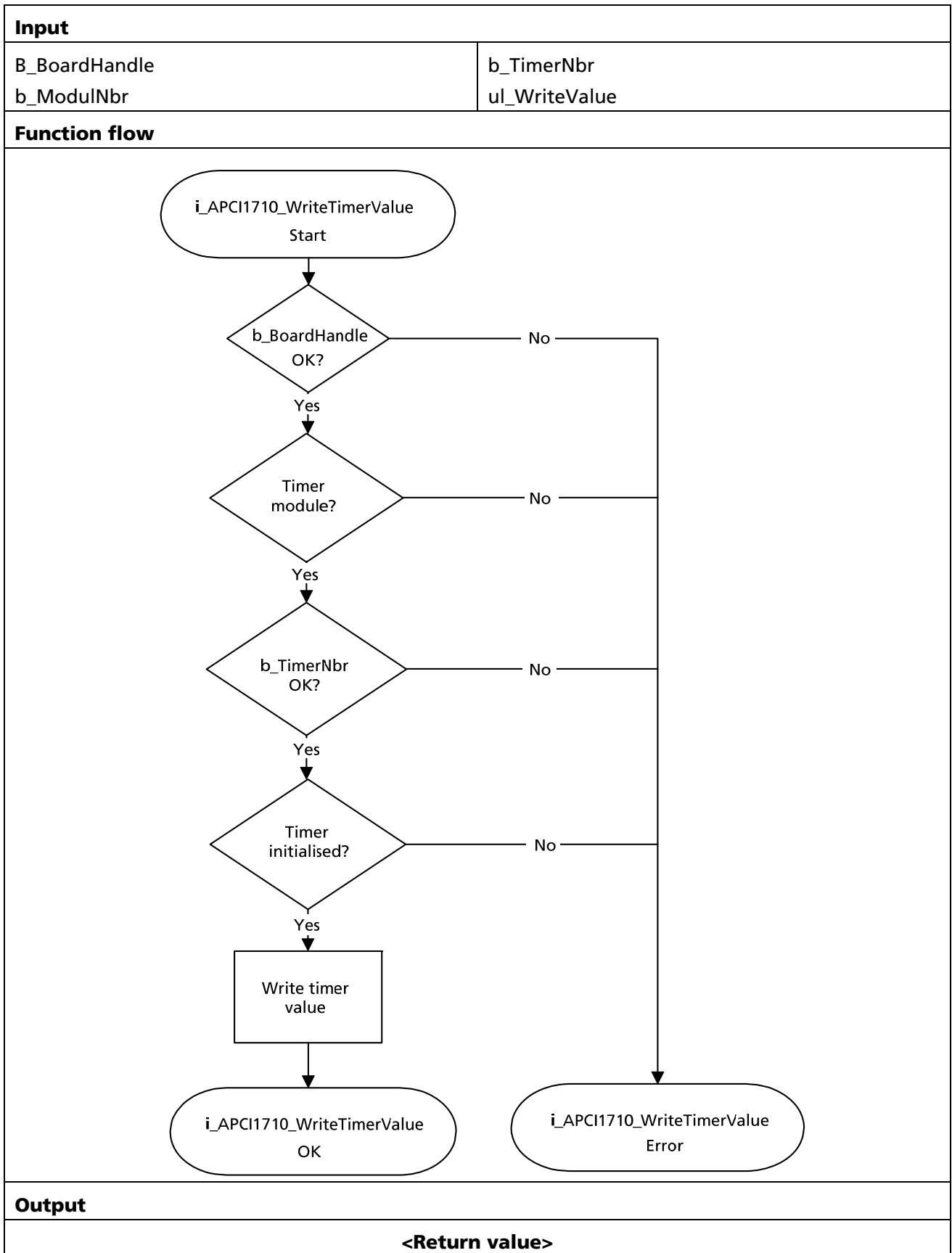
-1: The handle parameter of the board is wrong.

-2: The module number is wrong.

-3: The timer is wrong.

-4: The module is not a timer module.

-5: The timer is not initialised (see function "i_APCI1710_InitTimer").



2.3 Functions in kernel mode

2.3.1 Reading the timer

1) i_APCI1710_KRNL_ReadTimerValue (...)

Syntax:

```
<Return value> = i_APCI1710_KRNL_ReadTimerValue
                                     (UINT      ui_BaseAddress,
                                     BYTE       b_ModulNbr,
                                     BYTE       b_TimerNbr,
                                     PULONG    pul_TimerValue)
```

Parameters:

- Input:

UINT	ui_BaseAddress	Base address of the board APCIe-1711 or APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TimerNbr	Number of the timer to be read (0 to 2)

- Output:

PULONG	pul_TimerValue	Timer value
--------	----------------	-------------

Task:

Returns the value of the timer (b_TimerNbr) of the selected module (b_ModulNbr).

Function call:

ANSI C:

```
int          i_ReturnValue;
unsigned int ui_BaseAddress;
unsigned long ul_TimerValue;
```

```
i_ReturnValue = i_APCI1710_KRNL_ReadTimerValue
                                     (ui_BaseAddress,
                                     0,
                                     0,
                                     &ul_TimerValue);
```

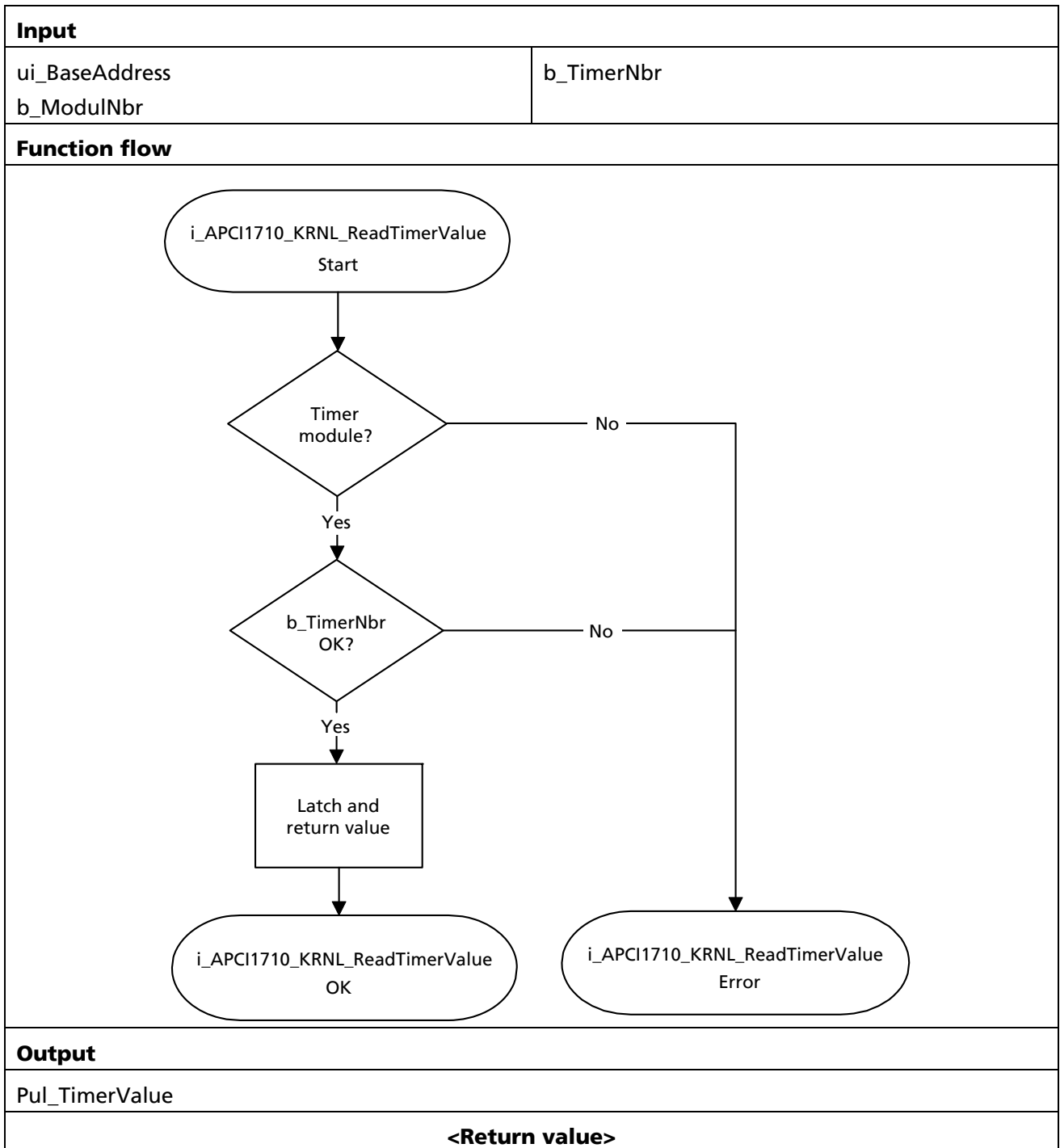
Return value:

0: No error

-1: The module number is wrong.

-2: The timer is wrong.

-3: The module is not a timer module.



2) i_APCI1710_KRNL_ReadAllTimerValue (...)**Syntax:**

```
<Return value> = i_APCI1710_KRNL_ReadAllTimerValue
                                     (UINT      ui_BaseAddress,
                                     BYTE      b_ModulNbr,
                                     PULONG   pul_TimerValueArray)
```

Parameters:**- Input:**

UINT	ui_BaseAddress	Base address of the board APCIe-1711 or APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)

- Output:

PULONG	pul_TimerValueArray	Timer value sequence Element 0 contains the value of timer 0. Element 1 contains the value of timer 1. Element 2 contains the value of timer 2.
--------	---------------------	--

Task:

Returns all timer values of the selected module (b_ModulNbr).

Function call:

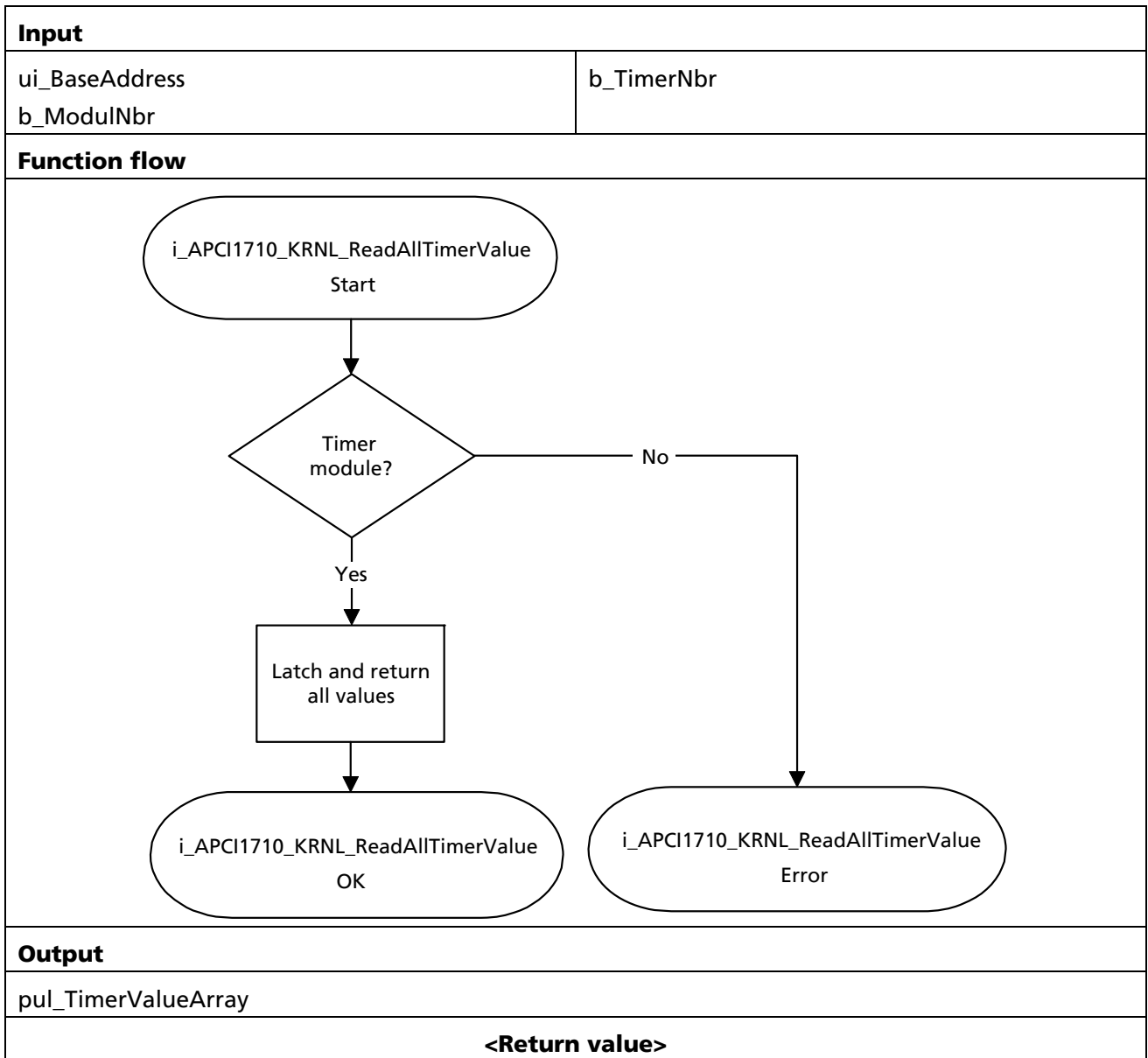
ANSI C:

```
int          i_ReturnValue;
unsigned int  ui_BaseAddress;
unsigned long ul_TimerValueArray [3];
```

```
i_ReturnValue = i_APCI1710_KRNL_ReadAllTimerValue
                                     (ui_BaseAddress,
                                     0,
                                     ul_TimerValueArray);
```

Return value:

0: No error
-1: The module number is wrong.
-2: The module is not a timer module.



2.3.2 Writing to the timer

1) i_APCI1710_KRNL_WriteTimerValue (...)

Syntax:

```
<Return value> = i_APCI1710_KRNL_WriteTimerValue
                                     (UINT    ui_BaseAddress,
                                     BYTE     b_ModulNbr,
                                     BYTE     b_TimerNbr,
                                     ULONG    ul_WriteValue)
```

Parameters:

- Input:

UINT	ui_BaseAddress	Base address of the board APCIe-1711 or APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_TimerNbr	Number of the timer to be checked (0 to 2)
ULONG	ul_WriteValue	Value to be written

- Output:

There is no output.

Task:

Writes the value (ul_WriteValue) to the selected timer (b_TimerNbr) of the selected module (b_ModulNbr). This function depends on the operating mode used.

Function call:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_WriteTimerValue
                                     (ui_BaseAddress,
                                     0,
                                     0,
                                     0xFF00FF00);
```

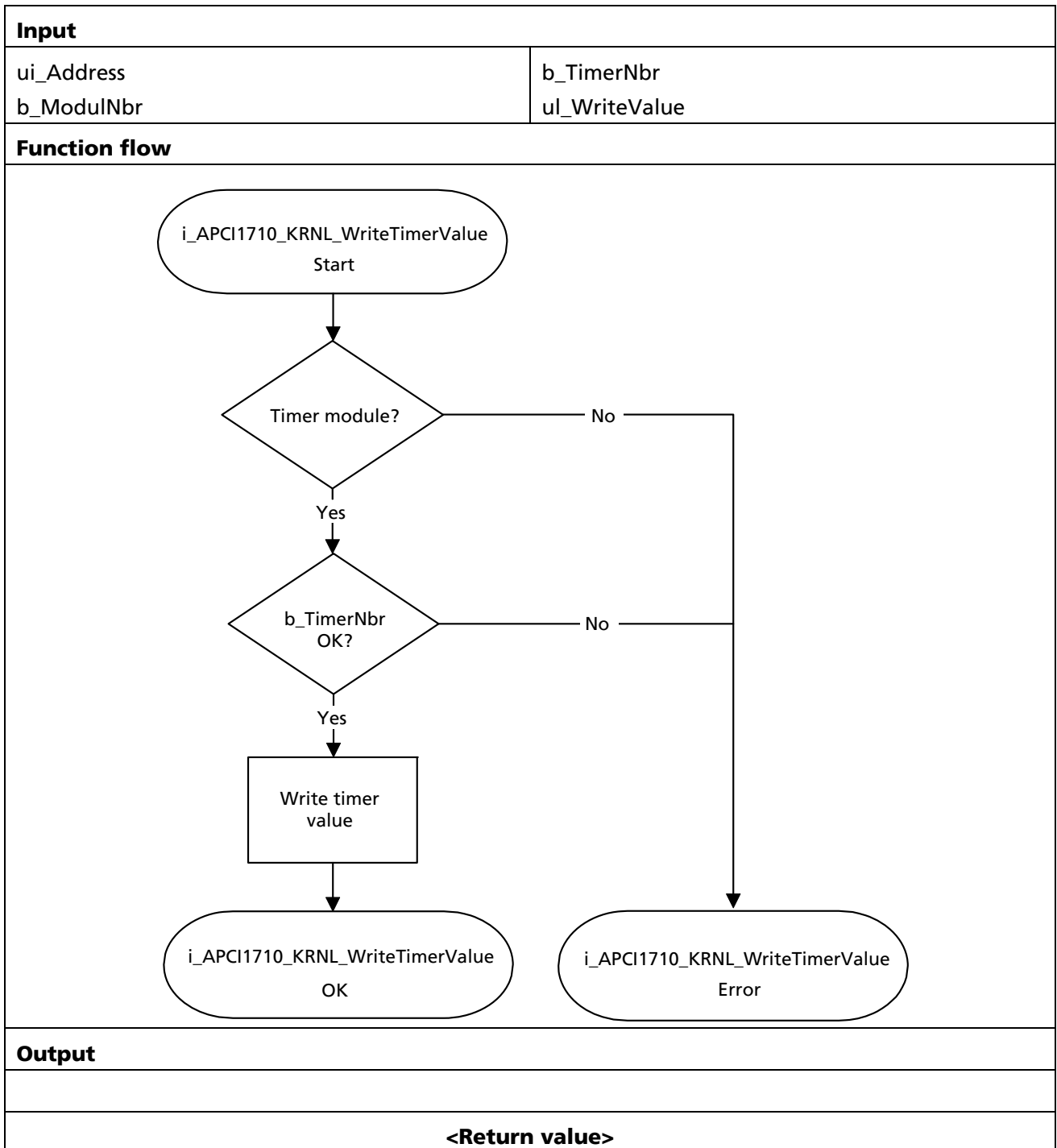
Return value:

0: No error

-1: The module number is wrong.

-2: The timer is wrong.

-3: The module is not a timer module.



3 Appendix

3.1 Index

Block diagram 7
Connection example 11
I/O address assignment 12
Modes 13
Pin assignment 9
Signals 8

Software functions
Define value 17
Initialisation 19
Interrupt mask 18
Kernel mode 43

4 Contact and support

Do you have any questions? Write or phone us:

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

Phone: +49 7229 1847-0

Fax: +49 7229 1847-222

E-mail: info@addi-data.com

Manual and software download from the Internet:

www.addi-data.com