



DIN EN ISO 9001:2000
certified



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



Technical support:
+49 7229 1847- 0

Function description

APCI-/CPCI-1710

Multifunction counter board
- Chronos -

Edition: 02.02 - 05/2008

Product information

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

Warranty and liability

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

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

Copyright

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

ADDI-DATA software product licence

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

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

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

Trademarks

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

WARNING

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



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

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

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

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

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

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

◆ **Used symbols:**



IMPORTANT!

designates hints and other useful information.



WARNING!

It designates a possibly dangerous situation.

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

- 1 DEFINITION OF APPLICATION 7**
- 1.1 Intended use7
- 1.2 Usage restrictions.....7
- 1.3 Technical description7
- 1.4 Function description8
- 1.5 Used abbreviations8
- 2 CHRONOS 9**
- 2.1 Function description9
- 2.1.1 Block diagram of the Chronos function9
- 2.1.2 Typical applications 10
- 2.2 Used signals11
- 2.3 Pin assignment of the front connector12
- 2.4 Connection example13
- 2.5 I/O mapping14
- 2.6 Description of the I/O functions.....15
- 2.6.1 Function description 15
- 2.6.2 Timer0-REGISTER (Base +0)..... 17
- 2.6.3 Timer1-Register (Base +4) 18
- 2.6.4 Chronometer-Status-Register (Base +8)..... 18
- 2.6.5 Interrupt-Status-Register (Base +12) 18
- 2.6.6 Chronometer-Configuration-Register (Base +16) 18
- 2.6.7 SET-OUT0 Register (Base +20)..... 19
- 2.6.8 SET-OUT1 Register (Base +24)..... 19
- 2.6.9 SET-OUT2 Register (Base +28)..... 19
- 2.6.10 Filter Register (Base +60)..... 19
- 2.6.11 Version Register (Base +60) 21
- 2.7 Working with the chronos function.....21
- 3 STANDARDSOFTWARE 22**
- 3.1 Introduction22
- 3.2 Interruptmask22
- 3.3 Chronos initialisation24
- 1) i_APCI1710_InitChrono (...)..... 24
- 2) i_APCI1710_EnableChrono (...) 28
- 3) i_APCI1710_DisableChrono (...) 31
- 3.4 Reading the chronometer33
- 1) i_APCI1710_GetChronoProgressStatus (...)..... 33
- 2) i_APCI1710_ReadChronoValue (...) 35
- 3) i_APCI1710_ConvertChronoValue (...)..... 38

3.5	Writing on a digital output	41
	1) i_APCI1710_SetChronoChlOn (...)	41
	2) i_APCI1710_SetChronoChlOff (...)	43
3.6	Reading a digital input	45
	1) i_APCI1710_ReadChronoChlValue (...)	45
	2) i_APCI1710_ReadChronoPortValue (...)	47
3.7	Functions in the kernel mode	49
3.7.1	Reading the chronometer.....	49
	1) i_APCI1710_KRNL_GetChronoProgressStatus (...).....	49
	2) i_APCI1710_KRNL_ReadChronoValue (...)	51
3.7.2	Writing on a digital output.....	54
	3) i_APCI1710_KRNL_SetChronoChlOn (...)	54
	4) i_APCI1710_KRNL_SetChronoChlOff (...)	56
3.7.3	Reading a digital input	58
	5) i_APCI1710_KRNL_ReadChronoChlValue (...)	58
	6) i_APCI1710_KRNL_ReadChronoPortValue (...)	60

Figures

Fig. 2-1: Block diagram of the Chronos function 10
 Fig. 2-2: Pin assignment of the front connector 12
 Fig. 2-3: Connection example 13

Tables

Table 1-1: Delivered manuals.....8
 Table 2-1: Used signals 11
 Table 2-2: I/O mapping of the Chronos function..... 14
 Table 2-3: Modes of the chronos function..... 15
 Table 3-1: Define value 22
 Table 3-2: Interruptmask of the function "chronos" 22
 Table 3-3: Counter value – return table 23
 Table 3-4: Value of the time base 25

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 is compatible with the board APCI-1710 as far as the installation of the software is concerned. The ADDIREG and SET1710 programs make no difference between PCI and CompactPCI boards.

The API functions of the standard software are also identical.

1.4 Function description

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

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

1.5 Used abbreviations

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

Please note the used abbreviations:

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

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

2 CHRONOS

2.1 Function description

The function "CHRONOS" is a timer interface which allows measuring the time between two "events" like a chronometer.

The following 3 functions are implemented

- 32-bit timer, in order to create a reference time
- 32-bit measurement timer, which determines and measures the time between start and stop pulses.
- 3 digital inputs and 3 digital outputs

The function is used for applications, in which high precision and reliability in tough industrial environment is required.

Properties:

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

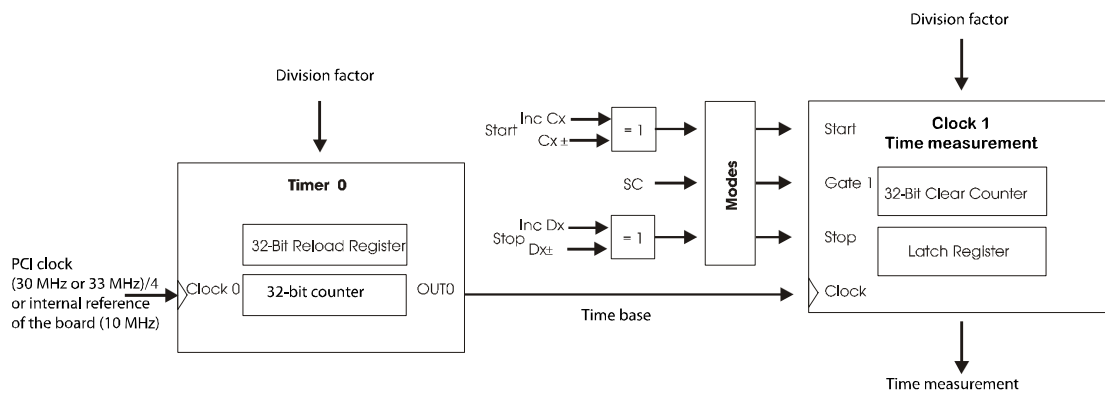
- Interrupt possibility at the end of measurement
- Signals up to 5 MHz can be processed
- Timer is rereadable
- Inputs and outputs can be inverted through software
- Software GATE possible.

2.1.1 Block diagram of the Chronos function

The interface contains:

- Digital inputs and outputs
- 2 from each other independent 32-bit timer, which can be read or written via the data bus.

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



2.1.2 Typical applications

- Time measurement between two events
- Real time clock
- Timer

2.2 Used signals

The function Chronos occupies **5 inputs** (C to G) and **3 outputs** (A, B, H) of the respecting function module of the **APCI-/CPCI-1710**.

On one board you can use max. 4 chronometers (1 per module).

Table 2-1: Used signals

ON THE CONNECTOR	POLARITY	FUNCTION
A x +/-	Diff. / TTL	Digital output 1 , after reset to logic 0
B x +/-	Diff. / TTL	Digital output 2 , after reset to logic 0
C x +/-	Diff. / TTL / Opt. 24V	Start pulse for the measurement, e.g. 0→1 edge = start Input filtered, pulse width > 100 ns
D x +/-	Diff. / TTL / Opt. 24V	Stop pulse for the measurement, e.g. 0→1 edge = stop Interruptable when the release bit is set. Input filtered Pulse width > 100 ns
E x	24V / Opt. 5V	Digital input 0 If voltage < than 15 V = Logic 1 If voltage > 17 V = Logic 0 (inverting)
F x	24V / Opt. 5V	Digital input 1 , inverting
G x	24V / Opt. 5V	Digital input 2 , inverting
H x	24 V / Opt. TTL	Digital output 0 , after reset to logic 0

x: Number of the function module.

2.3 Pin assignment of the front connector



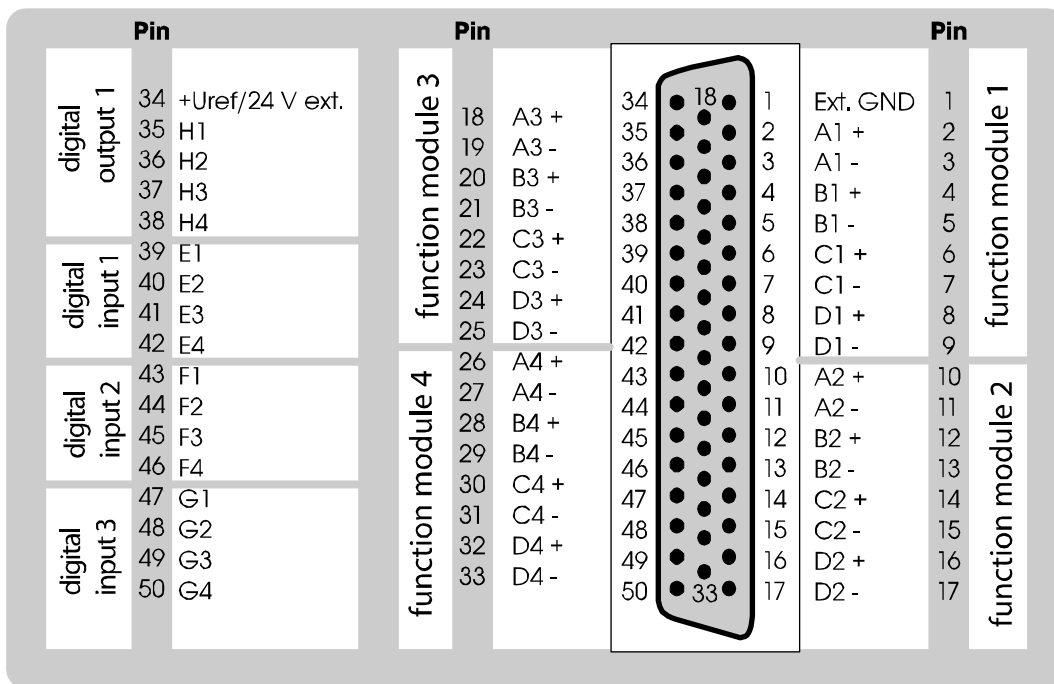
IMPORTANT!

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

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

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

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

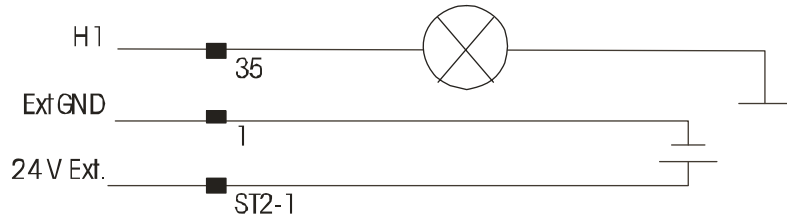


2.4 Connection example

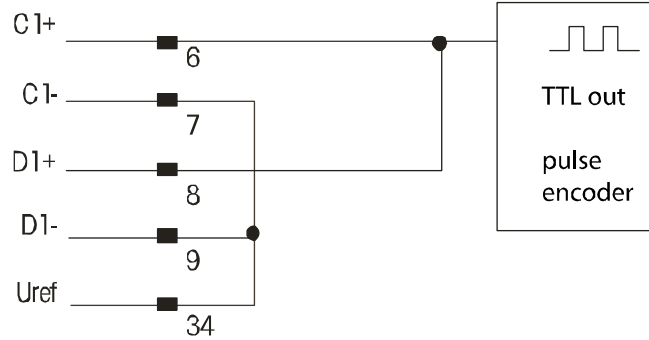
Fig. 2-3: Connection example

Function module 1

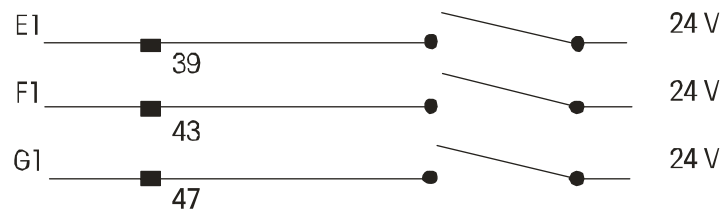
Digital output, 24 V



Time measurement in any pulse sequence



Digital inputs, 24 V



2.5 I/O mapping

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

			D31...D24	D23...D16	D15.....D8	D7.....D0
BYTES	Rd	Wr	HIGHBYTE	MIDHIGHBYTE	MIDLOWBYTE	LOWBYTE
BASE _x + 0	☑	☑	TIMER0	TIMER0	TIMER0	TIMER0
BASE _x + 4	☑		ZT-TIMER	ZT-TIMER	ZT-TIMER	ZT-TIMER
BASE _x + 8	☑		-	-	-	CHRONOMETER STATUS
BASE _x + 12	☑		-	-	-	INTERRUPT STATUS
BASE _x + 16		☑	-	-	-	CHRONOMETER CONFIG
BASE _x + 20		☑	-	-	-	SET DIGITAL OUT0
BASE _x + 24		☑	-	-	-	SET DIGITAL OUT1
BASE _x + 28		☑	-	-	-	SET DIGITAL OUT2
BASE _x + 32		☑	RESET INTERRUPT REQUEST			
BASE _x + 36		☑	CLEAR MESUREMENT			
BASE _x + 60	☑	☑	-	-	FILTER CONFIG	FILTER VALUE
			-	-	-	-
BASE _x + 60	☑		FUNKNBR2	FUNKNBR1	REVBYTE2	REVBYTE1

-: No function; x: Number of the function module.
The accesses are always read or written in 32-bit.

2.6 Description of the I/O functions

2.6.1 Function description

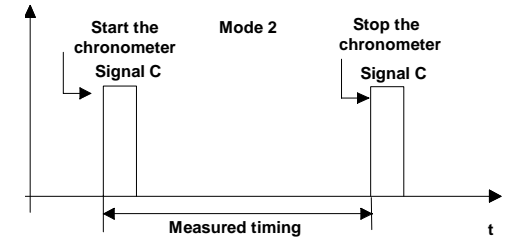
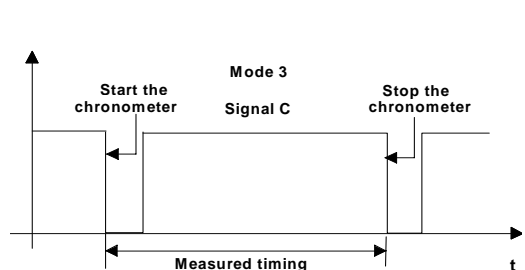
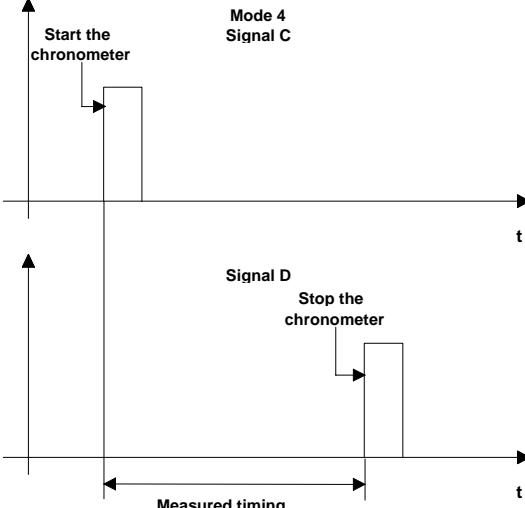
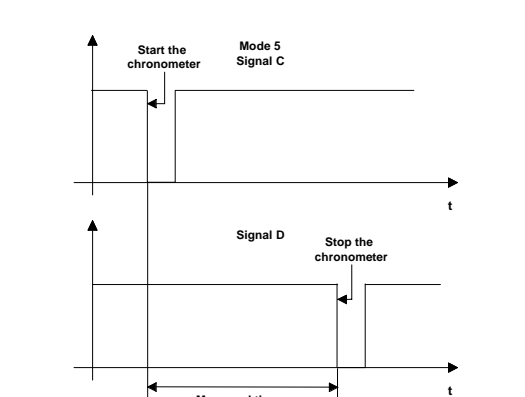
The function "Chronos" is a scaled-down version of the counter/timer. Basically, the pulse signals from Timer 0 are counted between the start pulse signal and the stop pulse signal. The number of pulses is then stored in the measuring timer and can be read through I/O access.

The timer 0 is used as a time reference generator.

The division factor is written into timer 0 and determines the time reference. The input clock is the PCI bus clock or a 40 MHz oscillator on-board. Timer 0 is synchronised with the start "event". Timer 0 can be read at any time. The function "Chronos" can be operated in different modes.

Table 2-3: Modes of the chronos function

Mode	Mode diagram	Function of signal C	Function of signal D
0	<p>Mode 0</p> <p>Start the chronometer</p> <p>Signal C</p> <p>Stop the chronometer</p> <p>Measured timing</p> <p>t</p>	<p>Der High Pegel startet die Zeitmessung.</p> <p>The high level starts the time measuring.</p> <p>The low level stops the time measuring.</p>	<p>Not used</p>
1	<p>Mode 1</p> <p>Start the chronometer</p> <p>Signal C</p> <p>Stop the chronometer</p> <p>Measured timing</p> <p>t</p>	<p>The low level starts the time measuring.</p> <p>The high level stops the time measuring</p>	<p>Not used</p>

<p>2</p>	 <p>Start the chronometer Signal C</p> <p>Mode 2</p> <p>Stop the chronometer Signal C</p> <p>Measured timing</p> <p>t</p>	<p>The high level starts the time measuring.</p> <p>The next high level stops the time measuring.</p>	<p>Not used</p>
<p>3</p>	 <p>Start the chronometer Signal C</p> <p>Mode 3</p> <p>Stop the chronometer Signal C</p> <p>Measured timing</p> <p>t</p>	<p>The low level starts the time measuring.</p> <p>The next low level stops the time measuring.</p>	<p>Not used</p>
<p>4</p>	 <p>Start the chronometer Signal C</p> <p>Mode 4</p> <p>Signal C</p> <p>Stop the chronometer Signal C</p> <p>Signal D</p> <p>Stop the chronometer Signal D</p> <p>Measured timing</p> <p>t</p>	<p>The high level starts the time measuring.</p>	<p>The high level stops the time measuring.</p>
<p>5</p>	 <p>Start the chronometer Signal C</p> <p>Mode 5</p> <p>Signal C</p> <p>Stop the chronometer Signal C</p> <p>Signal D</p> <p>Stop the chronometer Signal D</p> <p>Measured time</p> <p>t</p>	<p>The low level starts the time measuring.</p>	<p>The low level stops the time measuring.</p>

<p>6</p>		<p>The high level starts the time measuring.</p>	<p>The low level stops the time measuring.</p>
<p>7</p>		<p>The low level starts the time measuring.</p>	<p>The high level stops the time measuring.</p>

Time measurement

As soon as a **start** event (e.g.: 0→1) occurs at input **Cx**, the timer for time measurement is reset. It counts the pulse signals of timer 0. During the process the status bit "Measurement in Progress" is set in the status register. As soon as a **stop** event (e.g.: 0→1) occurs at input **Dx**, the timer is stopped and the status bit "Measurement in Progress" is reset. An interrupt can also be generated. The value is then to be read. The latest measured value can be read in the time measurement register. The timer for time measurement can be read at any time.

2.6.2 Timer0-REGISTER (Base +0)

32-bit register: The "reload" value for **TIMER 0** is written. The output frequency of timer 0 is determined by the division factor.

0 = division factor 2 → Frequency max. 20 MHz or PCI-clock/2

By reading this address, the currently latched value of **TIMER 0** is read.

2.6.3 Timer1-Register (Base +4)

32-bit register. When reading this register, the current value (i.e. last measurement) of the time measurement is read.

2.6.4 Chronometer-Status-Register (Base +8)

A 32-bit register returns the status information of the chronometer. The register can only be read.

DQ0 :	0: Measurement not started 1: Measurement started; Start event has occurred, Reset by writing on Base + 36
DQ1 :	0: Measurement not ended 1: Measurement ended; Stop event has occurred, Reset by writing on Base + 36
DQ2 :	0: Measurement not running 1: Measurement running Reset by writing on Base + 36
DQ3 :	0: Timer for time measurement has not run down 1: Timer has run down
DQ4 :	0: Single mode selected The measurement is stopped with the stop-event 1: Continuous mode selected The measurement is started automatically through the next start-event

The remaining bits will be set to 0 by reading.

2.6.5 Interrupt-Status-Register (Base + 12)

Only readable

DQ0 : Inverted image of digital input 0

DQ1 : Inverted image of digital input 1

DQ2 : Inverted image of digital input 2

DQ3 : =1: Interrupt at end of measurement
(Reset of the interrupt request by writing on Base + 32)

DQ31..4 Always set on 0.

2.6.6 Chronometer-Configuration-Register (Base + 16)

Not rereadable.

DQ0: Mode bit 0

DQ1: Mode bit 1

The mode bits determine the measurement:

B00: Time between 2 pulses on Cx line

B01: Time of the pulse on Cx line

B10: Time between pulses on Cx and Dx line

The pulses can be inverted. Additional modes (see table 2-3)

DQ2: = 1 Inversion of the signal on Cx line,
= 0 not inverted (after reset)

DQ3: = 1 Inversion of the signal on Dx line,
= 0 not inverted (after reset)

DQ4: = 1 Software release of the chronometer,
= 0 Chronometer locked (after reset)

DQ5: = 1 Interrupt generated at end of measurement
= 0 No interrupt (after reset)

DQ6: = 1 Continuous_mode set,
= 0 Single mode set (after reset)

DQ31..7: No function

2.6.7 SET-OUT0 Register (Base + 20)

DQ0 =1 causes the setting of output Hx: Current flows

DQ0 =0 causes the resetting of output Hx: Current does not flow

2.6.8 SET-OUT1 Register (Base + 24)

DQ0 =1 causes the setting of output Ax: Current flows

DQ0 =0 causes the resetting of output Ax: Current does not flow.

2.6.9 SET-OUT2 Register (Base + 28)

DQ0 =1 causes the setting of output Bx : Current flows

DQ0 =0 causes the resetting of output Bx : Current does not flow.

2.6.10 Filter Register (Base + 60)

By a simple reading access to Register 60, the Version Register described in chapter 2.6.11 is read. By a writing access to Register 60, a digital filter for the inputs C, D, E, F, G can be parameterized. The set filter values refer to positive and negative pulses. All pulses that are smaller than the set time, will be filtered. For reading back this range, the bit DQ15 must be set. Then by a reading access to the register, the filter status can be read.

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

DQ3..0 : Filter values for the inputs C, D, E, F, G base 40 MHz

0	0000	=	Filter disabled
1	0001	=	Filter of 100 ns
2	0010	=	Filter of 150 ns
3	0011	=	Filter of 200 ns
4	0100	=	Filter of 250 ns
5	0101	=	Filter of 300 ns
6	0110	=	Filter of 350 ns
7	0111	=	Filter of 400 ns
8	1000	=	Filter of 450 ns
9	1001	=	Filter of 500 ns
10	1010	=	Filter of 550 ns
11	1011	=	Filter of 600 ns
12	1100	=	Filter of 650 ns
13	1101	=	Filter of 700 ns
14	1110	=	Filter of 750 ns
15	1111	=	Filter of 800 ns

DQ3..0 : Filter values for the inputs C, D, E, F, G base 33 MHz

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

DQ3..0 : Filter values for the inputs C, D, E, F, G base 30 MHz

0	0000	=	Filter disabled
1	0001	=	Filter of 133 ns
2	0010	=	Filter of 200 ns
3	0011	=	Filter of 267 ns
4	0100	=	Filter of 333 ns
5	0101	=	Filter of 400 ns
6	0110	=	Filter of 467 ns
7	0111	=	Filter of 533 ns

8	1000	=	Filter of 600 ns
9	1001	=	Filter of 667 ns
10	1010	=	Filter of 733 ns
11	1011	=	Filter of 800 ns
12	1100	=	Filter of 867 ns
13	1101	=	Filter of 933 ns
14	1110	=	Filter of 1000 ns
15	1111	=	Filter of 1067 ns

DQ8 : 40MHz status (can only be read)

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

DQ9 : Filter clock

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

DQ15 : Enable to read the filter status

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

2.6.11 Version Register (Base + 60)

Function and revision are recognised (read command, ASCII format) **BASE + 60**
"C" "H" "1" "1"

Meaning: Chronos revision 1.1

2.7 Working with the chronos function

1. Connecting the signal encoder to the board
2. Parameterizing the API function (clock selection, time reference, start/stop-event, single or continuous mode)
3. Evaluating the status of the measurement via polling or interrupt
4. Reading out the time measurement timer
5. Time between start event and stop event is the value from the time measurement timer and time reference.

3 STANDARDSOFTWARE

3.1 Introduction



IMPORTANT!

Note the following style conventions in the text:

Function: "i_APCI1710_SetBoardInformation"

Variable *ui_Address*

Table 3-1: Define value

Define name	Decimal value	Hexadecimal value
DLL_COMPILER_C	1	1
DLL_COMPILER_VB	2	2
DLL_COMPILER_PASCAL	3	3
DLL_LABVIEW	4	4
APCI1710_30MHZ	30	1E
APCI1710_33MHZ	33	21
APCI1710_40MHZ	40	28

3.2 Interruptmask

Each pulse counter can generate an interrupt. To receive this interrupt you shall activate the interrupt and install the interrupt routine via the function "i_APCI1710_SetBoardIntRoutineX"

Table 3-2: Interruptmask of the function "chronos"

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

Table 3-3: Counter value – return table

b_ModuleMask	ul_InterruptMask	Source	ul_CounterLatchValue
b_ModuleMask = 1	ul_InterruptMask = 128	Stop signal from module 0. Measurement is stopped.	The chronometer measured a counter value.
b_ModuleMask = 2	ul_InterruptMask = 128	Stop signal from module 1. Measurement is stopped.	The chronometer measured a counter value.
b_ModuleMask = 4	ul_InterruptMask = 128	Stop signal from module 2. Measurement is stopped.	The chronometer measured a counter value.
b_ModuleMask = 8	ul_InterruptMask = 128	Stop signal from module 3. Measurement is stopped.	The chronometer measured a counter value.

3.3 Chronos initialisation

1) i_APCI1710_InitChrono (...)

Syntax:

```
<Return Wert> = i_APCI1710_InitChrono
                                (BYTE      b_BoardHandle,
                                BYTE      b_ModulNbr,
                                BYTE      b_ChronoMode,
                                BYTE      b_PCIInputClock,
                                BYTE      b_TimeUnit,
                                ULONG     ul_TimeInterval,
                                PULONG    pul_RealTimeInterval)
```

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_ChronoMode	Mode of the chronometer. (0 to 7) See table 2-2
BYTE	b_PCIInputClock	Selection of the PCI bus clock - 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_40MHZ: The board uses a 40 MHz quartz clock.
BYTE	b_TimeUnit	Unit of the time base (0 to 4) 0: ns 1: µs 2: ms 3: s 4: min.
ULONG	ul_TimeInterval	Value of the time base. See table "Value of the time base"

-Output:

PULONG	pul_RealTimeInterval	Correct value of the time base. Returns the value that corresponds mostly with the value inserted in ul_TimeInterval.
--------	----------------------	---

Table 3-4: Value of the time base

PCI Bus-Takt	<i>b_TimeUnit</i>	<i>ul_TimeInterval</i> Min. value	<i>ul_TimeInterval</i> Max. value
APCI1710_30MHz	ns (0)	66	4294967295
	µs (1)	1	143165576
	ms (2)	1	143165
	s (3)	1	143
	mn (4)	1	2
APCI1710_33MHz	ns (0)	60	4294967295
	µs (1)	1	130150240
	ms (2)	1	130150
	s (3)	1	130
	mn (4)	1	2
APCI1710_40MHz	ns (0)	50	4294967295
	µs (1)	1	107374182
	ms (2)	1	107374
	s (3)	1	107
	mn (4)	1	2

Task:

Configures the chronometer operation mode (*b_ChronoMode*) of the selected module (*b_ModulNbr*). The parameters *ul_TimeInterval* and *ul_TimeUnit* determine the time base for the measurement. *pul_RealTimeInterval* retruns the correct time value.

Call this function before you call any other function, which accesses the chronometer.

This function allows the time measurement between 2 events.

Mode 0 and 1 are considered for period measurement.

Mode 2 and 3 are considered for frequency measurement.

Mode 4 to 7 are considered for time measurement between 2 events.

Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI1710_InitChrono
                (b_BoardHandle,
                 0,
                 0,
                 APCI1710_33MHZ,
                 1,
                 100,
                 &ul_RealTimeInterval);
```

Return value:

0: No error

-1: Handle parameter of the board is wrong

-2: Selected module number is wrong.

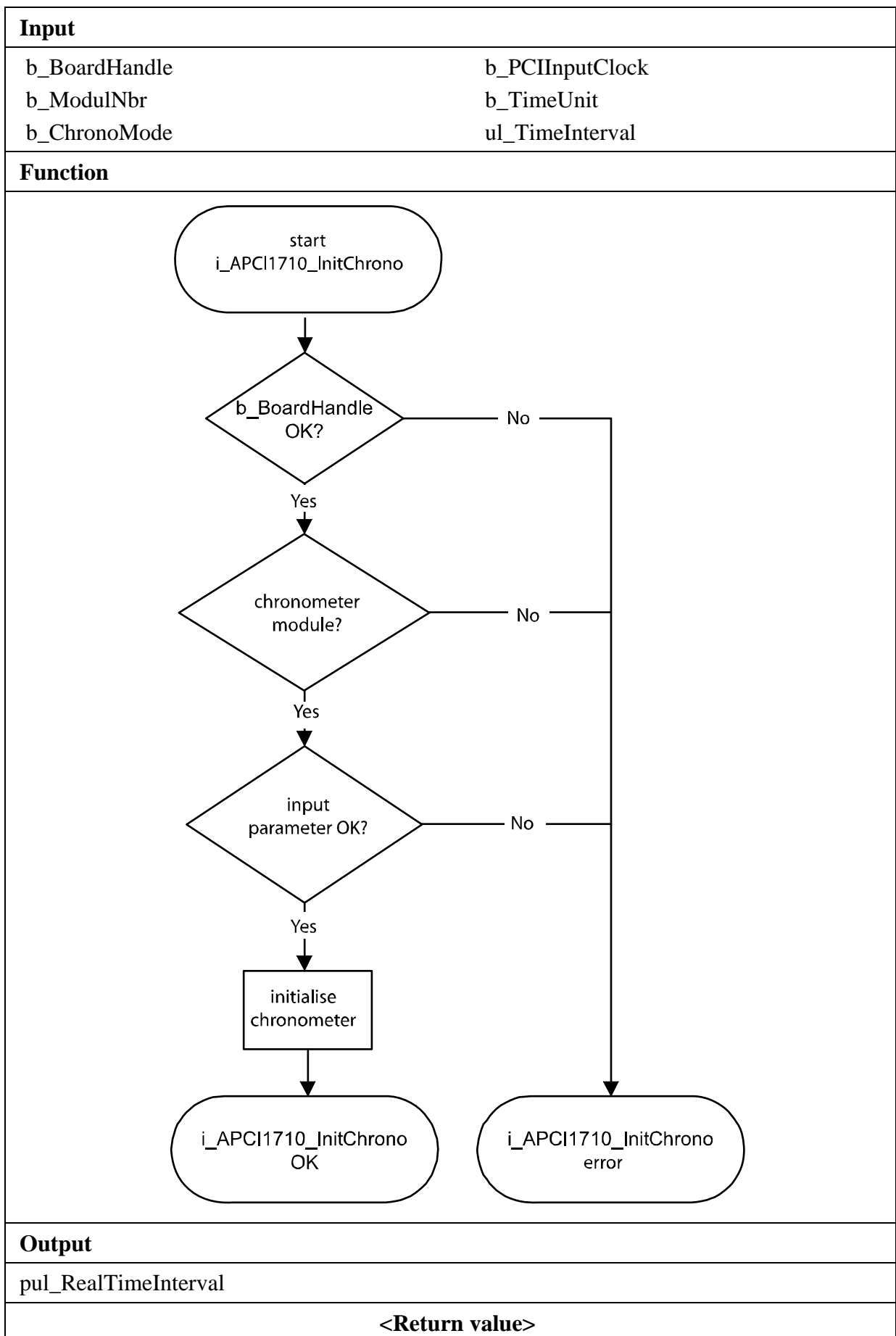
-3: Selected module is no "Chronometer" module.

-4: Selected operation mode is wrong.

-5: Selected PCI input clock is wrong.

-6: Selected time unit is wrong.

-7: Selected time base is wrong.



2) **i_APCI1710_EnableChrono (...)****Syntax:**

```
<Return Wert> = i_APCI1710_EnableChrono
                                     (BYTE      b_BoardHandle,
                                     BYTE      b_ModulNbr,
                                     BYTE      b_CycleMode,
                                     BYTE      b_InterruptEnable)
```

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_CycleMode	Selection of the "Chronos" acquisition mode - APCI1710_SINGLE: Single mode. The chronometer measures a cycle. This function is called for each cycle measurement. - APCI1710_CONTINUOUS: Continuous mode. Each start signal starts a new measurement.
BYTE	b_InterruptEnable	Enables or disables the interrupt APCI1710_ENABLE: Interrupt enabled APCI1710_DISABLE: Interrupt disabled

-Output:

There is no output.

Task:

Enables the chronometer of the selected module (*b_ModulNbr*).
The function "i_APCI1710_InitChrono" shall be called firstly. If the chronos
interrupt is enabled, the chronometer generates an interrupt after a stop signal.
See function "i_APCI1710_SetBoardIntRoutineXX" and table 3-4.
The *b_CycleMode* parameter determines if one or more cycles are read.

Calling convention:

ANSI C :

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

```
i_ReturnValue = i_APCI1710_EnableChrono
               (b_BoardHandle,
               0,
               APCI1710_SINGLE,
               APCI1710_DISABLE);
```

Return value:

0: No error

-1: Handle parameter of the board is wrong

-2: Selected module number is wrong.

-3: Selected module is no "Chronometer" module.

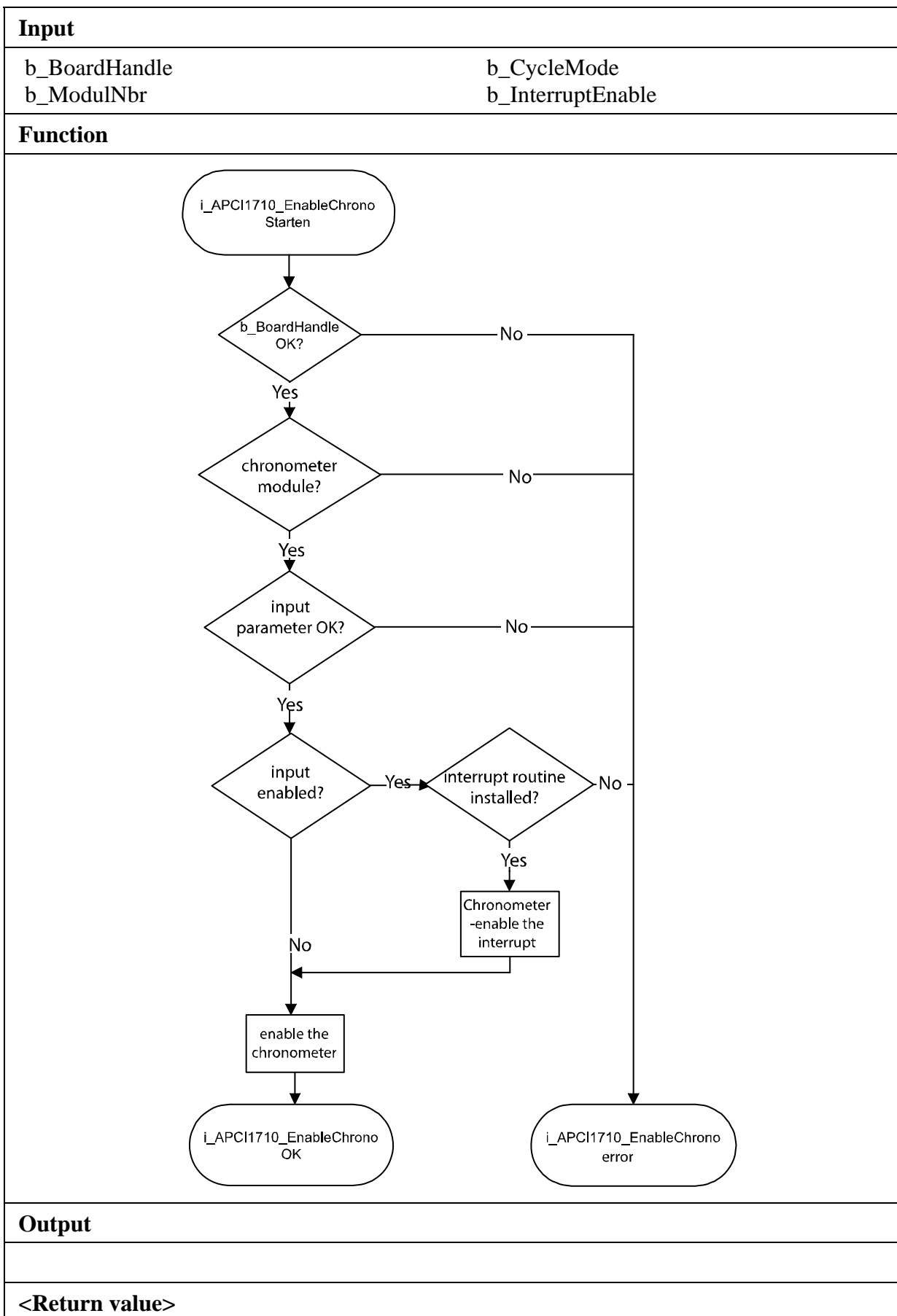
-4: Chronometer not initialised. See function "i_APCI1710_InitChrono".

-5: The cycle of the chronos acquisition mode is wrong.

-6: Interrupt parameter is wrong.

-7: Interrupt function not initialised.

See function "i_APCI1710_SetBoardIntRoutineXX".



3) i_APCI1710_DisableChrono (...)

Syntax:

```
<Return Wert> = i_APCI1710_DisableChrono
                                     (BYTE      b_BoardHandle,
                                     BYTE      b_ModulNbr)
```

Parameter:

-Input:

BYTE	b_BoardHandle	Handle of the APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)

-Output:

There is no output.

Task:

Disables the chronometer of the selected module (*b_ModulNbr*). If the chronometer is disabled after a start signal and then is restarted with the function "i_APCI1710_EnableChrono", this start signal will be ignored if no stop signal occurs.

Calling convention:

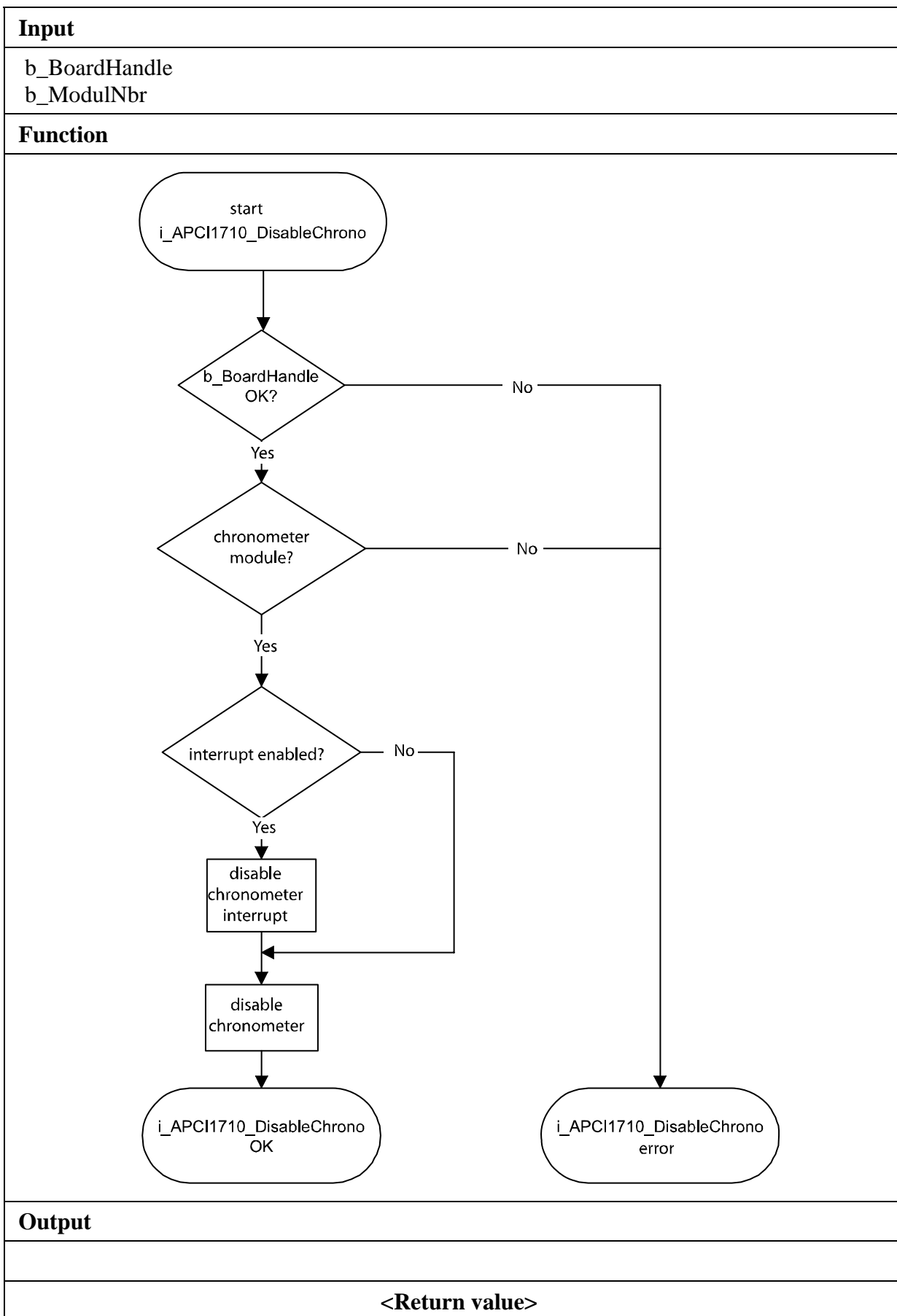
ANSI C :

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

```
i_ReturnValue = i_APCI1710_DisableChrono (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 "Chronometer" module.
- 4: Chronometer not initialised. See function "i_APCI1710_InitChrono".



3.4 Reading the chronometer

1) i_APCI1710_GetChronoProgressStatus (...)

Syntax:

```
<Return Wert> = i_APCI1710_GetChronoProgressStatus
                                     (BYTE      b_BoardHandle,
                                     BYTE      b_ModulNbr,
                                     PBYTE    pb_ChronoStatus)
```

Parameter:

-Input:

BYTE	b_BoardHandle	Handle of the APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)

-Output:

PULONG	pb_ChronoStatus	Return of the chronometer status. 0: Measurement not started. No start signal arrived. 1: Measurement started. Start signal has arrived. 2: Measurement stopped.. Stop signal has arrived. The measurement is over. 3: An overflow occurred. Please change the time base with the function "i_APCI1710_InitChrono".
--------	-----------------	---

Task:

Returns the chronometer status (*pb_ChronoStatus*) of the selected module (*b_ModulNbr*).

Calling convention:

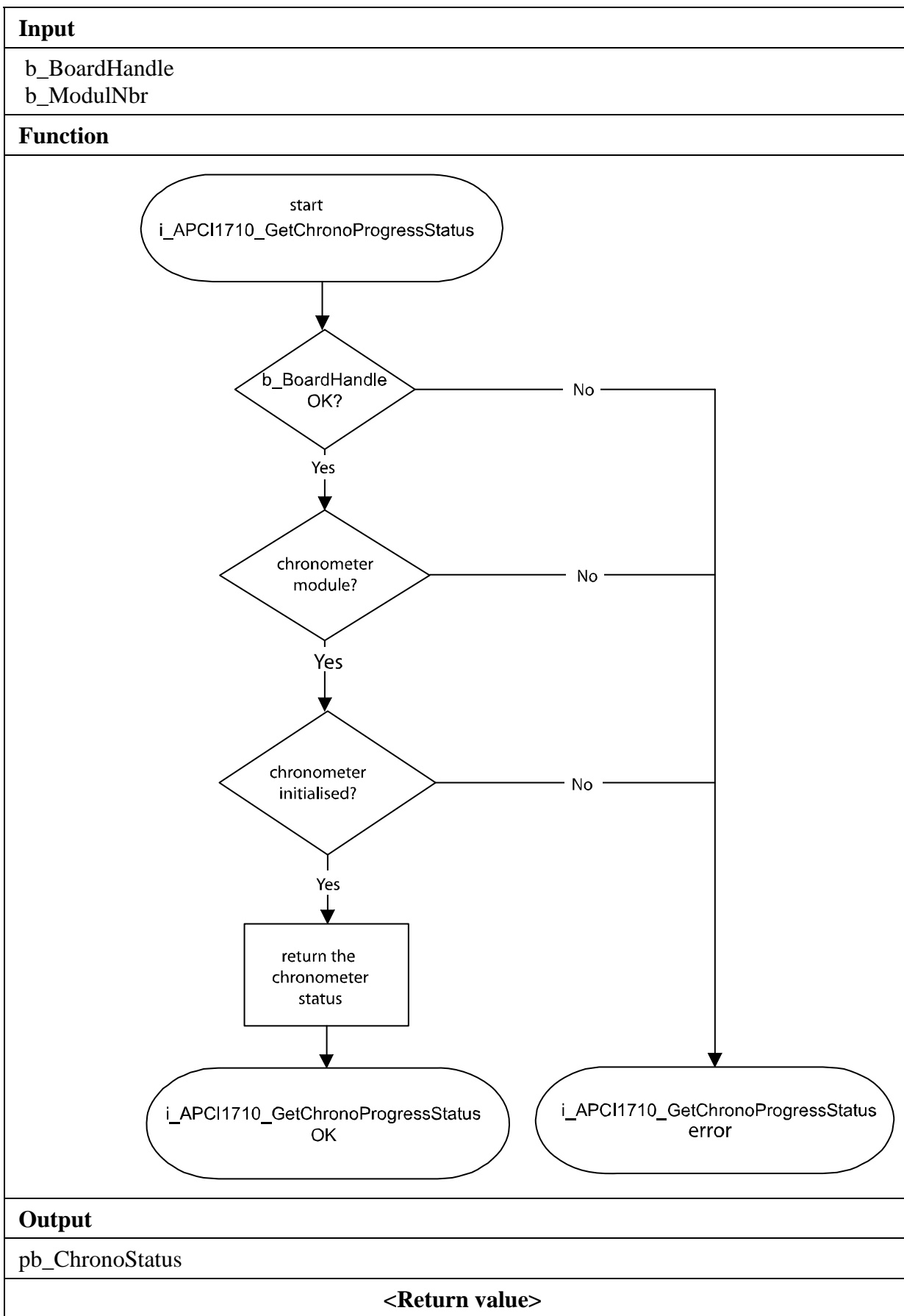
ANSI C :

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

```
i_ReturnValue = i_APCI1710_GetChronoProgressStatus
                (b_BoardHandle,
                 0,
                 &pb_ChronoStatus);
```

Return value

0: No error
-1: Handle parameter of the board is wrong
-2: Selected module number is wrong.
-3: Selected module is no "Chronometer" module.
-4: Chronometer not initialised. See function "i_APCI1710_InitChrono".



2) **i_APCI1710_ReadChronoValue (...)**

Syntax:

<Return Wert> = i_APCI1710_ReadChronoValue
 (BYTE b_BoardHandle,
 BYTE b_ModulNbr,
 UINT ui_TimeOut,
 PBYTE pb_ChronoStatus,
 PULONG pul_ChronoValue)

Parameter:

-Input:

BYTE	b_BoardHandle	Handle of the APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
UINT	ui_TimeOut	Selection of the timeout (0 to 65535) 0: Timeout not used. The function returns the chronometer status and measures the counter value, when a stop signal has arrived. 1 to 65535: Determines the timeout in ms. The function will be stopped after a timeout or a stop signal.

-Output:

PULONG	pb_ChronoStatus	Return of the chronometer status. 0: Measurement not started. No start signal arrived. 1: Measurement started. A start signal has arrived. 2: Measurement stopped. A stop signal has arrived. The measurement is over and <i>pul_ChronoValue</i> returns the chronometer time. 3: An overflow occurred. Please change the time base with the function "i_APCI1710_InitChrono" 4: Timeout occurred.
PULONG	pul_ChronoValue	Chronometer time value.

Task:

Return of the chronometer status (*pb_ChronoStatus*) and of the time value (*pul_ChronoValue*) after a stop signal on the selected module (*b_ModulNbr*). This function is only available when you have disabled the interrupt function. See function "i_APCI1710_EnableChrono" and table 3-4. The chronometer status can be tested with the function "i_APCI1710_GetChronoProgressStatus". The value returned by *pul_ChronoValue* is not the correct time value. Use the "i_APCI1710_ConvertChronoValue" function. Otherwise, the following formula is applied in order to calculate the correct time value:

$$\text{Time value} = \text{pul_ChronoValue} \times \text{pul_RealTimeInterval}.$$

pul_RealTimeInterval is the returned value from "i_APCI1710_InitChrono". The time function is the variable *b_TimeUnit* of the function "i_APCI1710_InitChrono".

Calling convention:

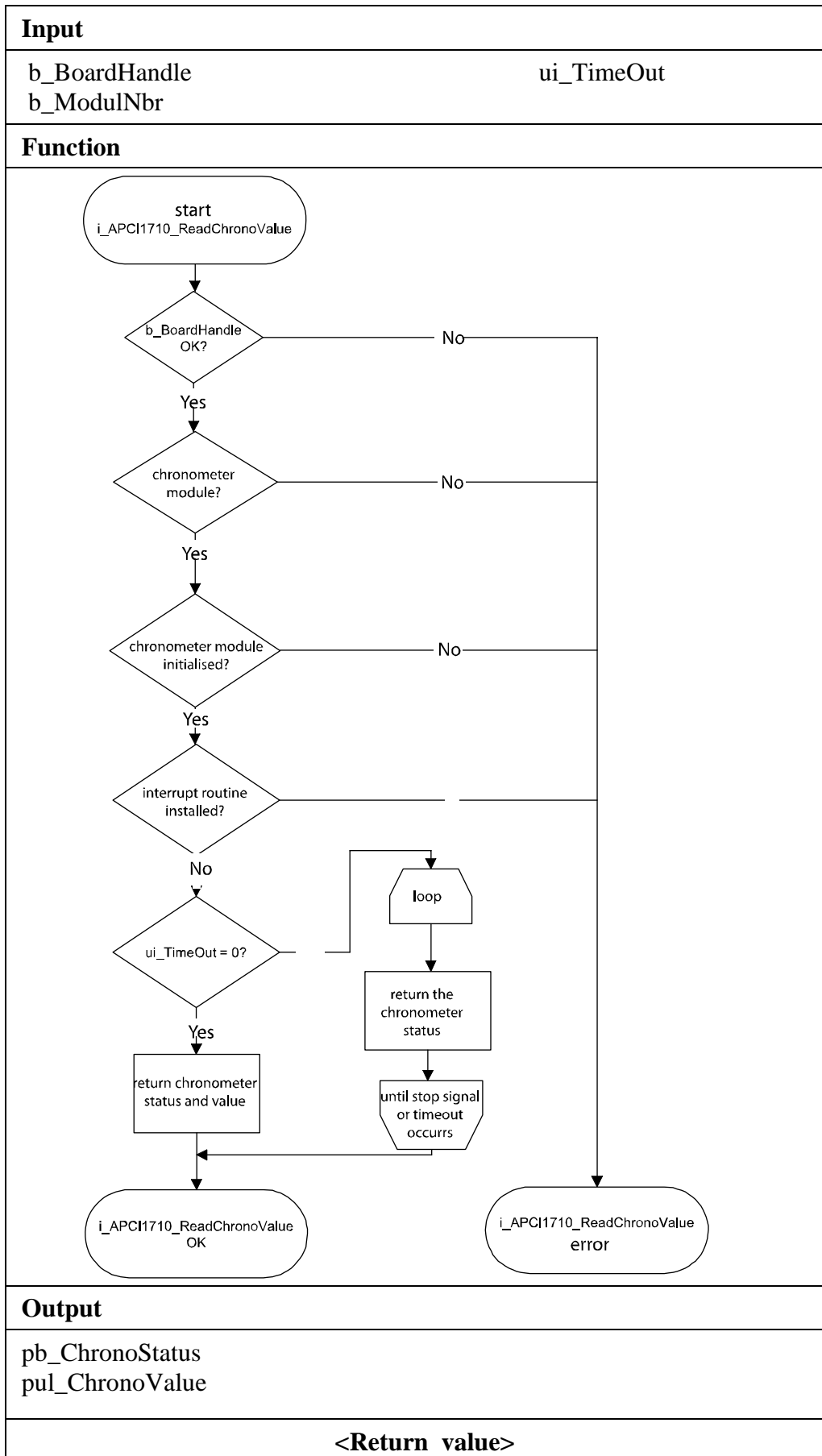
ANSI C:

```
int          i_ReturnValue;  
unsigned char b_BoardHandle;  
unsigned char b_ChronoStatus;  
unsigned long ul_ChronoValue;
```

```
i_ReturnValue = i_APCI1710_ReadChronoValue  
                (b_BoardHandle,  
                 0,  
                 0,  
                 &pb_ChronoStatus,  
                 &ul_ChronoValue);
```

Return value:

- 0: No error
- 1: Handle parameter of the board is wrong
- 2: Selected module number is wrong.
- 3: The selected module is no "Chronometer" module.
- 4: Chronometer not initialised. See function "i_APCI1710_InitChrono".
- 5: Timeout parameter is wrong (0 to 65535).
- 6: Interrupt routine installed. The measured chronometer time cannot be read directly.



3) `i_APCI1710_ConvertChronoValue (...)`**Syntax:**

```
<Return Wert> = i_APCI1710_ConvertChronoValue
                (BYTE      b_BoardHandle,
                 BYTE      b_ModulNbr,
                 ULONG     ul_ChronoValue,
                 PULONG    pul_Hour,
                 PBYTE     pb_Minute,
                 PBYTE     pb_Second,
                 PUINT     pui_MilliSecond,
                 PUINT     pui_MicroSecond,
                 PUINT     pui_NanoSecond)
```

Parameter:**-Input:**

BYTE	b_BoardHandle	Handle of the APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
ULONG	ul_ChronoValue	Chronometer time value. See "i_APCI1710_ReadChronoValue"

-Output:

PULONG	pul_Hour	Time measurement in hours
PBYTE	pb_Minute	Time measurement in minutes
PBYTE	pb_Second	Time measurement in seconds
PUINT	pui_MilliSecond	Time measurement in milliseconds
PUINT	pui_MicroSecond	Time measurement in microseconds.
PUINT	pui_NanoSecond	Time measurement in nanoseconds

Task:

Conversion of the measured chronometer time (*ul_ChronoValue*) in h, mn, s, ms, μ s and ns.

Calling convention:ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned int  ui_MilliSecond;
unsigned int  ui_MicroSecond;
unsigned int  ui_NanoSecond;
unsigned char b_Second;
unsigned char b_Minute;
i_ReturnValue = i_APCI1710_ConvertChronoValue
                (b_BoardHandle,
                 0,
                 0,
                 &b_Minute,
                 &b_Second,
                 &ui_MilliSecond,
                 &ui_MicroSecond,
                 &ui_NanoSecond);
```

Return value

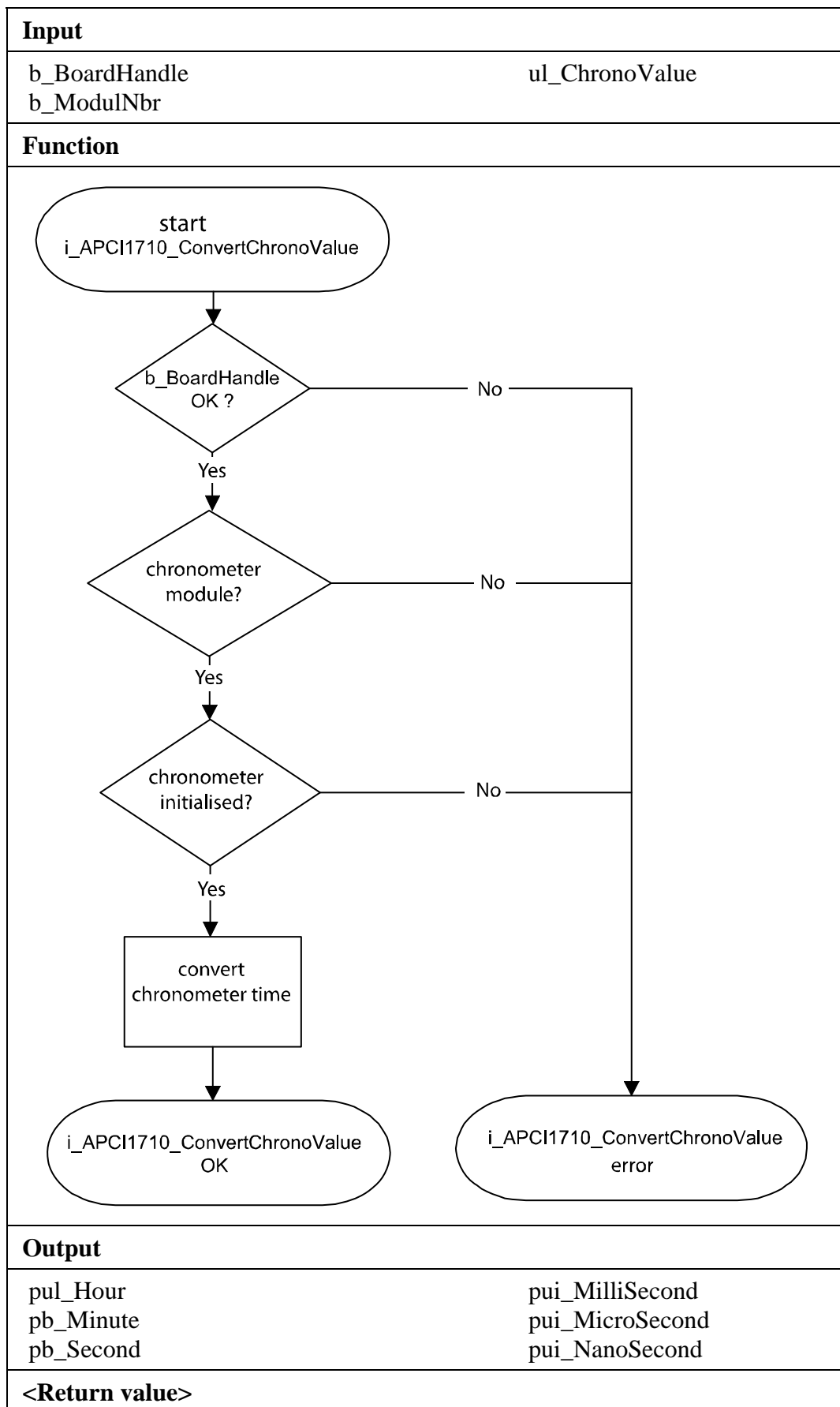
0: No error

-1: Handle parameter of the board is wrong

-2: Selected module number is wrong.

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

-4: Chronometer not initialised. See function "i_APCI1710_InitChrono".



3.5 Writing on a digital output

1) i_APCI1710_SetChronoChlOn (...)

Syntax:

```
<Return Wert> = i_APCI1710_SetChronoChlOn
                                     (BYTE      b_BoardHandle,
                                     BYTE      b_ModulNbr,
                                     BYTE      b_OutputChannel)
```

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_OutputChannel	Selection of the digital output (0 to 2) 0: Output H 1: Output A 2: Output B

-Output:

There is no output.

Task:

Sets the output entered by b_OutputChannel. Setting an output means setting an output on "High".

Calling convention:

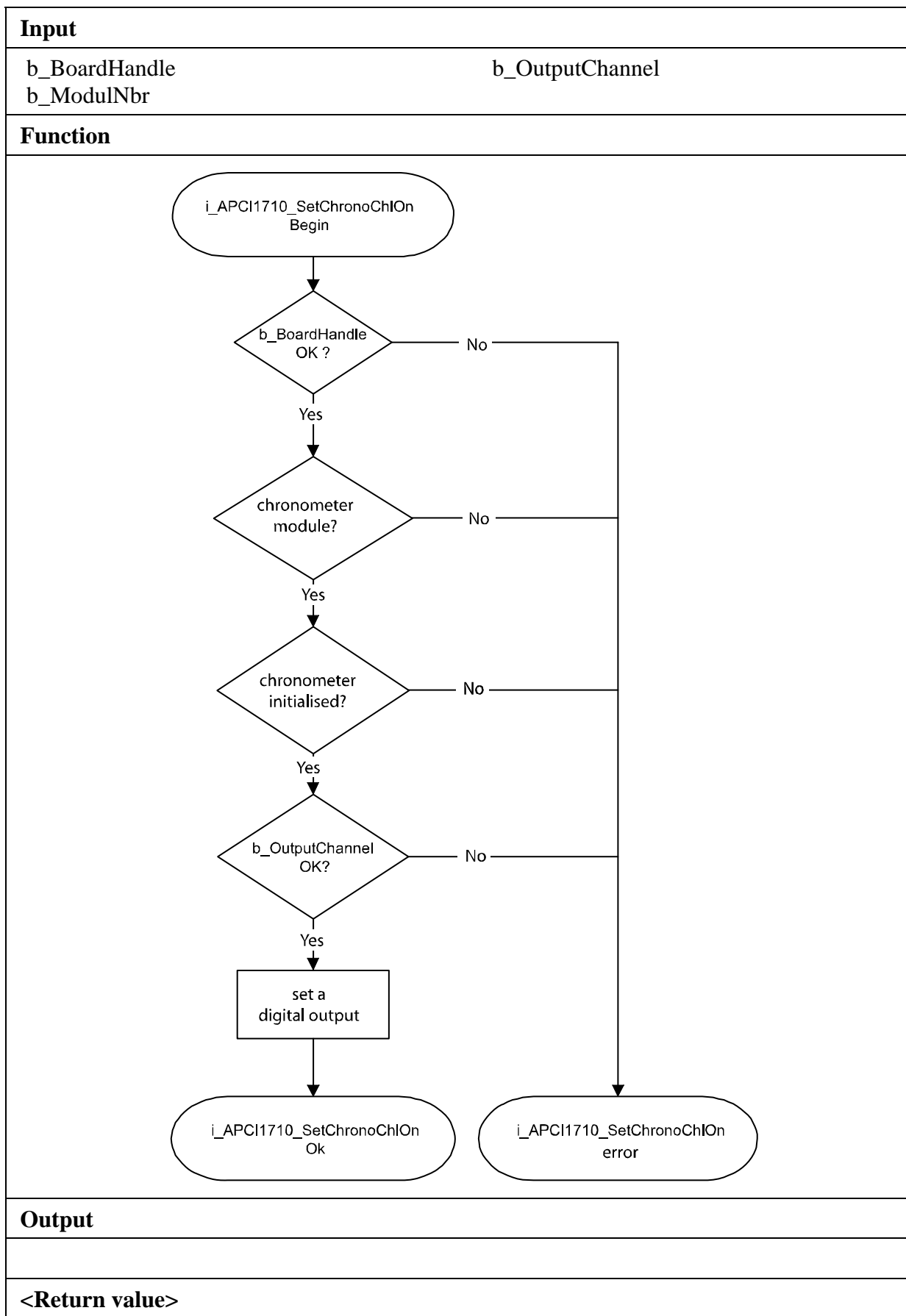
ANSI C:

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

```
i_ReturnValue = i_APCI1710_SetChronoChlOn
                (b_BoardHandle,
                 0,
                 0);
```

Return value

0: No error
-1: Handle parameter of the board is wrong
-2: Selected module number is wrong.
-3: Selected module is not "Chronometer" module.
-4: The selected digital output is wrong.
-5: Chronometer not initialised. See function "i_APCI1710_InitChrono".



2) **i_APCI1710_SetChronoChlOff (...)**

Syntax:

```
<Return Wert> = i_APCI1710_SetChronoChlOff
                                     (BYTE      b_BoardHandle,
                                     BYTE      b_ModulNbr,
                                     BYTE      b_OutputChannel)
```

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_OutputChannel	Selection of a digital output (0 to 2) 0: Output H 1: Output A 2: Output B

-Output:

There is no output.

Task:

Resets the output entered by b_OutputChannel. Resetting an output means setting an output to Low.

Calling convention:

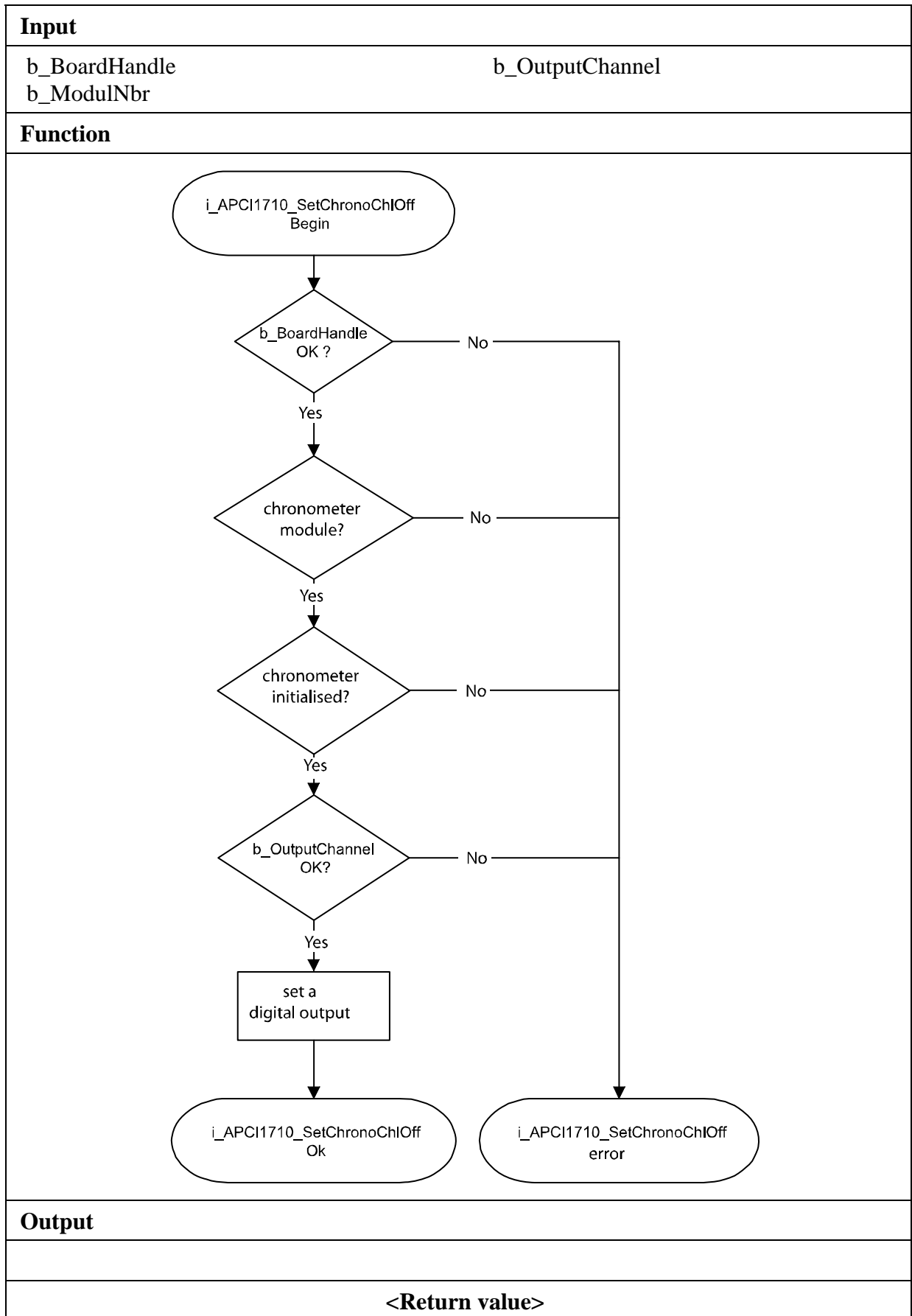
ANSI C :

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

```
i_ReturnValue = i_APCI1710_SetChronoChlOff (b_BoardHandle,
                                             0,
                                             0);
```

Return value

- 0: No error
- 1: Handle parameter of the board is wrong
- 2: Selected module number is wrong.
- 3: Selected module is no "Chronometer" module.
- 4: The selected digital output is wrong.
- 5: Chronometer not initialised. See function "i_APCI1710_InitChrono"



3.6 Reading a digital input

1) i_APCI1710_ReadChronoChlValue (...)

Syntax:

```
<Return Wert> = i_APCI1710_ReadChronoChlValue
                                     (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 a digital input (0 to 2) 0: Input E 1: Input F 2: Input G

-Output:

PBYTE	pb_ChannelStatus	Status of the digital input. 0: Channel enabled 1: Channel disabled
-------	------------------	---

Task:

Return of the status of the digital input (*b_InputChannel*) in the selected module (*b_ModulNbr*).

Calling convention:

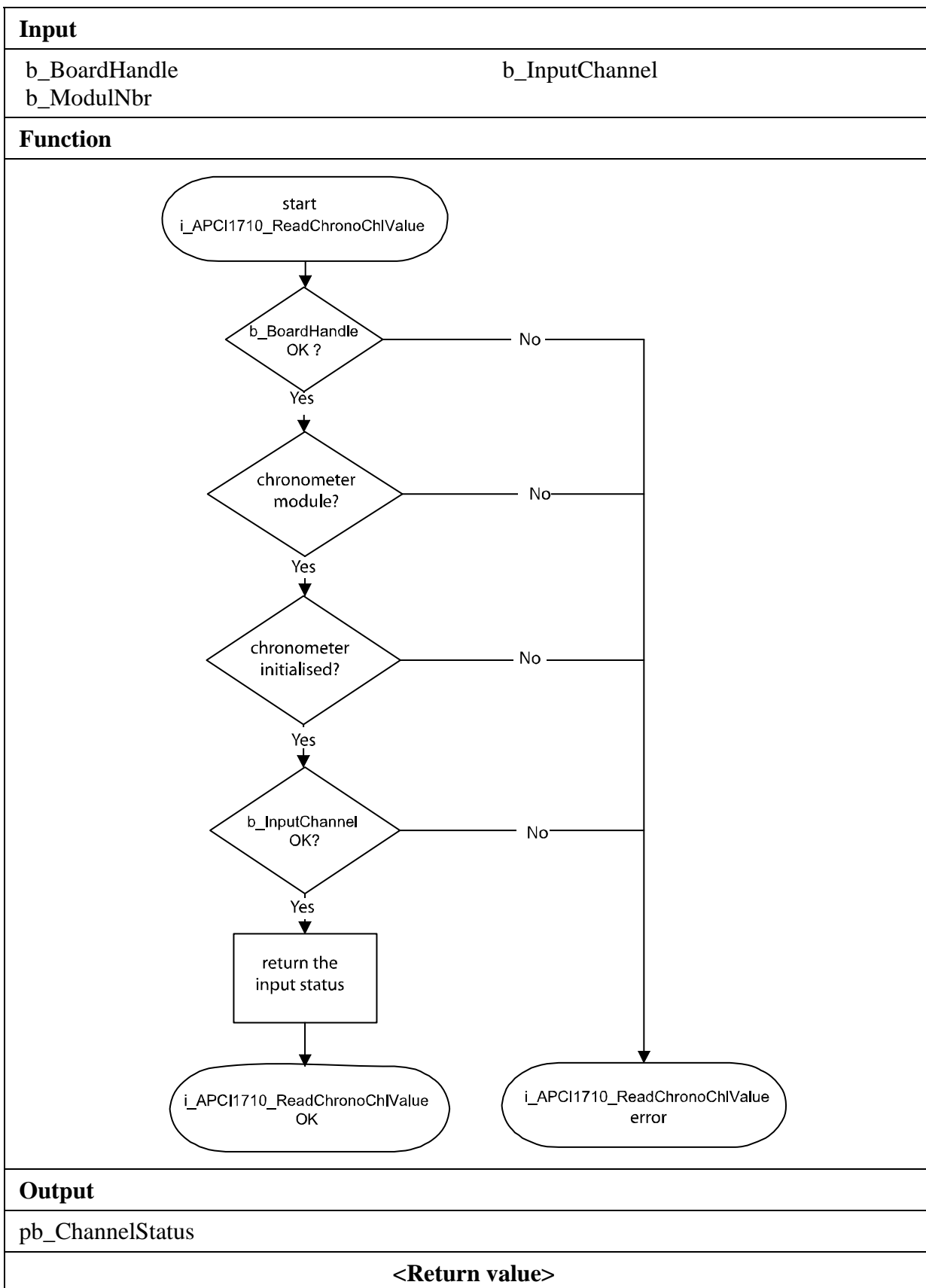
ANSI C :

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

i_ReturnValue = i_APCI1710_ReadChronoChlValue
               (b_BoardHandle,
                0,
                0,
                &b_ChannelStatus);
```

Return value:

- 0: No error
- 1: Handle parameter of the board is wrong
- 2: Selected module number is wrong.
- 3: Selected module is no "Chronometer" module.
- 4: Selected digital input is wrong.
- 5: Chronometer not initialised. See function "i_APCI1710_InitChrono".



2) `i_APCI1710_ReadChronoPortValue (...)`**Syntax:**

```
<Return Wert> = i_APCI1710_ReadChronoPortValue
                    (BYTE b_BoardHandle,
                     BYTE b_ModulNbr,
                     PBYTE pb_PortValue)
```

Parameter:**-Input:**

BYTE	b_BoardHandle	Handle of the APCI-/CPCI-1710
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)

-Output:

PBYTE	pb_PortValue	Status of the digital input port
-------	--------------	----------------------------------

Task:

Return of the status of the digital input port in the selected module (*b_ModulNbr*).

D0	D1	D2
Input E	Input F	Input G

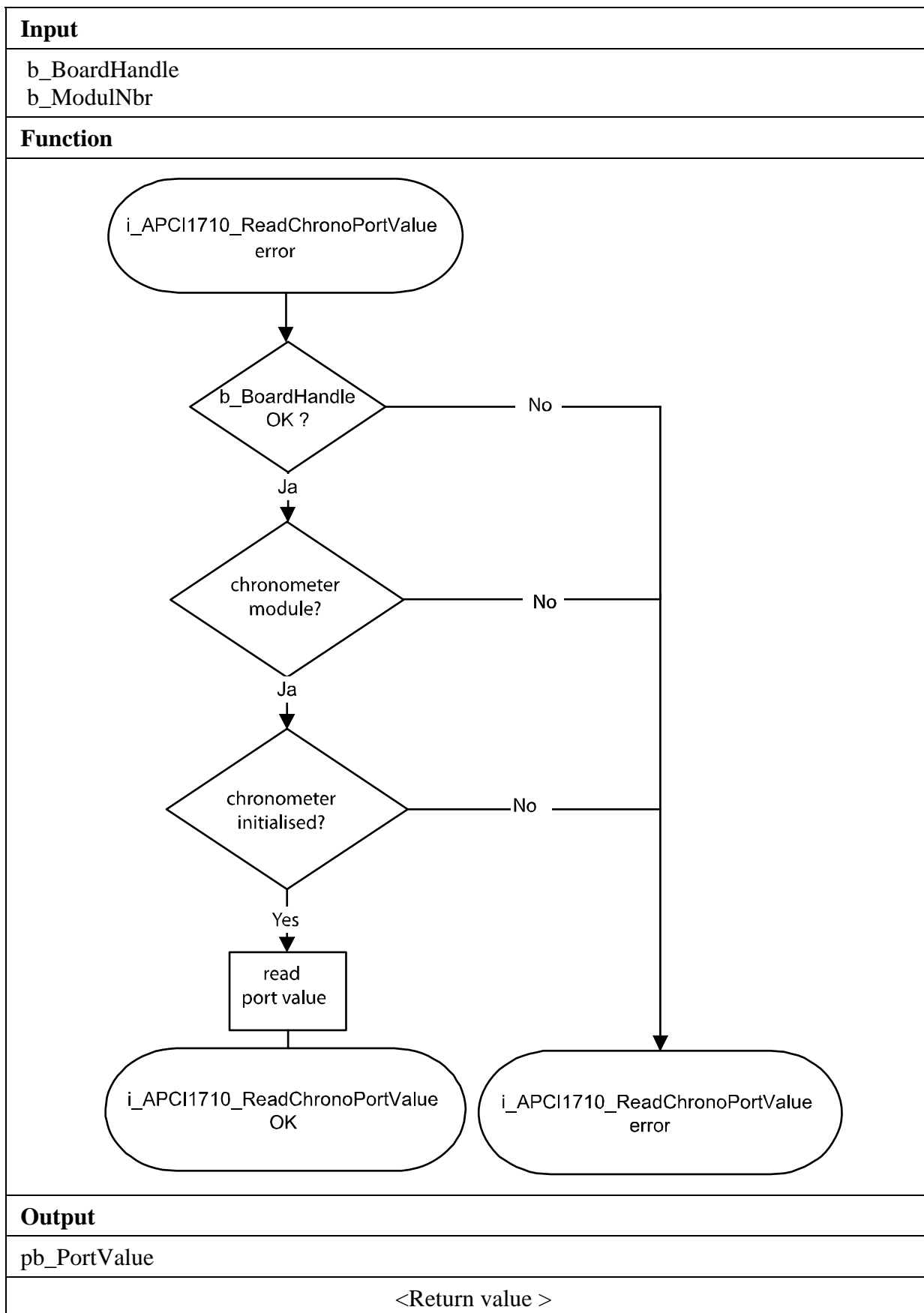
Calling convention:ANSI C:

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

```
i_ReturnValue = i_APCI1710_ReadChronoPortValue
                (b_BoardHandle,
                 0,
                 &b_PortValue);
```

Return value:

- 0: No error
- 1: Handle parameter of the board is wrong
- 2: Selected module number is wrong.
- 3: Selected module is no "Chronometer" module.
- 4: Chronometer not initialised. See function "i_APCI1710_InitChrono".



3.7 Functions in the kernel mode

i

IMPORTANT!

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

3.7.1 Reading the chronometer

1) i_APCI1710_KRNL_GetChronoProgressStatus (...)

Syntax:

```
<Return Wert> = i_APCI1710_KRNL_GetChronoProgressStatus
                (UINT          ui_BaseAddress,
                 BYTE          b_ModulNbr,
                 PBYTE pb_ChronoStatus)
```

Parameter:

-Input:

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

-Output:

PULONG	pb_ChronoStatus	Return of the chronometer status. 0: Measurement not started. No start signal arrived. 1: Measurement started. A start signal has arrived. 2: Measurement stopped. A Stopp signal has arrived. The measurement is over. 3: An overflow occurred. Please change the time base with the function "i_APCI1710_InitChrono".
--------	-----------------	---

Task:

Returns the chronometer status (*pb_ChronoStatus*) of the selected module (*b_ModulNbr*).

Calling convention:

ANSI C:

```
int          i_ReturnValue;
unsigned int  ui_BaseAddress;
unsigned char b_ChronoStatus;
```

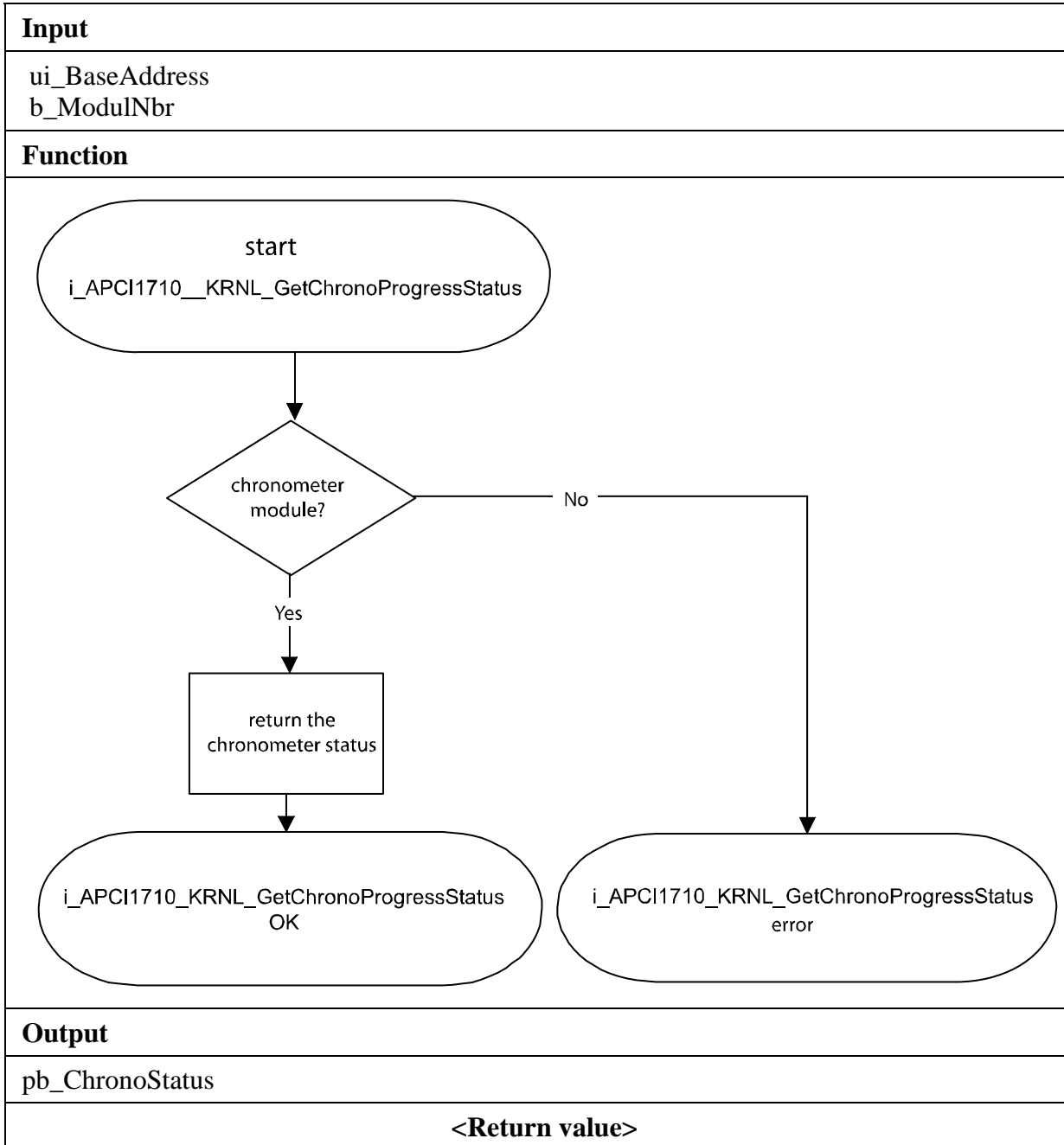
```
i_ReturnValue = i_APCI1710_KRNL_GetChronoProgressStatus
                (ui_BaseAddress,
                 0,
                 &pb_ChronoStatus);
```

Return value

0: No error

-1: Selected module number is wrong.

-2: The selected module is no "Chronometer" module.



2) **i_APCI1710_KRNL_ReadChronoValue (...)****Syntax:**

<Return Wert> = i_APCI1710_KRNL_ReadChronoValue
 (UINT ui_BaseAddress,
 BYTE b_ModulNbr,
 PBYTE pb_ChronoStatus,
 PULONG pul_ChronoValue)

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)

-Output:

PBYTE	pb_ChronoStatus	Return of the chronometer status. 0: Measurement not started. No start signal arrived. 1: Measurement started. A start signal has arrived. 2: Measurement stopped. A stop signal has arrived. The measurement is over and <i>pul_ChronoValue</i> returns the chronometer time. 3: An overflow occurred. Please change the time base with the function "i_APCI1710_InitChrono"
PULONG	pul_ChronoValue	Chronometer-Zeitwert.

Task:

Return of the chronometer status (*pb_ChronoStatus*) and the time value (*pul_ChronoValue*) after a stop signal to the selected module (*b_ModulNbr*). This function is only available if you disabled the interrupt function. See function "i_APCI1710_EnableChrono" and table 3-4.

The chronometer status can be tested with the function "i_APCI1710_KRNL_GetChronoProgressStatus".

The returned value by *pul_ChronoValue* is not the correct time value. Use the function "i_APCI1710_ConvertChronoValue".

Otherwise, the following formula is applied in order to calculate the correct time value:

Time value = *pul_ChronoValue* x *pul_RealTimeInterval*.

pul_RealTimeInterval ist he returned value from "i_APCI1710_InitChrono". The time unit is the variable *b_TimeUnit* of the function "i_APCI1710_InitChrono".

Calling convention:ANSI C :

```
int          i_ReturnValue;  
unsigned int ui_BaseAddress;  
unsigned char b_ChronoStatus;  
unsigned long ul_ChronoValue;
```

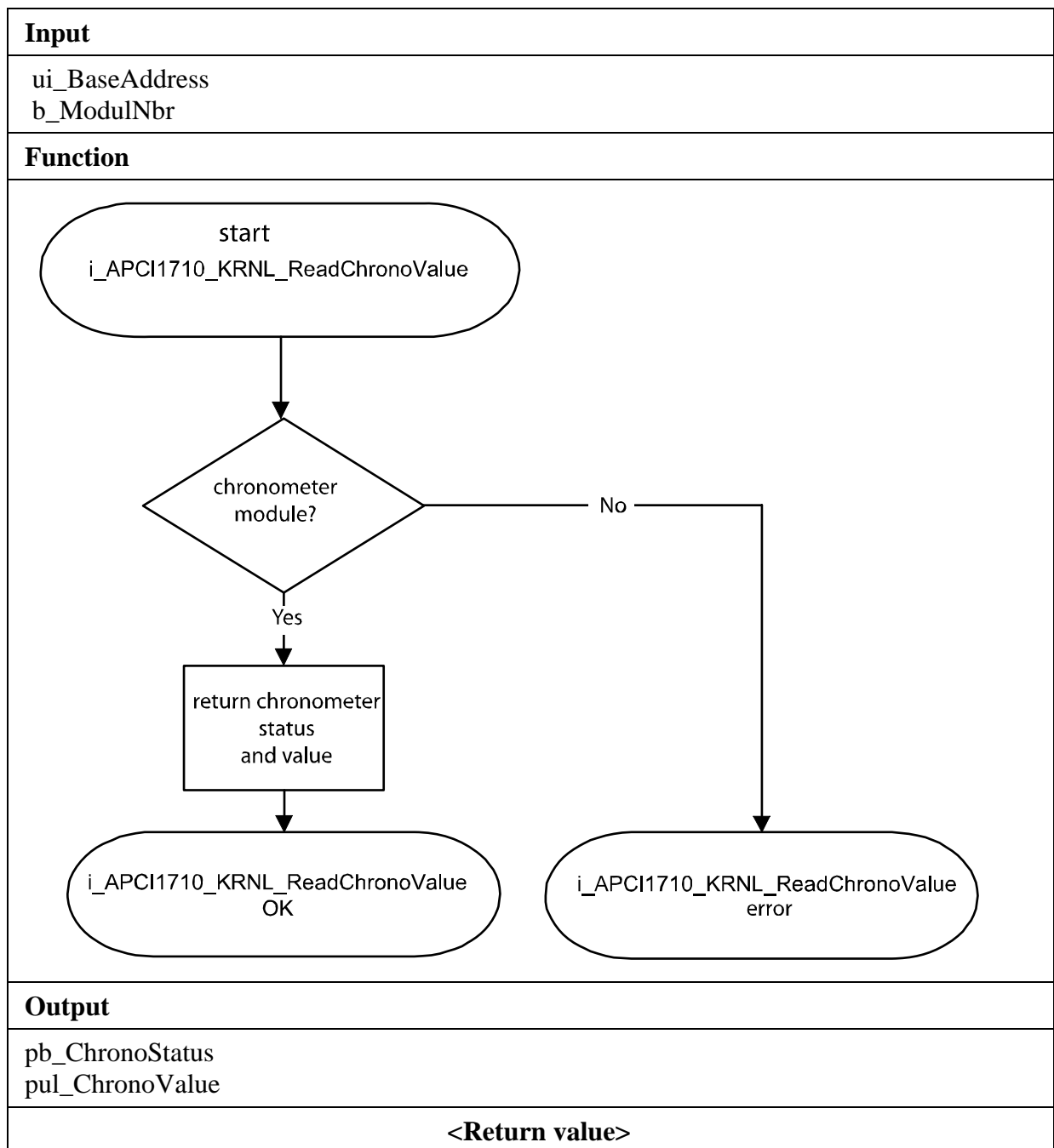
```
i_ReturnValue = i_APCI1710_KRNL_ReadChronoValue  
                (ui_BaseAddress,  
                 0,  
                 &pb_ChronoStatus,  
                 &ul_ChronoValue);
```

Return value:

0: No error

-1: Selected module number is wrong.

-2: The selected module is no "Chronometer" module.



3.7.2 Writing on a digital output

3) i_APCI1710_KRNL_SetChronoChlOn (...)

Syntax:

```
<Return Wert> = i_APCI1710_KRNL_SetChronoChlOn
                    (UINT  ui_BaseAddress,
                     BYTE  b_ModulNbr,
                     BYTE  b_OutputChannel)
```

Parameter:

-Input:

UINT	ui_BaseAddress	Base address of the APCI-/CPCI-1710 board
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_OutputChannel	Selection of a digital output (0 to 2) 0: Output H 1: Output A 2: Output B

-Output:

There is no output.

Task:

Sets the output entered by b_OutputChannel. Setting an output means setting the output on "High".

Calling convention:

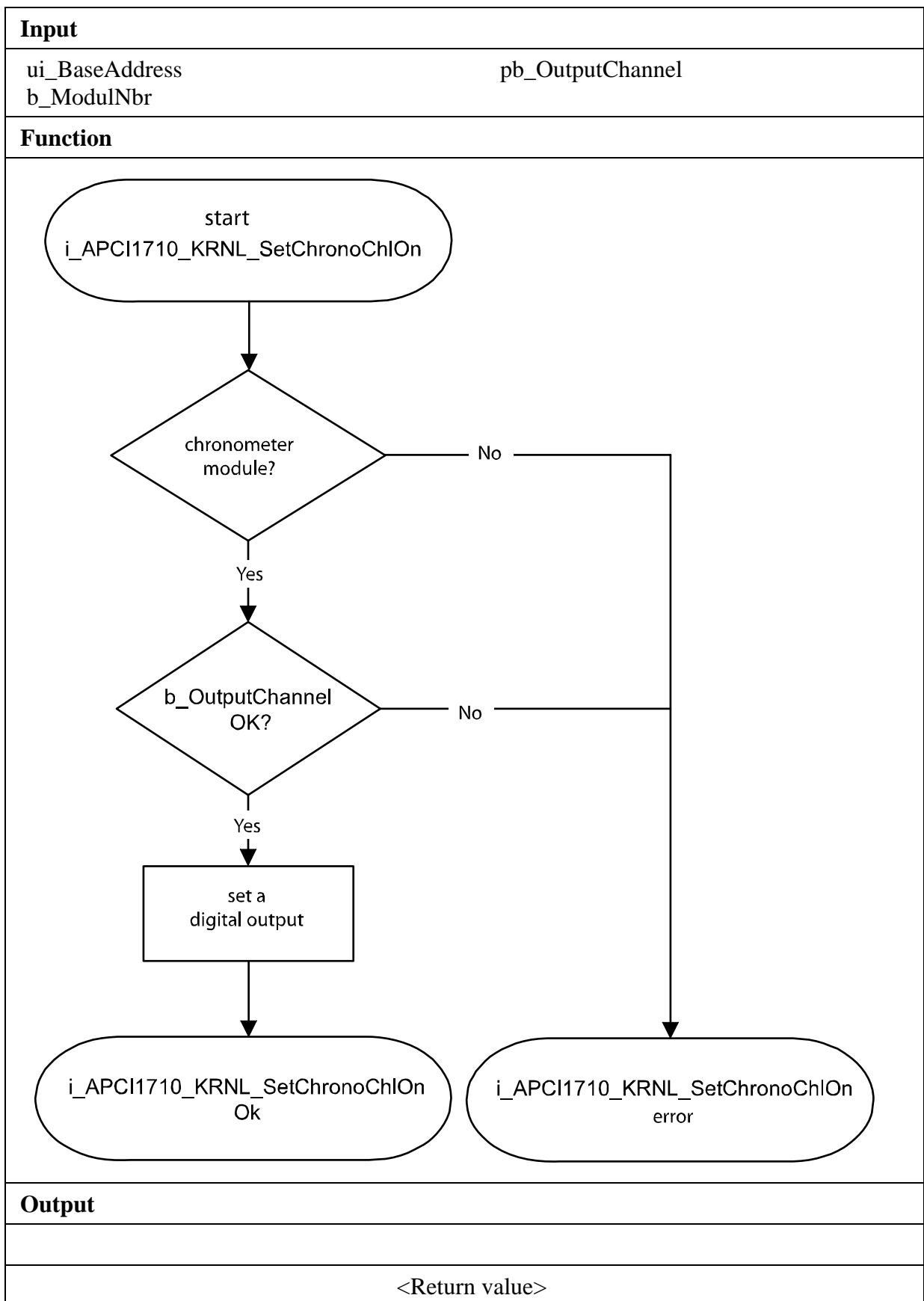
ANSI C:

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

```
i_ReturnValue = i_APCI1710_KRNL_SetChronoChlOn
                (ui_BaseAddress,
                 0,
                 0);
```

Return value

0: No error
-1: Selected module number is wrong.
-2: The selected module is no "Chronometer" module.
-3: The selected digital output is wrong.



4) `i_APCI1710_KRNL_SetChronoChlOff (...)`**Syntax:**

```
<Return Wert> = i_APCI1710_KRNL_SetChronoChlOff
                    (UINT          ui_BaseAddress,
                     BYTE          b_ModulNbr,
                     BYTE          b_OutputChannel)
```

Parameter:**-Input:**

UINT	ui_BaseAddress	Base address of the APCI-/CPCI-1710 board
BYTE	b_ModulNbr	Number of the module to be configured (0 to 3)
BYTE	b_OutputChannel	Selection of a digital output (0 to 2) 0: Output H 1: Output A 2: Output B

-Output :

There is no output.

Task:

Resets the output entered by `b_OutputChannel`. Resetting an output means setting an output to "Low".

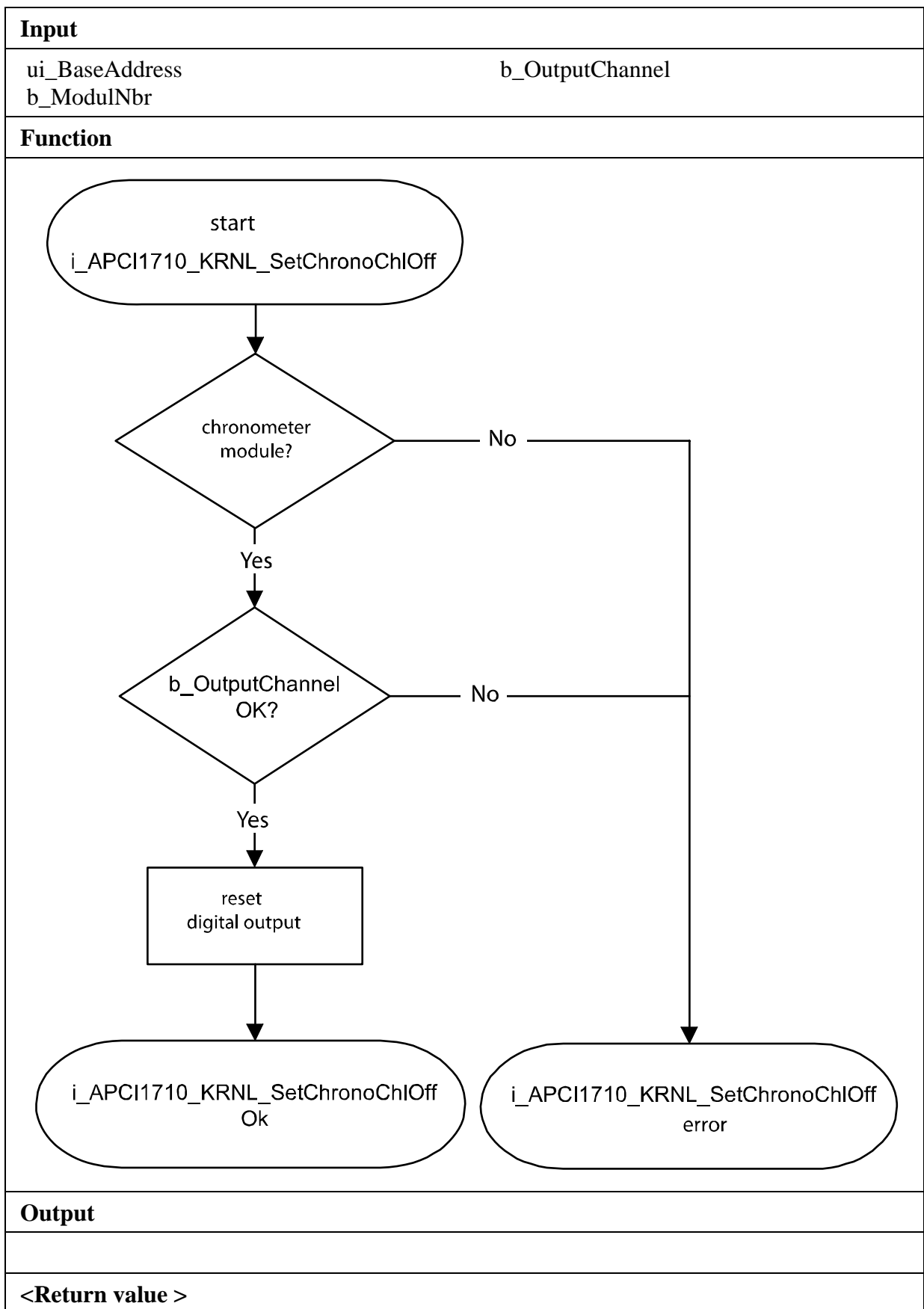
Calling convention:ANSI C:

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

```
i_ReturnValue = i_APCI1710_KRNL_SetChronoChlOff
                (ui_BaseAddress,
                 0,
                 0);
```

Return value:

0: No error
-1: Selected module number is wrong.
-2: The selected module is no "Chronometer"-module.
-3: The selected digital output is wrong.



3.7.3 Reading a digital input

5) i_APCI1710_KRNL_ReadChronoChIValue (...)

Syntax:

```
<Return Wert> = i_APCI1710_KRNL_ReadChronoChIValue
                    (UINT ui_BaseAddress,
                     BYTE b_ModulNbr,
                     BYTE b_InputChannel,
                     PBYTE pb_ChannelStatus)
```

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_InputChannel	Selection of a digital input (0 to 2) 0: Input E 1: Input F 2: Input G

-Output :

PBYTE	pb_ChannelStatus	Status of the digital input. 0: Channel enabled 1: Channel disabled
-------	------------------	---

Task:

Return of the status of the digital input (*b_InputChannel*) in the selected module (*b_ModulNbr*).

Calling convention:

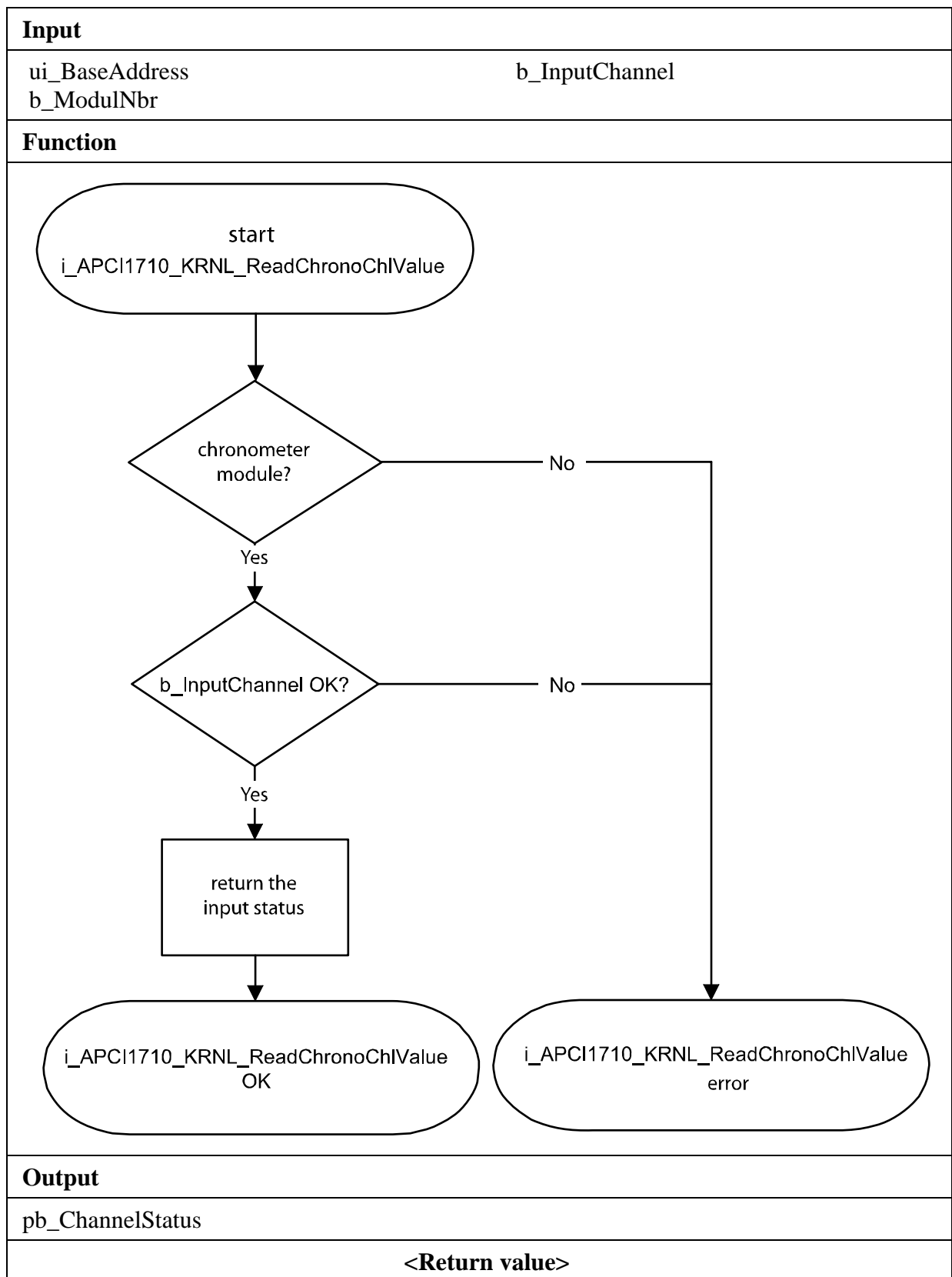
ANSI C:

```
int          i_ReturnValue;
unsigned int  ui_BaseAddress;
unsigned char b_ChannelStatus;
```

```
i_ReturnValue = i_APCI1710_KRNL_ReadChronoChIValue
                (ui_BaseAddress,
                 0,
                 0,
                 &b_ChannelStatus);
```

Return value:

0: No error
-1: Selected module number is wrong.
-2: Selected module is no "Chronometer"-module.
-3: The selected digital input is wrong.



6) **i_APCI1710_KRNL_ReadChronoPortValue (...)****Syntax:**

```
<Return Wert> = i_APCI1710_KRNL_ReadChronoPortValue
                    (UINT          ui_BaseAddress,
                     BYTE          b_ModulNbr,
                     PBYTE pb_PortValue)
```

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)
```

-Output:

```
PBYTE   pb_PortValue      Status of the digital input port
```

Task:

Returns the status of the digital input port in the selected module (*b_ModulNbr*).

D0	D1	D2
Input E	Input F	Input G

Calling convention:ANSI C:

```
int          i_ReturnValue;
unsigned int  ui_BaseAddress;
unsigned char b_PortValue;
```

```
i_ReturnValue = i_APCI1710_KRNL_ReadChronoPortValue
                (ui_BaseAddress,
                 0,
                 &b_PortValue);
```

Return value

```
0: No error
-1: Selected module number is wrong.
-2: The selected module is no "Chronometer" module.
```

