



ISO 9001 certified



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

Technical description

ADDIALOG APCI-/CPCI-3120

Standard software / API functions

Edition: 06.01-07/2005

1	INTRODUCTION.....	4
2	NORM DIN 66001 FOR PROGRAM OPERATION	8
3	SOFTWARE FUNCTIONS (API)	9
3.1	Initialisation.....	9
	1) i_APCI3120_InitCompiler (..)	9
	2) i_APCI3120_CheckAndGetPCISlotNumber (...)	11
	3) i_APCI3120_SetBoardInformation (...).....	13
	4) i_APCI3120_GetHardwareInformation (...)	15
	5) i_APCI3120_CloseBoardHandle (..)	17
3.2	Interrupt.....	19
	1) i_APCI3120_SetBoardIntRoutineDos (..)	19
	2) i_APCI3120_SetBoardIntRoutineVBDOs (..).....	23
	3) i_APCI3120_SetBoardIntRoutineWin16 (..).....	26
	4) i_APCI3120_SetBoardIntRoutineWin32 (..).....	29
	5) i_APCI3120_TestInterrupt (..).....	36
	6) i_APCI3120_ResetBoardIntRoutine (..).....	39
3.3	Direct conversion of analog input channels.....	41
	1) i_APCI3120_Read1AnalogInput (...)	41
	2) i_APCI3120_ReadMoreAnalogInput (...).....	46
3.4	Cyclic conversion of analog input channels.....	50
	1) i_APCI3120_InitAnalogInputAcquisition (...)	50
	2) i_APCI3120_StartAnalogInputAcquisition (...)	61
	3) i_APCI3120_StopAnalogInputAcquisition (...).....	64
	4) i_APCI3120_ClearAnalogInputAcquisition(...)	67
3.5	Analog output channels	69
	1) i_APCI3120_Write1AnalogValue (...)	69
	2) i_APCI3120_WriteMoreAnalogValue (...).....	71
3.6	Timer.....	74
	1) i_APCI3120_InitTimerWatchdog (...)	74
	2) i_APCI3120_StartTimerWatchdog (...)	77
	3) i_APCI3120_StopTimerWatchdog (...)	79
	4) i_APCI3120_ReadTimer (...).....	81
	5) i_APCI3120_WriteTimer (...).....	83
	6) i_APCI3120_ReadWatchdogStatus (...)	85
3.7	Digital input channels	88
	1) i_APCI3120_Read1DigitalInput (...)	88
	2) i_APCI3120_Read4DigitalInput (...)	90
3.8	Digital output channels.....	92
	1) i_APCI3120_Set1DigitalOutputOn (...).....	92
	2) i_APCI3120_Set1DigitalOutputOff (...).....	94
	3) i_APCI3120_Set4DigitalOutputOn (...).....	96
	4) i_APCI3120_Set4DigitalOutputOff (...).....	98
	5) i_APCI3120_SetOutputMemoryOn (...).....	100
	6) i_APCI3120_SetOutputMemoryOff (...)	102

3.9 Functions to be used in Kernel mode.....104

1) i_APCI3120_KRNL_Read1DigitalInput (...)	104
2) i_APCI3120_KRNL_Read4DigitalInput (...)	106
3) i_APCI3120_KRNL_Set1DigitalOutputOn (...)	108
4) i_APCI3120_KRNL_Set4DigitalOutputOn (...)	110
5) i_APCI3120_KRNL_Write1AnalogValue (...)	112

Tables

Table 1-1: Type Declaration for Dos and Windows 3.1X	4
Table 1-2: Type Declaration for Windows 95/NT	5
Table 1-3: Define value	6
Table 1-3: Define value (continued)	7
Table 3-1: Values returned for the analog input channels	20
Table 3-2: Interrupt mask	21
Table 3-3: Selection of the analog input channels	43
Table 3-4: Gain selection	43
Table 3-5: Selection of the input voltage range	43
Table 3-6: Selection of the output voltage range	71

1 INTRODUCTION

i

IMPORTANT!

Note the following conventions in the text:

Function: "i_APCI3120_SetBoardInformation"
Variable *ui_Address*

Table 1-1: Type Declaration for Dos and Windows 3.1X

	Borland C	Microsoft C	Borland Pascal	Microsoft Visual Basic Dos	Microsoft Visual Basic Windows
VOID	void	void	pointer		any
BYTE	unsigned char	unsigned char	byte	integer	integer
INT	int	int	integer	integer	integer
UINT	unsigned int	unsigned int	word	long	long
LONG	long	long	longint	long	long
PBYTE	unsigned char *	unsigned char *	var byte	integer	integer
PINT	int *	int *	var integer	integer	integer
PUINT	unsigned int *	unsigned int *	var word	long	long
PCHAR	char *	char *	var string	string	string

Table 1-2: Type Declaration for Windows 95/NT

	Borland C	Microsoft C	Borland Pascal	Microsoft Visual Basic Dos	Microsoft Visual Basic Windows
VOID	void	void	pointer		any
BYTE	unsigned char	unsigned char	byte	integer	integer
INT	int	int	integer	integer	integer
UINT	unsigned int	unsigned int	long	long	long
LONG	long	long	longint	long	long
PBYTE	unsigned char *	unsigned char *	var byte	integer	integer
PINT	int *	int *	var integer	integer	integer
PUINT	unsigned int *	unsigned int *	var long	long	long
PCHAR	char *	char *	var string	string	string

Table 1-3: Define value

Define name	Decimal value	Hexadecimal value
DLL_COMPILER_C	0	1
DLL_COMPILER_PASCAL	1	2
DLL_COMPILER_VB	2	3
DLL_LABVIEW	3	4
DLL_COMPILER_VB5	4	4
APCI3120_DISABLE	0	0
APCI3120_ENABLE	1	1
APCI3120_CHANNEL0	0	0
APCI3120_CHANNEL1	1	1
APCI3120_CHANNEL2	2	2
APCI3120_CHANNEL3	3	3
APCI3120_CHANNEL4	4	4
APCI3120_CHANNEL5	5	5
APCI3120_CHANNEL6	6	6
APCI3120_CHANNEL7	7	7
APCI3120_CHANNEL8	8	8
APCI3120_CHANNEL9	9	9
APCI3120_CHANNEL10	10	A
APCI3120_CHANNEL11	11	B
APCI3120_CHANNEL12	12	C


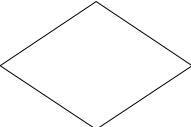

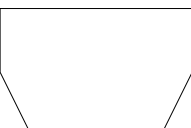

Table 1-3: Define value (continued)

Define name	Decimal value	Hexadecimal value
APCI3120_CHANNEL13	13	D
APCI3120_CHANNEL14	14	E
APCI3120_CHANNEL15	15	F
APCI3120_1_GAIN	0	0
APCI3120_2_GAIN	16	10
APCI3120_5_GAIN	64	20
APCI3120_10_GAIN	48	30
APCI3120_UNIPOLAR	128	80
APCI3120_BIPOLAR	0	0
APCI3120_SIMPLE_MODUS	0	0
APCI3120_DELAY_MODUS	1	1
APCI3120_DELAY_1_MODUS	2	2
APCI3120_DMA_NOT_USED	0	0
APCI3120_DMA_USED	1	1
APCI3120_SINGLE	0	0
APCI3120_CONTINUOUS	1	1
APCI3120_ASYNCHRONOUS_MODE	0	0
APCI3120_SYNCHRONOUS_MODE	1	1
APCI3120_TIMER	1	1
APCI3120_WATCHDOG	2	2

2 NORM DIN 66001 FOR PROGRAM OPERATION

All the API software functions necessary to the operation of the board **APCI-/CPCI-3120** are listed in the following chapter (Device driver).

Functions diagrams have been designed with the following symbols. The user is hence able to follow the different steps of the driver functions.

	Process, general (including inputs and outputs)
	Decision Selection unit
	Loop limit Beginning
	Loop limit End
	Terminator (eg. Beginning or end of a sequence, origin or place of data)

Remark: The **CPCI-3120** board is compatible with the **APCI-3120** board regarding the standard software installation. The program ADDIREG will thus make no difference between the systems (PCI board or **CompactPCI** board).

The API functions of the standard software are also the same.

In the following chapter, a common name is used for both boards: **xPCI-3120**

3 SOFTWARE FUNCTIONS (API)

3.1 Initialisation

1) i_APCI3120_InitCompiler (..)

Syntax:

<Return value> = i_APCI3120_InitCompiler (BYTE b_CompilerDefine)

Parameters:**- Input:**

BYTE b_CompilerDefine

The user has to choose the language under Windows in which he/she wants to program

- DLL_COMPILER_C:

The user programs in C.

- DLL_COMPILER_VB:

The user programs in Visual Basic for Windows.

- DLL_COMPILER_VB_5:

The user programs in Visual Basic 5 for Windows.

- DLL_COMPILER_PASCAL:

The user programs in Pascal or Delphi.

- DLL_LABVIEW :

The user programs in Labview.

- Output:

No output signal has occurred.

Task:

If you want to use the DLL functions, choose the language in which you want to program. This function must be the first to be called up.

i**IMPORTANT!**

This function is only available with a Windows environment.

Calling convention:

ANSI C:

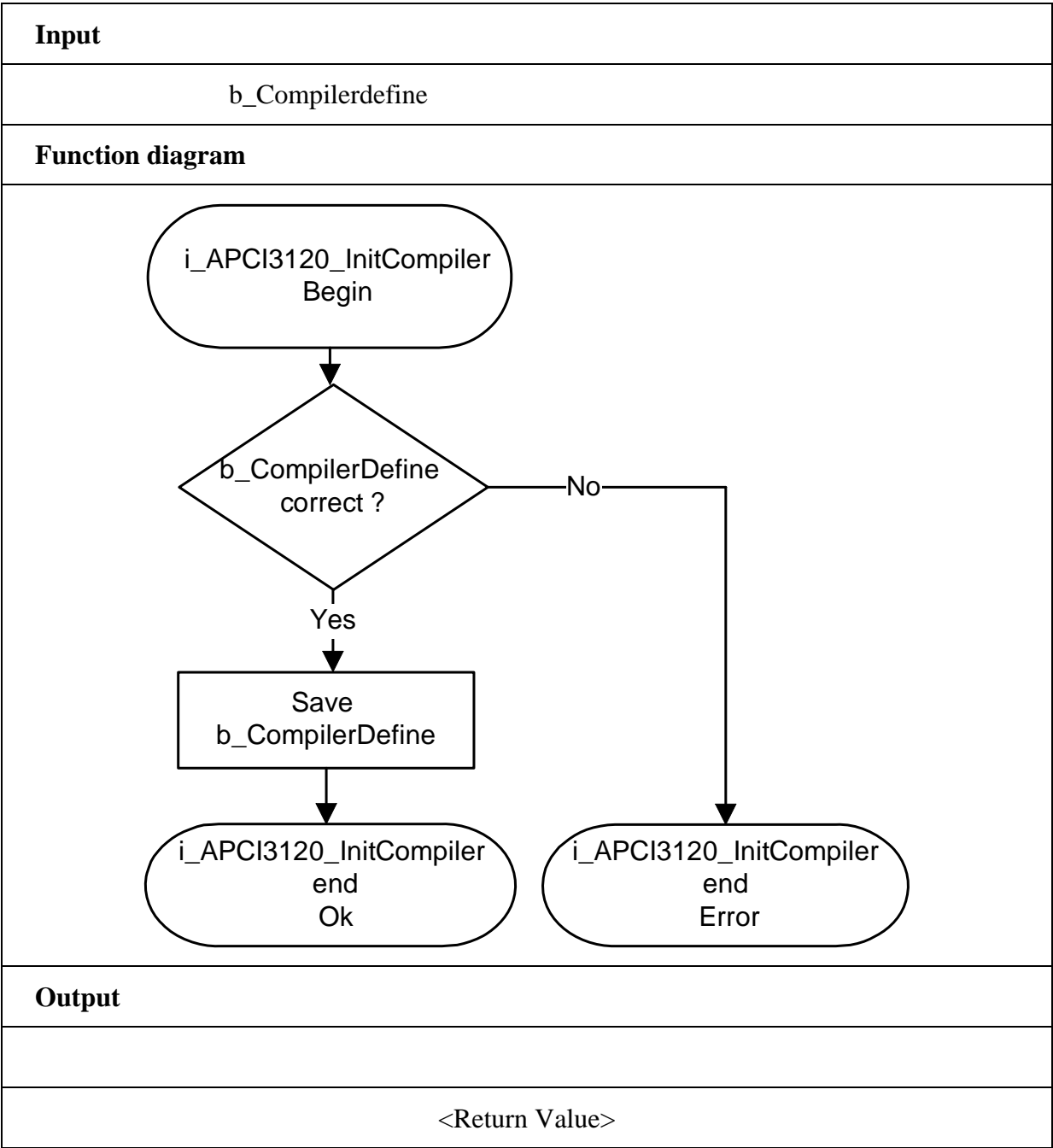
int i_ReturnValue;

i_ReturnValue = i_APCI3120_InitCompiler (DLL_COMPILER_C);

Return value:

0: No error

-1: Compiler parameter is wrong



2) i_APCI3120_CheckAndGetPCISlotNumber (...)**Syntax:**

<Return value> = i_APCI3120_CheckAndGetPCISlotNumber
(PBYTE pb_SlotNumberArray)

Parameters:**- Input:**

No input signal occurs

- Output:

PBYTE pb_SlotNumberArray Slot number list.

Task:

Checks all **xPCI-3120** and returns the slot number of each **xPCI-3120** board. Each *pb_SlotNumberArray* parameter contains the slot number (1 to 10) of 1 **xPCI-3120** board.

Calling convention:

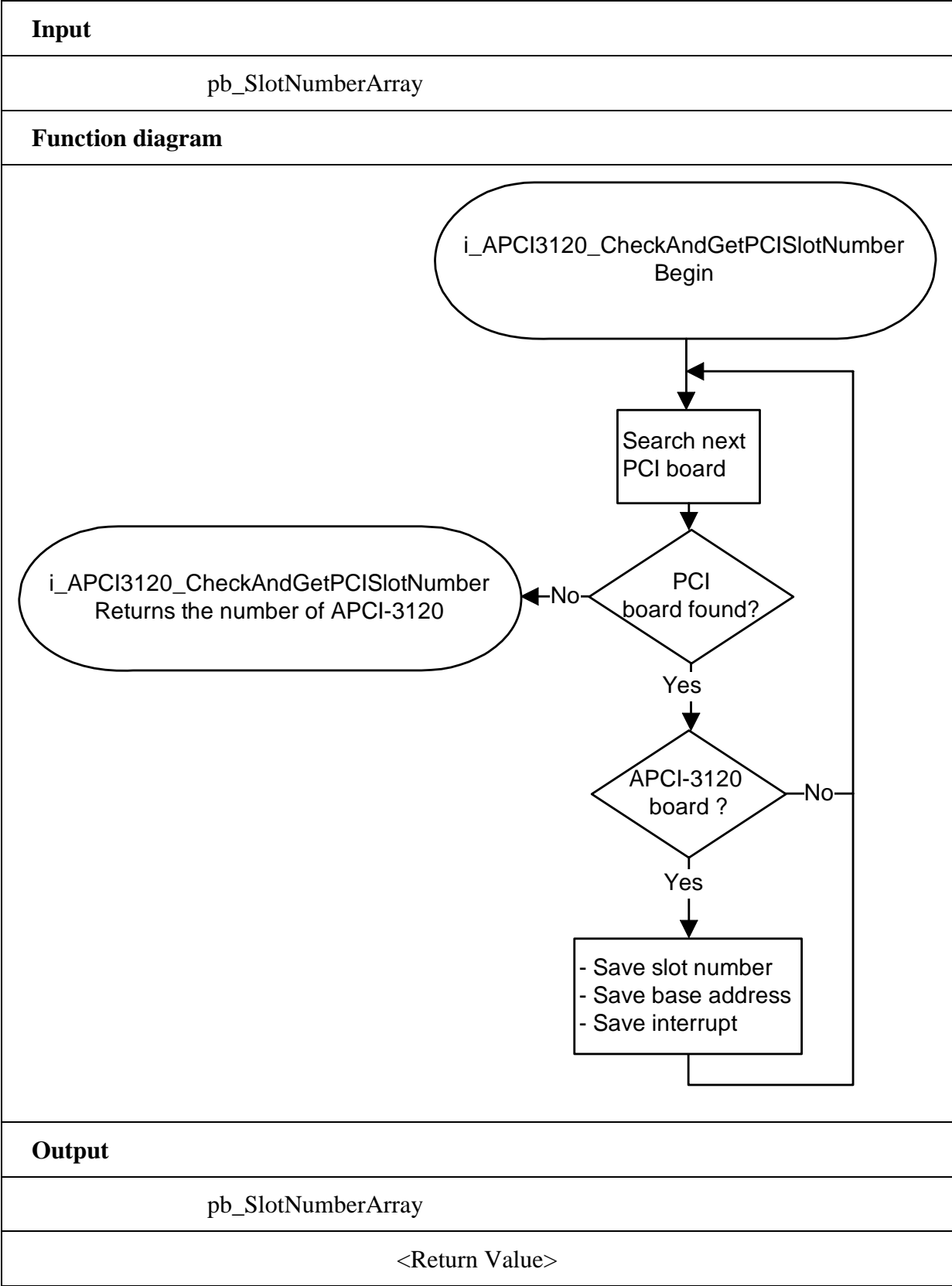
ANSI C :

```
int          i_ReturnValue;  
unsigned char b_SlotNumberArray [10];
```

```
i_ReturnValue = i_APCI3120_CheckAndGetPCISlotNumber  
                (b_SlotNumberArray);
```

Return value:

Returns the number of **xPCI-3120** which are inserted in your PC.
If the return value equals 0 then no **xPCI-3120** was found in your PC.



3) i_APCI3120_SetBoardInformation (...)**Syntax:**

```
<Return value> = i_APCI3120_SetBoardInformation
                    (BYTE    b_SlotNumber,
                     BYTE    b_AnalogInputChannelNbr,
                     BYTE    b_AnalogOutputChannelNbr,
                     PBYTE   pb_BoardHandle)
```

Parameters:**- Input:**

BYTE	b_SlotNumber	Slot number of board xPCI-3120
BYTE	b_AnalogInputChannelNbr	Number of analog input channels (0 to 16)
BYTE	b_AnalogOutputChannelNbr	Number of analog output channels (0 to 8)

- Output:

PBYTE	pb_BoardHandle	Handle ¹ of board xPCI-3120 for using the functions
-------	----------------	--

Task:

Verifies if board **xPCI-3120** is present. Stores the following information:

- slot number,
- the number of analog input channels
- and the number of analog output channels.

A handle is returned to the user which allows to use the next functions.
Handles allow to operate several boards.

Calling convention:ANSI C :

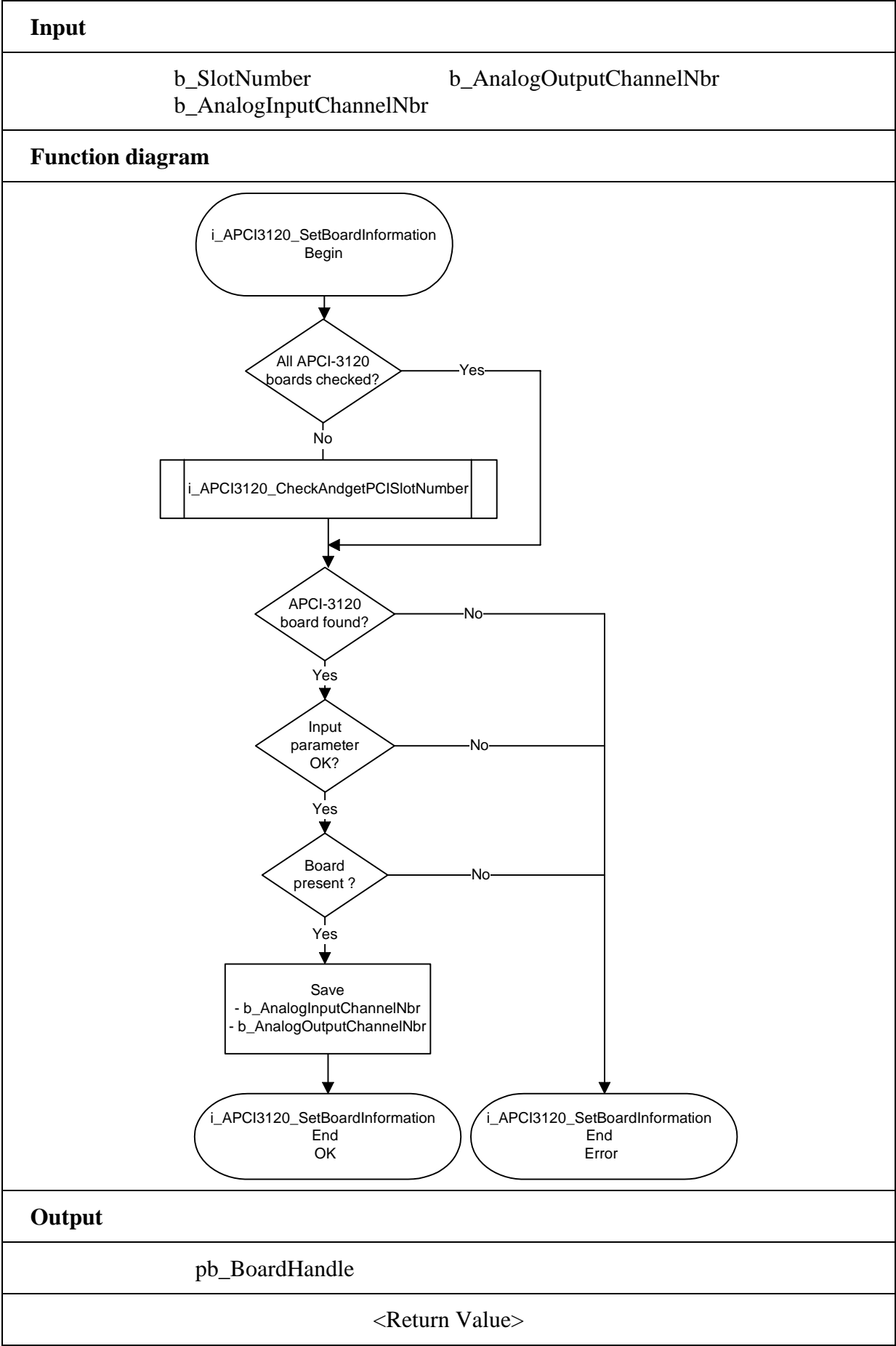
```
int          i_ReturnValue;
unsigned char b_BoardHandle;
```

```
i_ReturnValue = i_APCI3120_SetBoardInformation (1, 16, 8,
                                                &b_BoardHandle);
```

Return value:

- 0: No error
- 1: Slot number not available
- 2: Board not present
- 3: Number of analog input channels is wrong
- 4: Number of analog output channels is wrong.
- 5: No handle is available for the board (up to 10 handles can be used)

¹ Identification number of the board



4) i_APCI3120_GetHardwareInformation (...)**Syntax:**

```
<Return value> = i_APCI3120_GetHardwareInformation
                    (BYTE   b_BoardHandle,
                     PUINT  pui_BaseAddress,
                     PBYTE  pb_InterruptNbr,
                     PBYTE  pb_SlotNumber)
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of the board
------	---------------	---------------------

- Output:

PUINT	pui_BaseAddress	Base address of the xPCI-3120
PBYTE	pb_InterruptNbr	Interrupt channel of the xPCI-3120
PBYTE	pb_SlotNumber	Slot number of the xPCI-3120

Task:

Returns the base address, the interrupt and slot number of the **xPCI-3120**.

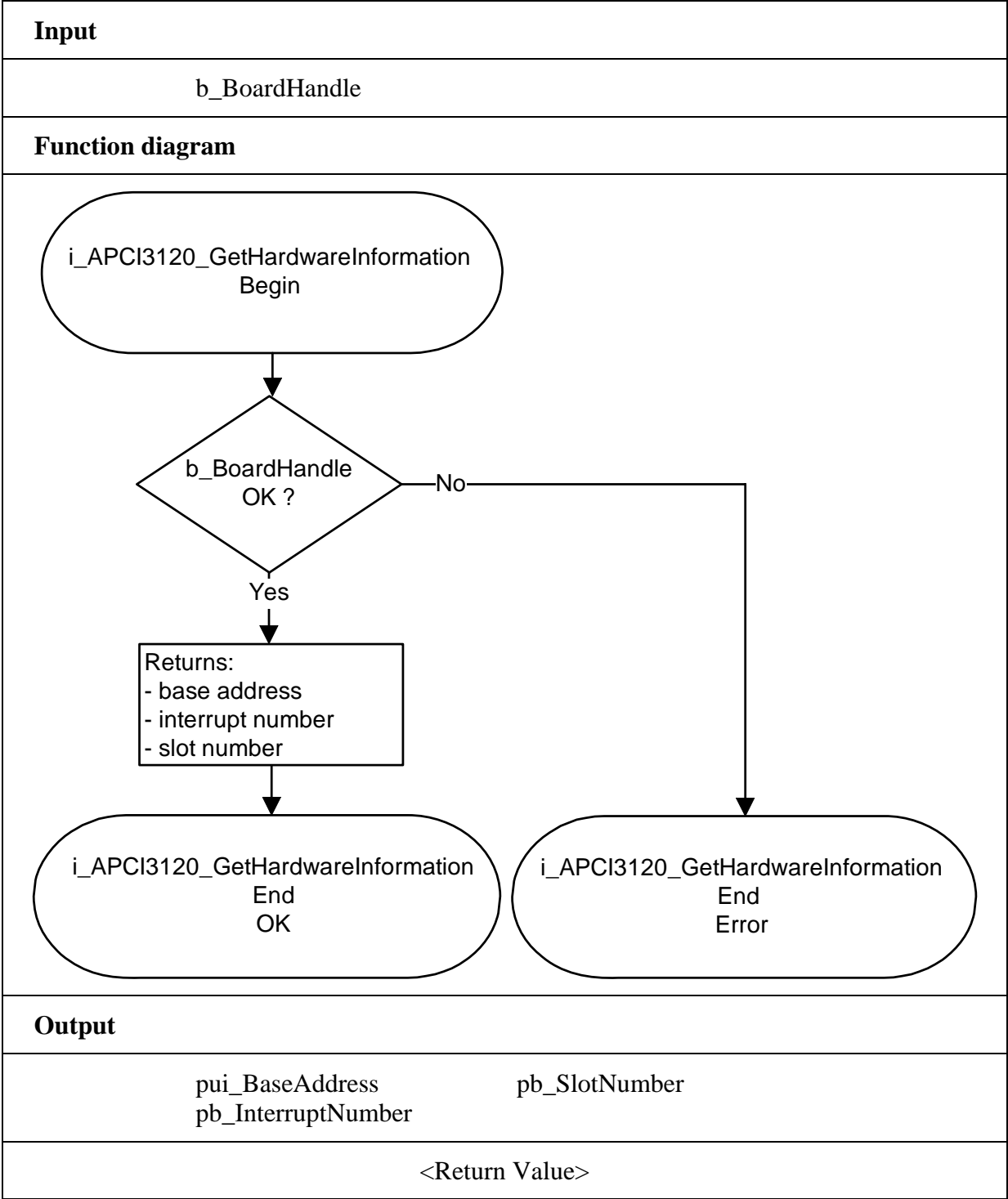
Calling convention:ANSI C :

```
int          i_ReturnValue;
unsigned int  ui_BaseAddress;
unsigned char b_InterruptNbr;
unsigned char b_SlotNumber;
unsigned char b_BoardHandle;
```

```
i_ReturnValue = i_APCI3120_GetHardwareInformation
                (b_BoardHandle,
                 &ui_BaseAddress,
                 &b_InterruptNbr,
                 &b_SlotNumber);
```

Return value:

0: No error
-1: The handle parameter of the board is wrong



5) i_APCI3120_CloseBoardHandle (..)**i****IMPORTANT!****Call up this function each time you want to quit the user program!****Syntax:**

<Return value> = i_APCI3120_CloseBoardHandle (BYTE b_BoardHandle)

Parameters:**- Input:**

BYTE b_BoardHandle Handle of the board

- Output:

No output signal has occurred

Task:

Releases the handle of the board. Blocks the access to the board.

Calling convention:ANSI C :

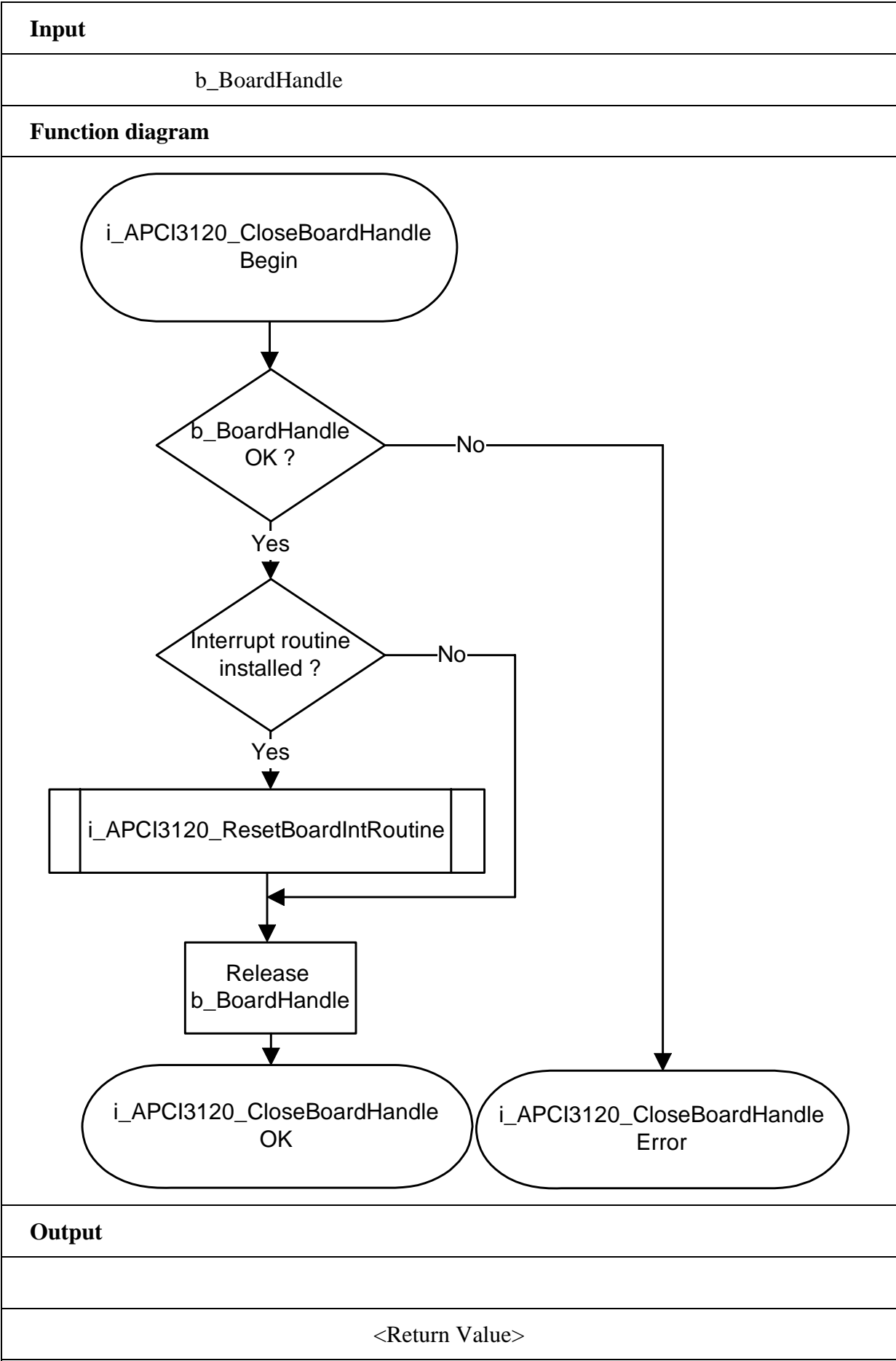
```
int                      i_ReturnValue;  
unsigned char    b_BoardHandle;
```

```
i_ReturnValue = i_APCI3120_CloseBoardHandle    (b_BoardHandle);
```

Return value:

0: No error

-1: The handle parameter of the board is wrong



3.2 Interrupt

i

IMPORTANT!

This function is only available for C/C++ and Pascal for DOS.

1) i_APCI3120_SetBoardIntRoutineDos (..)

Syntax:

```
<Return value> = i_APCI3120_SetBoardIntRoutineDos
                (BYTE    b_BoardHandle,
                 VOID     v_FunctionName
                 (BYTE    b_BoardHandle,
                  BYTE    b_InterruptMask,
                  PUINT   pui_AnalogInputValue))
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of the board
VOID	v_FunctionName	Name of the user interrupt routine

- Output:

No output signal has occurred

Task:

This function must be called up for each **xPCI-3120** on which an interrupt action is to be enabled.

First calling (first board):

- the user interrupt routine is installed
- interrupts are enabled.

If you operate several boards **xPCI-3120** which have to react to interrupts, call up the function as often as you operate boards **xPCI-3120**. The variable *v_FunctionName* is only relevant **for the first calling**.

From the second call of the function (next board):

- interrupts are enabled.

Interrupt

The user interrupt routine is called up by the system when an interrupt is generated.

If several boards are operated and if they have to react to interrupts, the variable *b_BoardHandle* returns the identification number (handle) of the board which has generated the interrupt.

The user interrupt routine must have the following syntax:

```
VOID  v_FunctionName (BYTE    b_BoardHandle,
                     BYTE    b_InterruptMask,
                     PUINT   pui_AnalogInputValue)
```

<i>v_FunctionName</i>	Name of the user interrupt routine
<i>b_BoardHandle</i>	Handle of the xPCI-3120 which has generated the interrupt
<i>b_InterruptMask</i>	Mask of the events which have generated the interrupt.
<i>pui_AnalogInputValue</i>	The values of the analog input channels and of the DMA buffer are returned.

Table 3-1: Values returned for the analog input channels

Event	Variable name	Variable content
<i>b_InterruptMask</i> = 1	<i>pui_AnalogInputValue</i> [0]	Number of the last analog input channel
	<i>pui_AnalogInputValue</i> [1]	Value of the analog input channel
<i>b_InterruptMask</i> = 2	<i>pui_AnalogInputValue</i> [0]	Number of analog input channels
	<i>pui_AnalogInputValue</i> [1] up to <i>pui_AnalogInputValue</i> [<i>pui_AnalogInputValue</i> [0]] See example 1	Values of the analog input channels
<i>b_InterruptMask</i> = 4	<i>pui_AnalogInputValue</i> [0]	Number of analog input channels
	<i>pui_AnalogInputValue</i> [1] up to <i>pui_AnalogInputValue</i> [<i>pui_AnalogInputValue</i> [0]] See example 1	Values of the analog input channels
<i>b_InterruptMask</i> = 8	<i>pui_AnalogInputValue</i>	Values of the DMA buffer

Example 1 *pui_AnalogInputValue* [0] = 3

3	= Number of analog input channels
1	= Analog value 0
2	= Analog value 1
3	= Analog value 2

Table 3-2: Interrupt mask

Mask	Meaning
0000 0001	End of Conversion (EOC)
0000 0010	Conversion driven by timer is completed
0000 0100	Conversion of a group of channels is completed (EOS)
0000 1000	DMA conversion cycle is completed
0001 0000	Timer 2 has run down
0010 0000	Analog output channels - Watchdog has run down

The user can give another name for *v_FunctionName*, *b_BoardHandle*, *b_InterruptMask*, *pui_AnalogInputValue*.

Calling convention:

ANSI C :

```

void    v_FunctionName    (unsigned char b_BoardHandle,
                           unsigned char b_InterruptMask,
                           unsigned int * ui_AnalogInputValue)
    {
        .
        .
    }

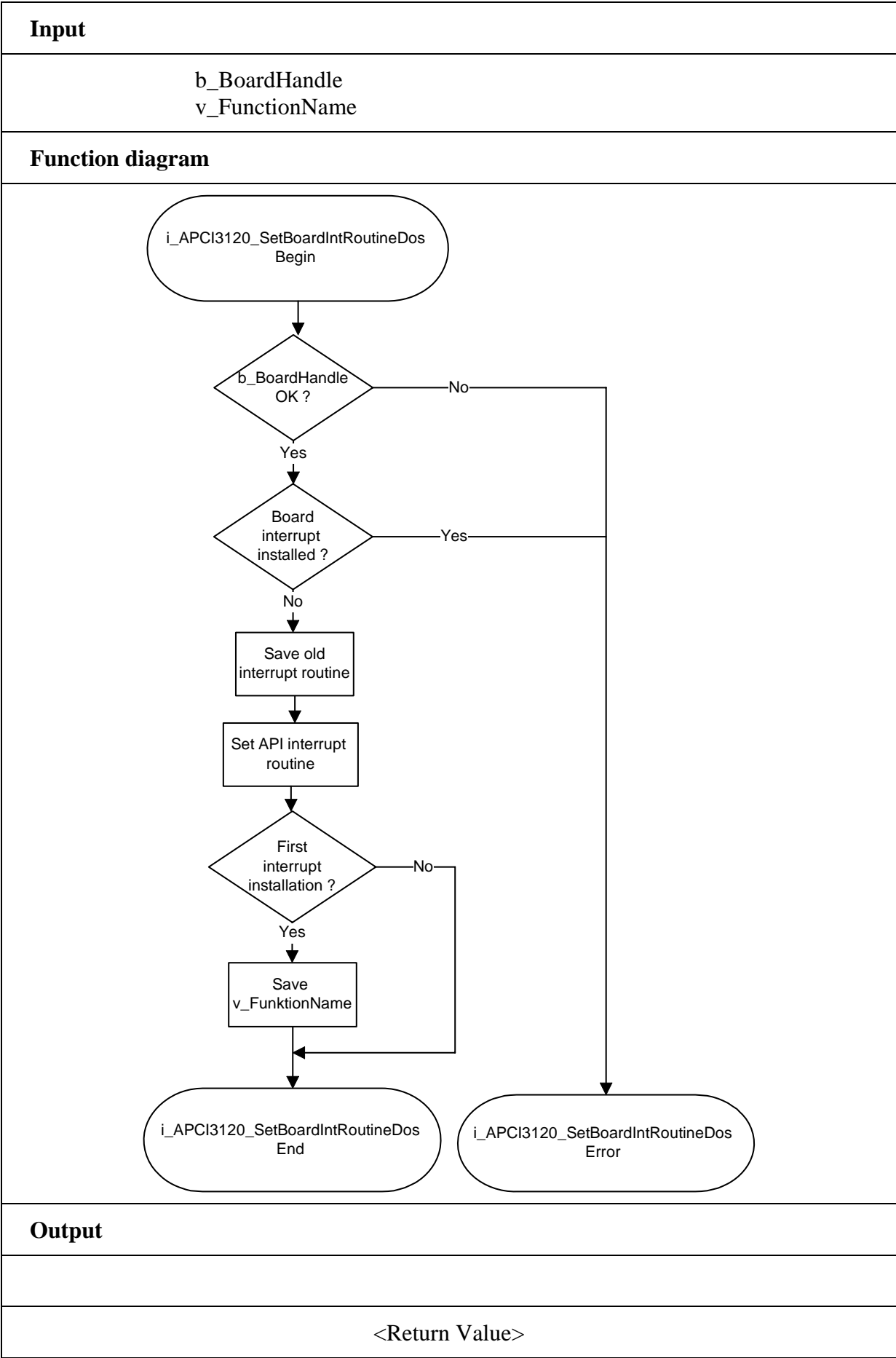
int      i_ReturnValue;
unsigned char  b_BoardHandle;

i_ReturnValue = i_APCI3120_SetBoardIntRoutineDos    (b_BoardHandle,
                                                    v_FunctionName );

```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: Interrupt already installed
- 3: The selected calling mode of the user interrupt routine is wrong
- 4: No more memory available for the user shared memory
- 5: Interrupt routine already used
- 6: No interrupt attributed to the board
- 7: ACPI cannot be activated under Windows NT 4.0 or MS-Dos.
- 8: PNP OS cannot be activated under Windows NT 4.0 or MS-Dos.



2) i_APCI3120_SetBoardIntRoutineVBDos (..)**i****IMPORTANT!**

This function is only available for Visual Basic DOS.

Syntax:

```
<Return value> = i_APCI3120_SetBoardIntRoutineVBDos
                    (BYTE    b_BoardHandle)
```

Parameters:**- Input:**

BYTE b_BoardHandle Handle of the board

- Output:

No output signal has occurred

Task:

This function must be called up for each **xPCI-3120** on which an interrupt action is to be enabled. When an interrupt is enabled, a Visual Basic Event is generated.

First calling (first board):

- the user interrupt routine is installed
- interrupts are enabled.

If you operate several boards **xPCI-3120** which have to react to interrupts, call up the function as often as you operate boards **xPCI-3120**.

Interrupt

The user interrupt routine is called up by the system when an interrupt is generated.

Controlling the interrupt management

Please use the following functions

"ON UEVENT GOSUB xxxxxxxxx" of Visual Basic for DOS

and

"i_APCI3120_TestInterrupt"

This function tests the interrupt of the **xPCI-3120**. It is used to obtain the values of *b_BoardHandle* , *b_InterruptMask* and *pui_AnalogInputValue*.

Calling convention:Visual Basic DOS:

Dim Shared i_ReturnValue	As Integer
Dim Shared i_BoardHandle	As Integer
Dim Shared i_InterruptMask	As Integer
Dim Shared l_AnalogInputValue	As Long

IntLabel:

```
i_ReturnValue = i_APCI3120_TestInterrupt (i_BoardHandle,
                                           i_InterruptMask,
                                           i_AnalogInputValue (0))
```

```
.
```

```
.
```

```
.
```

Return

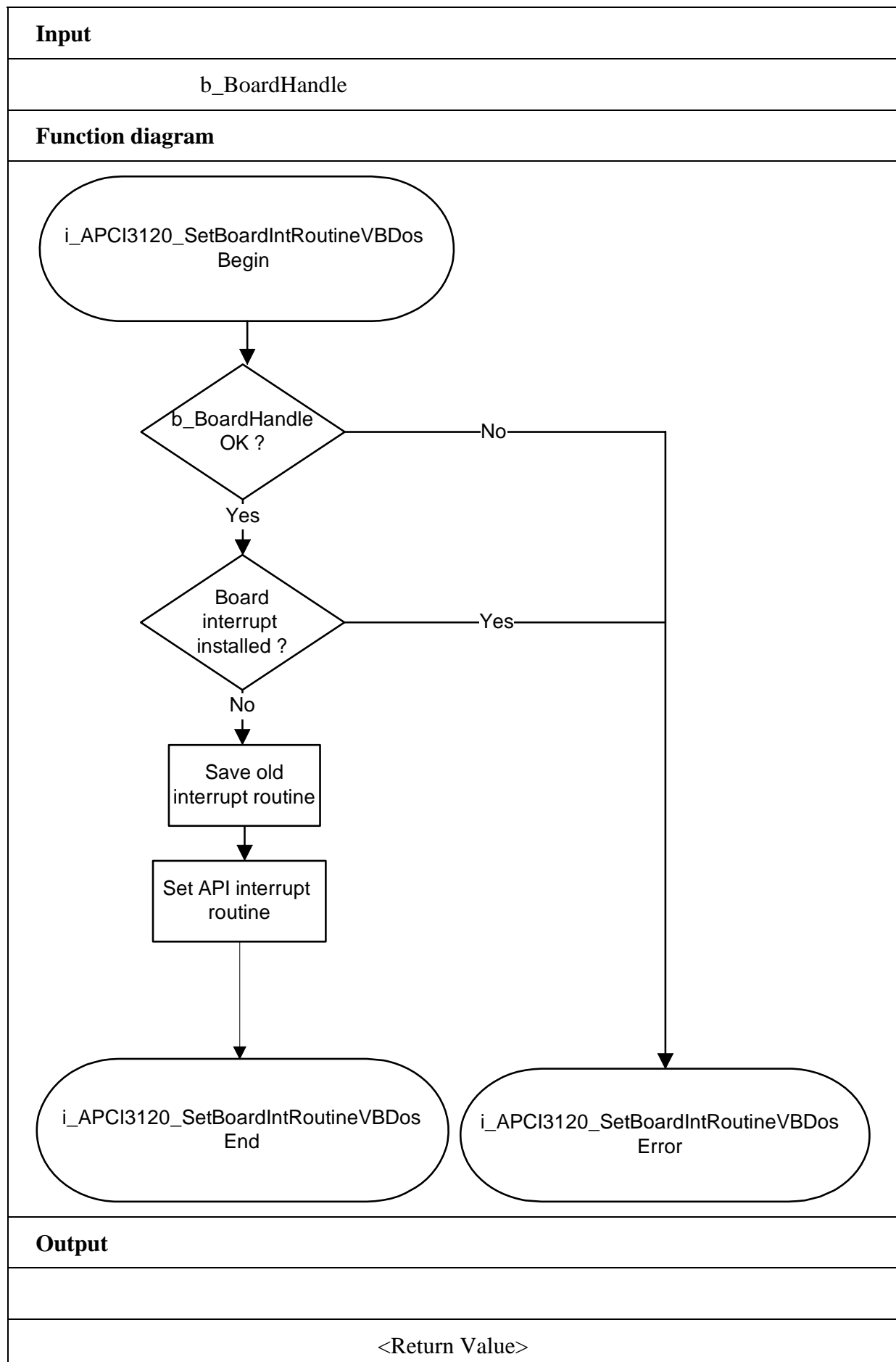
ON UEVENT GOSUB IntLabel

UEVENT ON

```
i_ReturnValue = i_APCI3120_SetBoardIntRoutineVBDos (b_BoardHandle)
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: Interrupt already installed
- 3: The selected calling mode of the user interrupt routine is wrong
- 4: No more memory available for the user shared memory
- 5: Interrupt routine already used
- 6: No interrupt attributed to the board
- 7: ACPI cannot be activated under Windows NT 4.0 or MS-Dos.
- 8: PNP OS cannot be activated under Windows NT 4.0 or MS-Dos.



i**IMPORTANT!**

This function is only available for Windows 3.1 and Windows 3.11

3) i_APCI3120_SetBoardIntRoutineWin16 (.)**Syntax:**

```
<Return value> = i_APCI3120_SetBoardIntRoutineWin16
    (BYTE    b_BoardHandle,
     VOID     v_FunctionName (BYTE    b_BoardHandle,
                               BYTE    b_InterruptMask,
                               PUINT   pui_AnalogInputValue))
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of the board
VOID	v_FunctionName	Name of the user interrupt routine

- Output:

No output signal has occurred

Task:

This function must be called up for each **xPCI-3120** on which an interrupt action is to be enabled.

First calling (first board):

- the user interrupt routine is installed
- interrupts are enabled.

If you operate several boards **xPCI-3120** which have to react to interrupts, call up the function as often as you operate boards **xPCI-3120**. The variable *v_FunctionName* is only relevant **for the first calling**.

From the second call of the function (next board):

- interrupts are enabled.

Interrupt

The user interrupt routine is called up by the system when an interrupt is generated.

If several boards are operated and if they have to react to interrupts, the variable *b_BoardHandle* returns the identification number (handle) of the board which has generated the interrupt.

The user interrupt routine must have the following syntax:

```
VOID  v_FunctionName    (BYTE   b_BoardHandle,
                        BYTE   b_InterruptMask,
                        PUINT  b_AnalogInputValue)
```

<i>v_FunctionName</i>	Name of the user interrupt routine
<i>b_BoardHandle</i>	Handle of the xPCI-3120 which has generated the interrupt
<i>b_InterruptMask</i>	Mask of the events which have generated the interrupt.
<i>pui_AnalogInputValue</i>	The latched values of the timer and of the DMA buffer are returned.

i

IMPORTANT!

If you use Visual Basic for Windows the following parameter has no meaning. You must use the „i_APCI3120_TestInterrupt“ function.

```
VOID  v_FunctionName (BYTE   b_BoardHandle,
                        BYTE   b_InterruptMask,
                        PUINT  pui_AnalogInputValue)
```

Calling convention:

ANSI C:

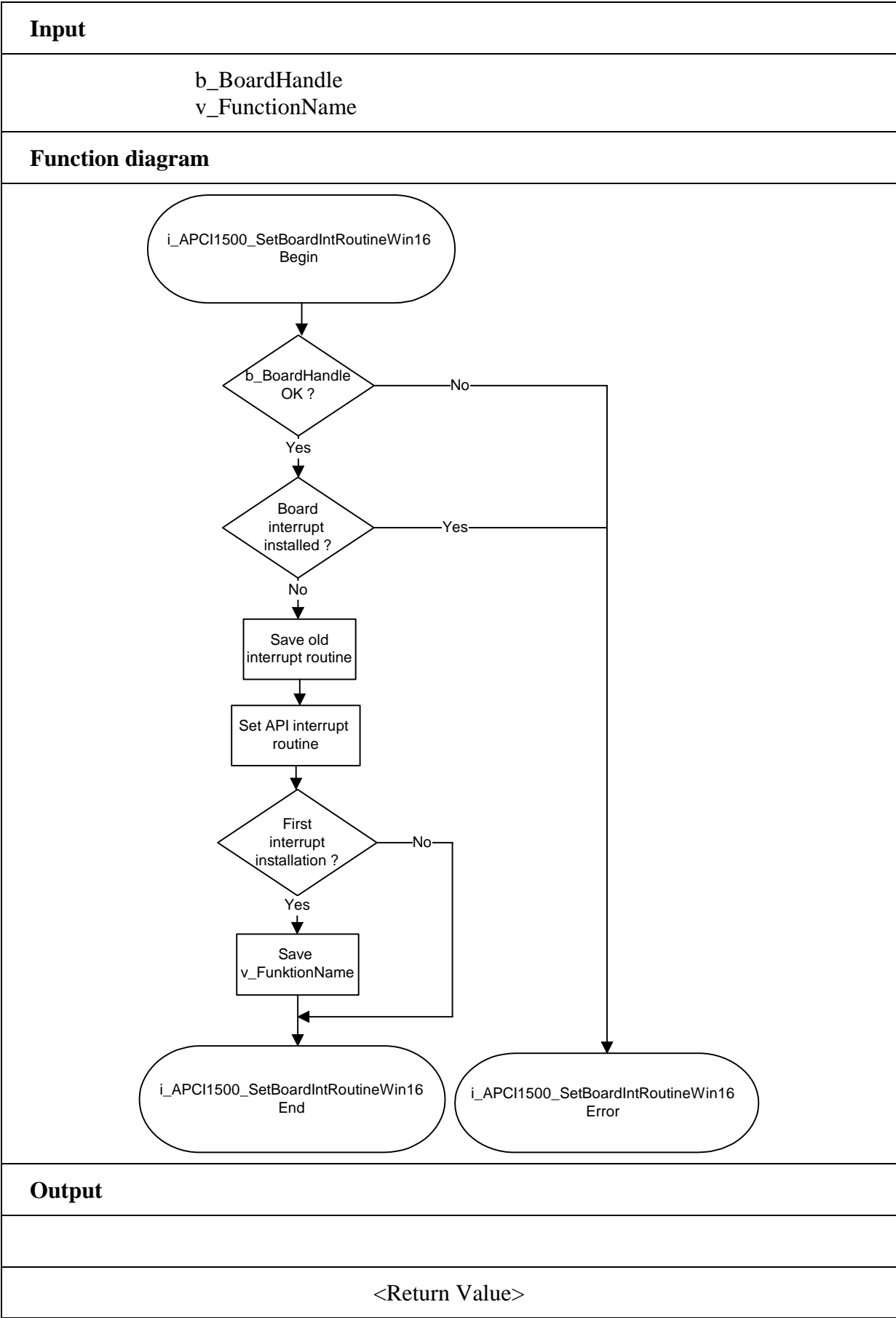
```
void    v_FunctionName    (unsigned char b_BoardHandle,
                          unsigned char b_InterruptMask,
                          unsigned int * ui_AnalogInputValue)
{
    .
    .
}
```

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

```
i_ReturnValue = i_APCI3120_SetBoardIntRoutineWin16
                (b_BoardHandle,
                 v_FunctionName );
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: Interrupt already installed
- 3: The selected calling mode of the user interrupt routine is wrong
- 4: No more memory available for the user shared memory
- 5: Interrupt routine already used
- 6: No interrupt attributed to the board
- 7: ACPI cannot be activated under Windows NT 4.0 or MS-Dos.
- 8: PNP OS cannot be activated under Windows NT 4.0 or MS-Dos.



i**IMPORTANT!**

This function is only available for Windows NT and Windows 95.

4) i_APCI3120_SetBoardIntRoutineWin32 (..)**Syntax:**

```
<Return value> = i_APCI3120_SetBoardIntRoutineWin32
    (BYTE      b_BoardHandle,
     BYTE      b_UserCallingMode,
     ULONG     ul_UserSharedMemorySize,
     VOID **    ppv_UserSharedMemory,
     VOID      v_FunctionName (BYTE  b_BoardHandle,
                                BYTE  b_InterruptMask,
                                PUINT b_AnalogInputValue,
                                BYTE  b_UserCallingMode
                                VOID * pv_UserSharedMemory))
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of the board
BYTE	b_UserCallingMode	APCI3120_SYNCHRONOUS_MODE: The user routine is directly called by the driver interrupt routine. APCI3120_ASYNCHRONOUS_MODE: The user routine is called by the driver interrupt thread.
VOID	v_FunctionName	Name of the user interrupt routine
ULONG	ul_UserSharedMemorySize	With Visual Basic 5.0: determines the number of allocated values (only used for acquisitions with DMA) Other compilers: Determines the size in bytes of the user shared memory. Only used if you have selected APCI3120_SYNCHRONOUS_MODE

i**IMPORTANT!**

The size of the User Shared Memory is limited on 63 MB. It could cause problems if more memory is needed.

- Output:

VOID **	ppv_UserSharedMemory	With Visual Basic 5.0: pointer on a dynamic array .(Only used for acquisition with DMA See function i_APCI3120_InitAnalogInputAcquisition) Other compilers: User shared memory address Only used if you have selected APCI3120_SYNCHRONOUS_MODE
---------	----------------------	---

Task:

If you use Visual Basic 5.0:

- only the asynchronous mode is available.
- the parameters *ppv_UserSharedMemory* and *ul_UserSharedMemorySize* are used for acquisition with DMA. When you want to start an analog acquisition with DMA you should pass a pointer and its size on a dynamic array. This pointer is used in order to store the analog value. The size corresponds to the number of acquisitions you want to make before an interrupt is generated. If you do not use the DMA mode, enter 0 for both parameters.

Important: The allocated size for the dynamic array must at least equal the number of acquisitions.

i**Windows 32-bit information**

For Windows NT and Windows 95, 4 rings (ring 0 to ring 3) are available.

- The user application operates in ring 3. This ring does not give access to hardware.
- VXD and SYS driver operate in ring 0 and give access to hardware.
- Ring 0 has no direct access to global variable from ring 3. It has to use a shared memory.
- Ring 0 and ring 3 have a pointer that points on this shared memory. The 2 pointers are not configured under the same address.

This function must be called up for each **xPCI-3120** for which an interrupt is to be enabled. It installs one user interrupt function in all boards on which an interrupt is to be enabled.

First calling (first board):

- the user interrupt routine is installed
- interrupts are enabled
- user shared memory is allocated if APCI3120_SYNCHROUNOUS_MODE has been selected.

If you operate several boards **xPCI-3120** which have to react to interrupts, call up the function as often as you operate boards **xPCI-3120**. The variable *v_FunctionName* is only relevant **for the first calling**.

From the second call of the function (next board):

- interrupts are enabled.

Interrupt

The user interrupt routine is called up by the system when an interrupt is generated.

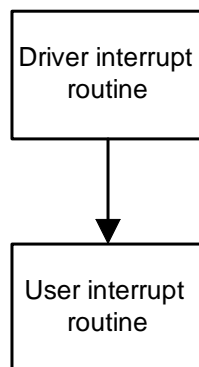
If several boards are operated and if they have to react to interrupts, the variable *b_BoardHandle* returns the identification number (handle) of the board which has generated the interrupt.

The user interrupt routine can be called:

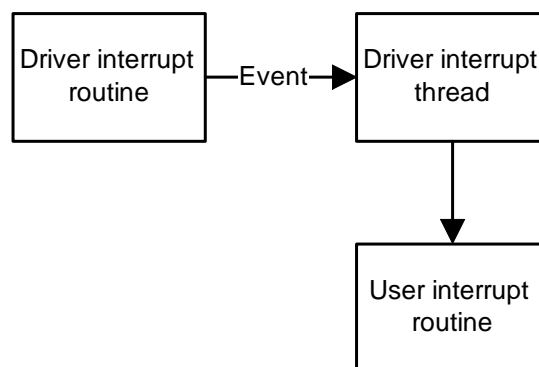
- directly by the driver interrupt routine (Synchronous mode). The code of the user interrupt routine directly operates in ring 0.
- by the driver interrupt thread (Asynchronous mode). An event is generated and the interrupt thread calls up the user interrupt routine. The code of the user interrupt routine operates in ring 3.

The driver interrupt thread have the highest priority (31) in the system.

Synchronous mode



Asynchronous mode



SYNCHRONOUS MODE	
ADVANTAGE	The code of the user interrupt routine is directly called by driver interrupt routine (ring 0). The time between interrupt and the user interrupt routine is reduced.
RESTRICTION	The user cannot debug the user interrupt routine.
	The user routine cannot call Windows API functions.
	The user routine cannot call functions which give access to global variables. The user can still use a shared memory.
	This mode is not available for Visual Basic

	ASYNCHRONOUS MODE
ADVANTAGE	The user can debug the user interrupt routine provided he has not programmed in Visual Basic 5
	The user routine can call Windows API functions.
	The user routine can call functions which give access to global variables.
	The user routine can call all xPCI-3120 driver functions with the following extension: "i_APCI3120_XXXX"
RESTRICTION	The code of the user interrupt routine is called by driver interrupt thread routine (ring 3). The time between interrupt and the user interrupt routine is increased.

Shared memory

If you have selected the APCI3120_SYNCHRONOUS_MODE you cannot have access to global variables. But you have the possibility to create a shared memory (*ppv_UserSharedMemory*). The user shared memory can have all predefined compiler types or user define types.

The variable *ul_UserSharedMemorySize* indicates the size in bytes of the selected user type. A pointer of the variable *ppv_UserSharedMemory* is given to the user interrupt routine with the variable *pv_UserSharedMemory*. This is not possible for Visual Basic.

The user interrupt routine must have the following syntax:

```
VOID   v_FunctionName (BYTE       b_BoardHandle,
                        BYTE       b_InterruptMask,
                        PUINT      b_AnalogInputValue,
                        BYTE       b_UserCallingMode,
                        VOID *      pv_UserSharedMemory)
```

<i>v_FunctionName</i>	Name of the user interrupt routine
<i>b_BoardHandle</i>	Handle of the xPCI-3120 which has generated the interrupt
<i>b_InterruptMask</i>	Mask of the events which have generated the interrupt.
<i>pui_AnalogInputValue</i>	The latched values of the counter are returned.
<i>b_UserCallingMode</i>	APCI3120_SYNCHRONOUS_MODE: The user routine is directly called by driver interrupt routine. APCI3120_ASYNCHRONOUS_MODE: The user routine is called by driver interrupt thread
<i>pv_UserSharedMemory</i>	Pointer of the user shared memory.

i**IMPORTANT!**

If you use Visual Basic 4 the following parameters have no meaning. You must use the „i_APCI3120_TestInterrupt“ function.

```

BYTE    b_UserCallingMode,
ULONG   ul_UserSharedMemorySize,
VOID ** ppv_UserSharedMemory,
VOID     v_FunctionName      (BYTE    b_BoardHandle,
                              BYTE    b_InterruptMask,
                              PUINT   pui_AnalogInputValue,
                              BYTE    b_UserCallingMode,
                              VOID *  pv_UserSharedMemory)

```

Calling convention:

ANSI C:

```

typedef struct
{
    .
    .
    .
}str_UserStruct;

str_UserStruct * ps_UserSharedMemory;

void    v_FunctionName    (unsigned char  b_BoardHandle,
                          unsigned char  b_InterruptMask,
                          unsigned int   *ui_AnalogInputValue,
                          unsigned char  b_UserCallingMode,
                          void *         pv_UserSharedMemory)
{
    str_UserStruct * ps_InterruptSharedMemory;

    ps_InterruptSharedMemory = (str_UserStruct *) pv_UserSharedMemory;
    .
    .
}

int      i_ReturnValue;
unsigned char  b_BoardHandle;

i_ReturnValue = i_APCI3120_SetBoardIntRoutineWin32
                (b_BoardHandle,
                 APCI3120_SYNCHRONOUS_MODE,
                 sizeof (str_UserStruct),
                 (void **) &ps_UserSharedMemory,
                 v_FunctionName);

```

i**IMPORTANT!****Do observe the following when using Visual Basic 5.0.**Visual Basic 5: Acquisition with DMA

```

Sub      v_FunctionName      (ByVal i_BoardHandle      As Integer,
                              ByVal i_InterruptMask     As Integer,
                              l_AnalogInputValue        As Long,
                              b_UserCallingMode         As Integer,
                              l_UserSharedMemory        As Long)

End Sub

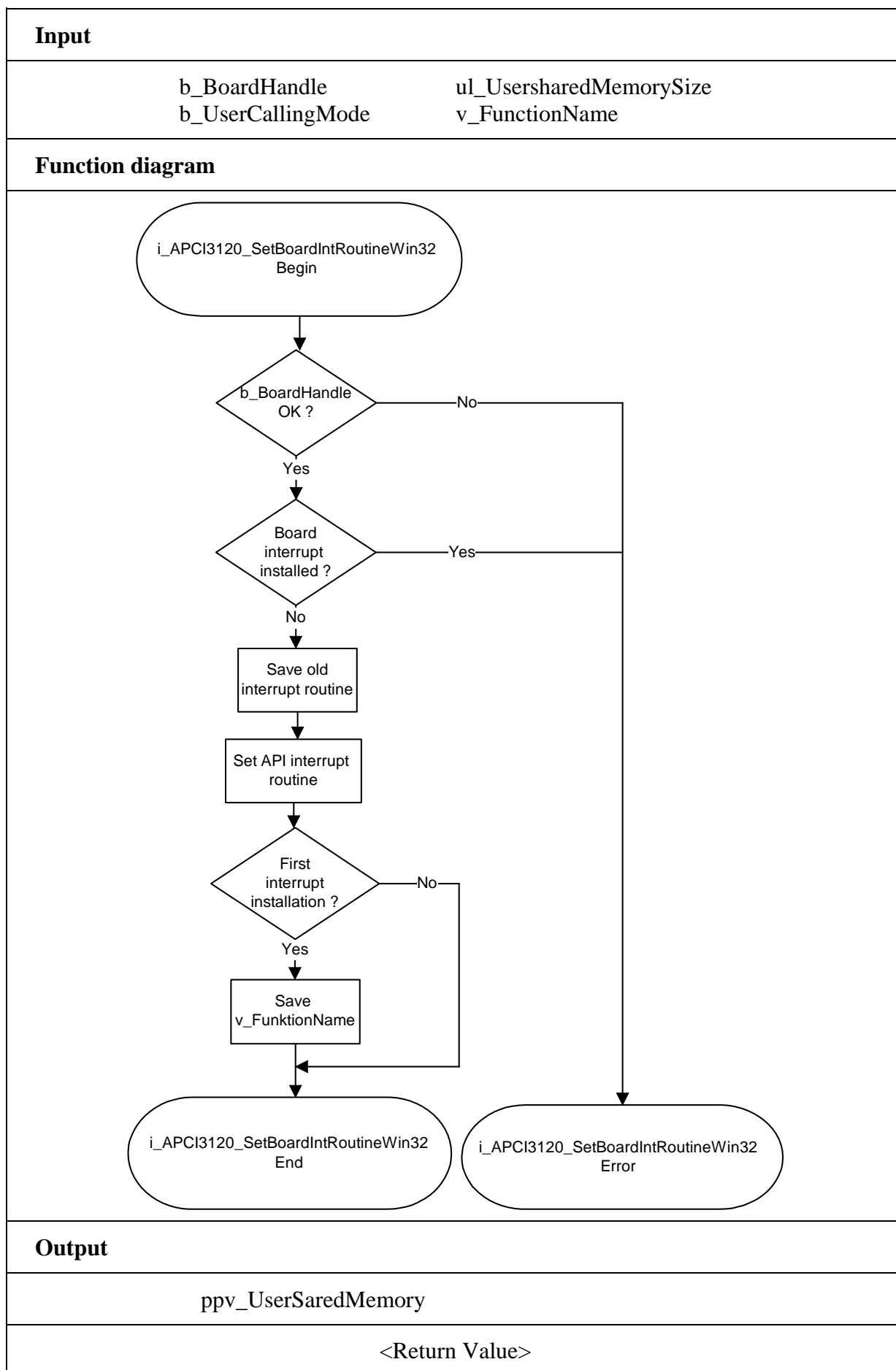
Dim i_ReturnValue As Integer
Dim i_BoardHandle As Integer
Dim DynamicArray() As Long
ReDim DynamicArray(NUMBER_OF_VALUE) As Long
...
i_ReturnValue = i_APCI3120_SetBoardIntRoutineWin32
                              (i_BoardHandle,
                               APCI3120_ASYNCHRONOUS_MODE,
                               NUMBER_OF_VALUE,
                               DynamicArray(0),
                               AddressOf v_FunctionName)

...

```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: Interrupt already installed
- 3: The selected calling mode of the user interrupt routine is wrong
- 4: No more memory available for the user shared memory
- 5: Interrupt routine already used
- 6: No interrupt attributed to the board
- 7: ACPI cannot be activated under Windows NT 4.0 or MS-Dos.
- 8: PNP OS cannot be activated under Windows NT 4.0 or MS-Dos.



5) i_APCI3120_TestInterrupt (..)**i****IMPORTANT!**

This function is only available in Visual Basic for DOS and Windows .

Syntax:

```
<Return value> = i_APCI3120_TestInterrupt
                        (PBYTE      pb_BoardHandle,
                        PBYTE      pb_InterruptMask,
                        PUINT      pui_AnalogInputValue)
```

Parameters:**- Input:**

No input signal occurs

- Output:

PBYTE	pb_BoardHandle	Handle of the board xPCI-3120 which has generated the interrupt
PBYTE	pb_InterruptMask	Mask of the events which have generated the interrupt
PBYTE	pui_AnalogInputValue	The values of the analog input channels and of the DMA buffer are returned.

Values returned for the analog input channels

Event	Variable name	Variable content
<i>b_InterruptMask</i> = 1	<i>pui_AnalogInputValue</i> [0]	Number of the last analog input channel
	<i>pui_AnalogInputValue</i> [1]	Value of the analog input channel
<i>b_InterruptMask</i> = 2	<i>pui_AnalogInputValue</i> [0]	Number of analog input channels
	<i>pui_AnalogInputValue</i> [1] up to <i>pui_AnalogInputValue</i> [<i>pui_AnalogInputValue</i> [0]] See example 1	Values of the analog input channels
<i>b_InterruptMask</i> = 4	<i>pui_AnalogInputValue</i> [0]	Number of analog input channels
	<i>pui_AnalogInputValue</i> [1] up to <i>pui_AnalogInputValue</i> [<i>pui_AnalogInputValue</i> [0]] See example 1	Values of the analog input channels
<i>b_InterruptMask</i> = 8	<i>pui_AnalogInputValue</i>	Values of the DMA buffer

Example 1 *pui_AnalogInputValue* [0] = 3

3	= Number of analog input channels
1	= Analog value 0
2	= Analog value 1
3	= Analog value 2

Interrupt mask

Mask	Meaning
0000 0001	End of Conversion (EOC)
0000 0010	Conversion driven by timer is completed
0000 0100	Conversion of a group of channels is completed (EOS)
0000 1000	DMA conversion cycle is completed
0001 0000	Timer 2 has run down
0010 0000	Analog output channels - Watchdog has run down

Task:

Verifies if a board **xPCI-3120** has generated an interrupt. If yes, the function returns the board handle and the interrupt source.

Calling convention:

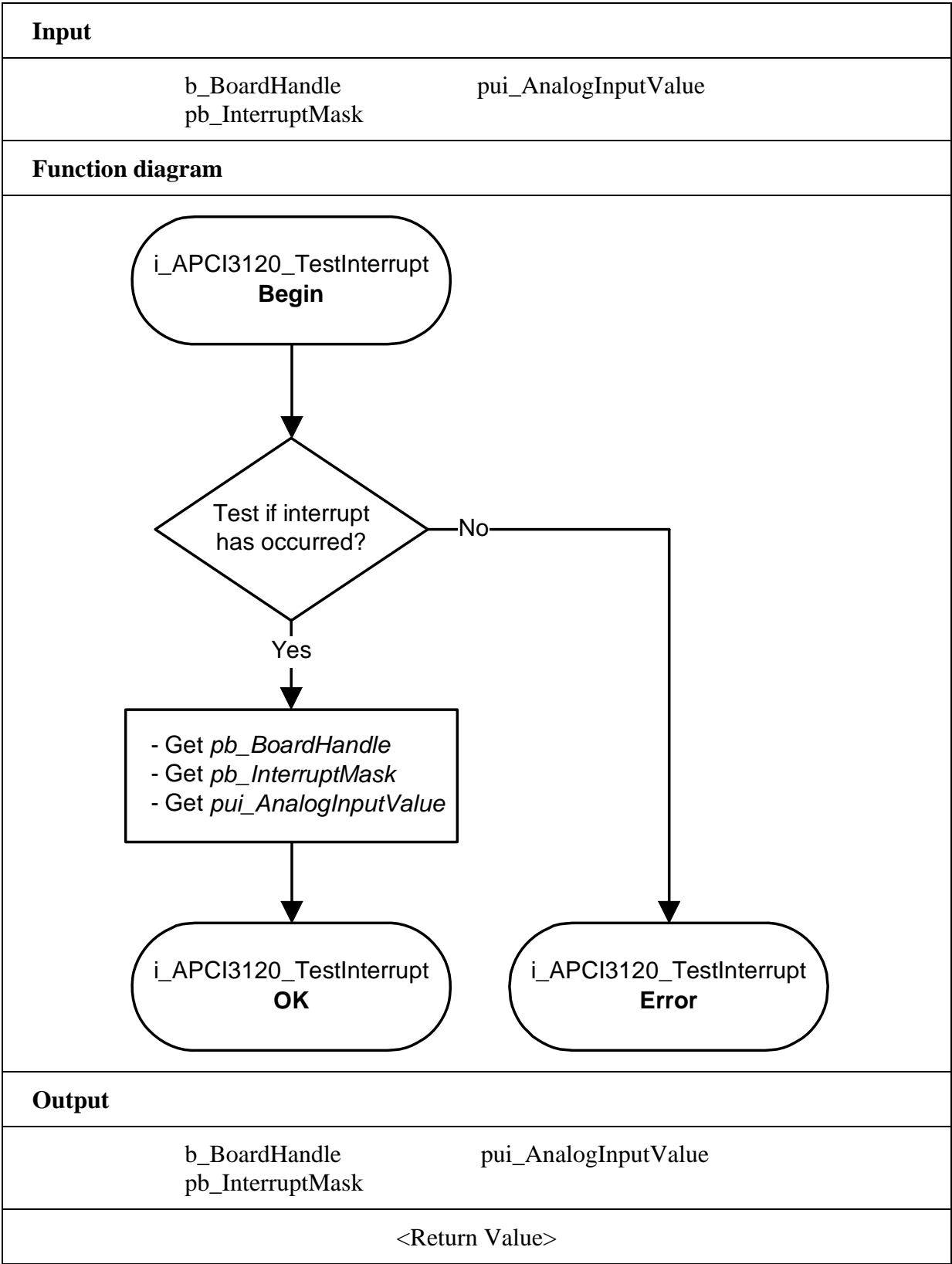
ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned char b_InterruptMask;
unsigned int  ui_AnalogInputValue [XX];

i_ReturnValue = i_APCI3120_TestInterrupt (&b_BoardHandle,
                                           &b_InterruptMask,
                                           ui_AnalogInputValue);
```

Return value:

-1 No interrupt
> 0 IRQ number



6) i_APCI3120_ResetBoardIntRoutine (..)**Syntax:**

<Return value> = i_APCI3120_ResetBoardIntRoutine (BYTE b_BoardHandle)

Parameters:**- Input:**

BYTE b_BoardHandle Handle of the board

- Output:

No output signal has occurred.

Task:

Stops the interrupt management of board **xPCI-3120**.

Deinstalls the user interrupt routine if the management of interrupts of all boards **xPCI-3120** is stopped.

Calling convention:ANSI C:

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

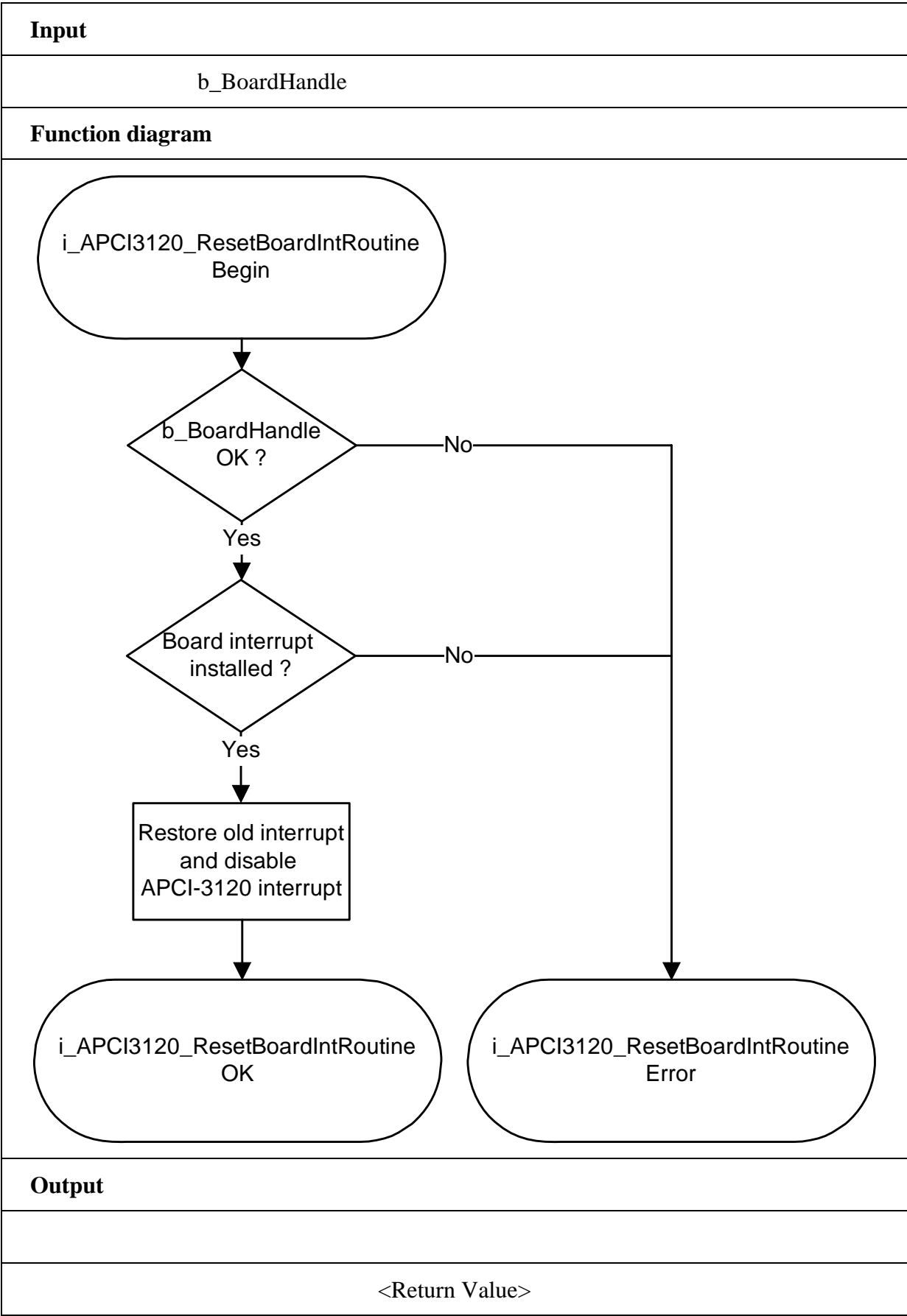
```
i_ReturnValue = i_APCI3120_ResetBoardIntRoutine (b_BoardHandle);
```

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: No interrupt function has been initialised with function
"i_APCI3120_SetBoardIntRoutineXXX"



3.3 Direct conversion of analog input channels

1) i_APCI3120_Read1AnalogInput (...)

Syntax:

```
<Return value> = i_APCI3120_Read1AnalogInput
                    (BYTE      b_BoardHandle,
                     BYTE      b_Channel,
                     BYTE      b_Gain,
                     BYTE      b_Polarity,
                     UINT       ui_ConvertTiming
                     BYTE      b_InterruptFlag,
                     PUINT      pui_AnalogInputValue)
```

Parameters:

-Output

BYTE	b_BoardHandle	Handle of the board
BYTE	b_Channel	Number of the analog input channel to be read (0 bis 15). See table 3-3
BYTE	b_Gain	Gain selection. See table 3-4 Gain 1: 0 Gain 2: 16 Gain 5: 32 Gain 10: 64
BYTE	b_Polarity	Selection of the input voltage range of the analog input channel to convert. See table 3-5. Unipolar: 128 Bipolar: 0
UINT	ui_ConvertTiming	Selection of the conversion time From 10 µs up to 32767 µs.
BYTE	b_InterruptFlag	APCI3120_ENABLE: An interrupt is generated at EOC. See function "i_APCI3120_SetBoardIntRoutineXX". APCI3120_DISABLE: No interrupt is generated at EOC. The analog value is located in parameter <i>pui_AnalogInputValue</i> .
- Output:		
PUINT	pui_AnalogInputValue	The analog value is returned (0 to 65535)

Task:

Reads the current value of the analog input channel *b_Channel* with a gain of *b_Gain*, in the input voltage range of *b_Polarity* and a conversion time of *ui_ConvertTiming*

Calling convention:ANSI C :

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

```
i_ReturnValue = i_APCI3120_Read1AnalogInput (b_BoardHandle,  
                                              APCI3120_CHANNEL_1,  
                                              APCI3120_1_GAIN,  
                                              APCI3120_UNIPOLAR,  
                                              10,  
                                              APCI3120_DISABLE,  
                                              & ui_AnalogInputValue);
```

Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: The number of the analog input channel is wrong. See table 3-3

-3: The selected gain is wrong. See table 3-4.

-4: The selected input voltage range is wrong. See table 3-5

-5: The conversion time selected is wrong

-6: Wrong parameter entered for *b_InterruptFlag* or the user interrupt routine has not been installed. See function "i_APCI3120_SetBoardIntRoutineXXX".

Table 3-3: Selection of the analog input channels

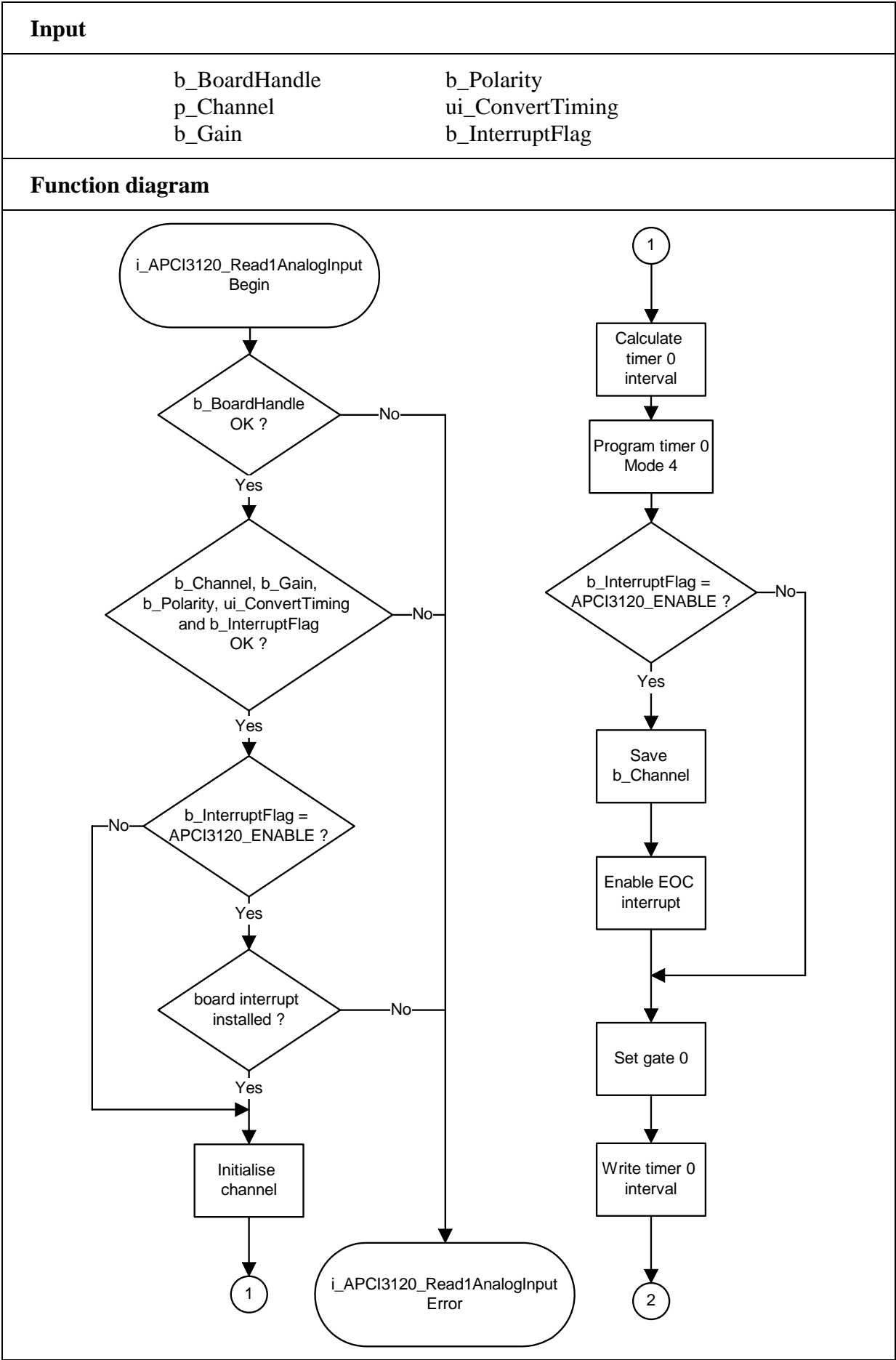
pb_ChannelArray Parameter	Analog input channel
APCI3120_CHANNEL_0	0
APCI3120_CHANNEL_1	1
APCI3120_CHANNEL_2	2
APCI3120_CHANNEL_3	3
APCI3120_CHANNEL_4	4
APCI3120_CHANNEL_5	5
APCI3120_CHANNEL_6	6
APCI3120_CHANNEL_7	7
APCI3120_CHANNEL_8	8
APCI3120_CHANNEL_9	9
APCI3120_CHANNEL_10	10
APCI3120_CHANNEL_11	11
APCI3120_CHANNEL_12	12
APCI3120_CHANNEL_13	13
APCI3120_CHANNEL_14	14
APCI3120_CHANNEL_15	15

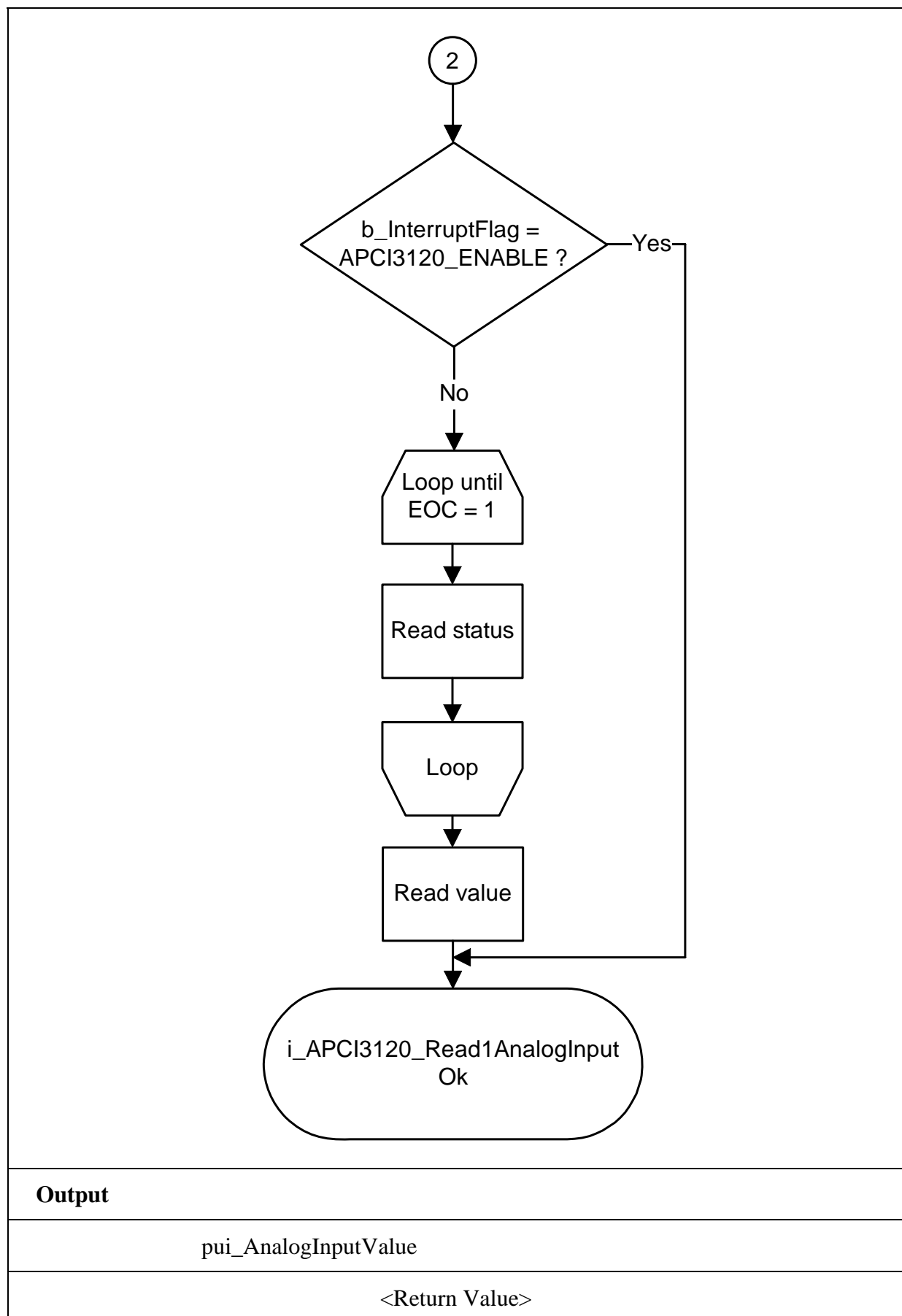
Table 3-4: Gain selection

pb_GainArray Parameter	Gain
APCI3120_1_GAIN	1
APCI3120_2_GAIN	2
APCI3120_5_GAIN	5
APCI3120_10_GAIN	10

Table 3-5: Selection of the input voltage range

pb_PolarityArray Parameter	Voltage range
APCI3120_UNIPOLAR	0-10V with gain 1
APCI3120_BIPOLAR	±10V with gain 1





2) i_APCI3120_ReadMoreAnalogInput (...)**Syntax:**

```

<Return value> = i_APCI3120_ReadMoreAnalogInput
                    (BYTE    b_BoardHandle,
                     BYTE    b_SequenzArraySize,
                     PBYTE   pb_ChannelArray,
                     PBYTE   pb_GainArray,
                     PBYTE   pb_PolarityArray,
                     UINT     ui_ConvertTiming
                     BYTE    b_InterruptFlag,
                     PUINT    pui_AnalogInputValueArray)

```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of the board
BYTE	b_SequenzArraySize	Size of the scan lists (1 up to 16 elements)
PBYTE	pb_ChannelArray	Scan list for the analog input channels. See table 3-3.
PBYTE	pb_GainArray	Scan list for gain See table 3-4.
PBYTE	pb_PolarityArray	Scan list for the input voltage range See table 3-5.
UINT	ui_ConvertTiming	Selection of the conversion time From 10 µs up to 32767 µs.
BYTE	b_InterruptFlag	APCI3120_ENABLE: An interrupt is generated when the last conversion of the channel group is completed (EOS). See function "i_APCI3120_SetBoardIntRoutine". APCI3120_DISABLE: No interrupt is generated at the end of conversion. The analog values are located in parameter <i>pui_AnalogInputValueArray</i> .

- Output:

PUINT pui_AnalogInputValueArray Output values are returned

Task:

Reads several analog input channels.

The priority of the analog input channels is set with the scan list.

The scan list allows to determine the input voltage range and the gain for each analog input channel. The gain is defined with parameter *pb_GainArray* for each analog input channel.

The input voltage range is defined with parameter *pb_PolarityArray* for each analog input channel.

Calling convention:ANSI C :

```

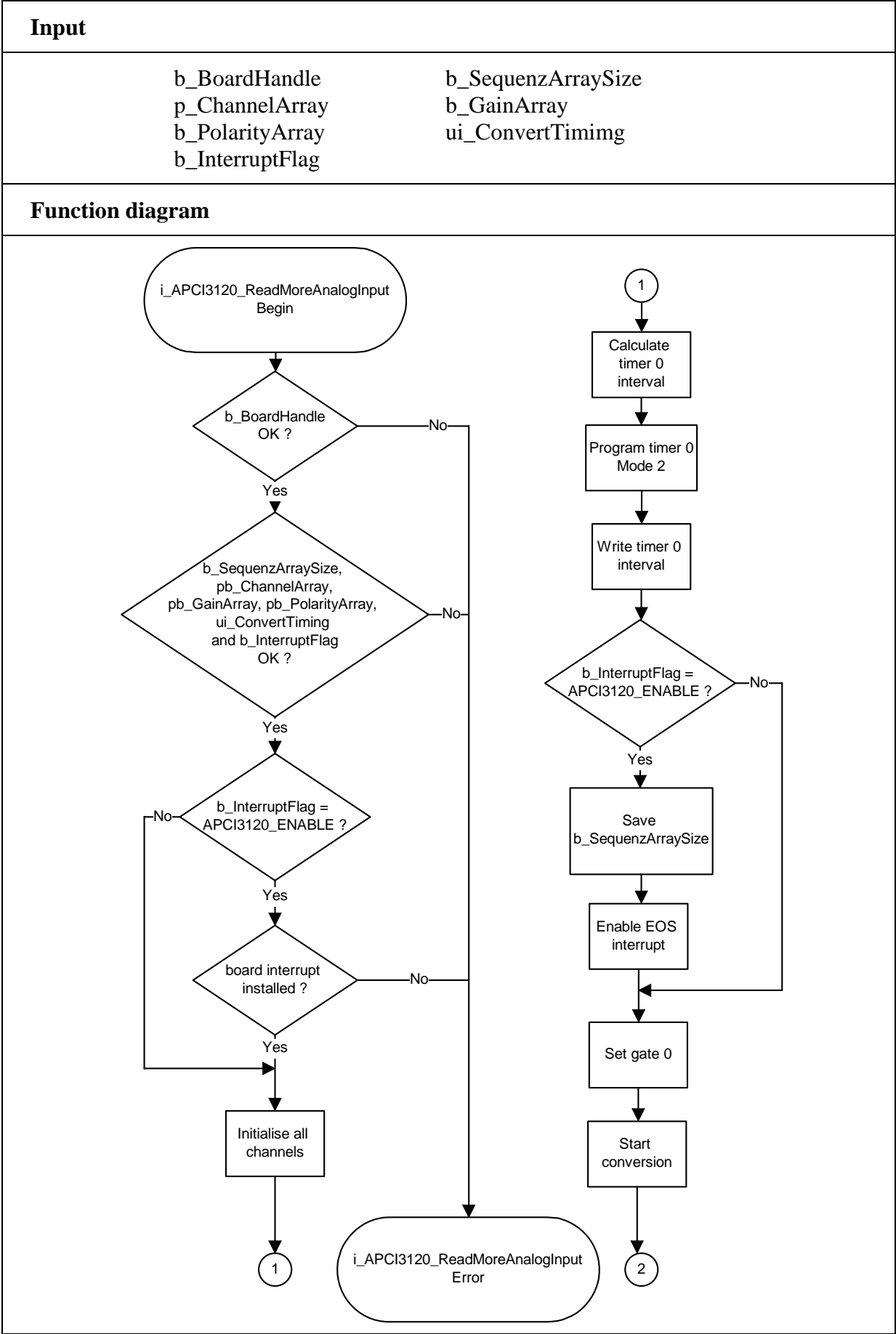
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned int  ui_AnalogInputValue;
unsigned char b_ChannelArray    [16];
unsigned char b_GainArray       [16];
unsigned char b_PolarityArray   [16];
unsigned int  ui_AnalogInputValueArray [16];

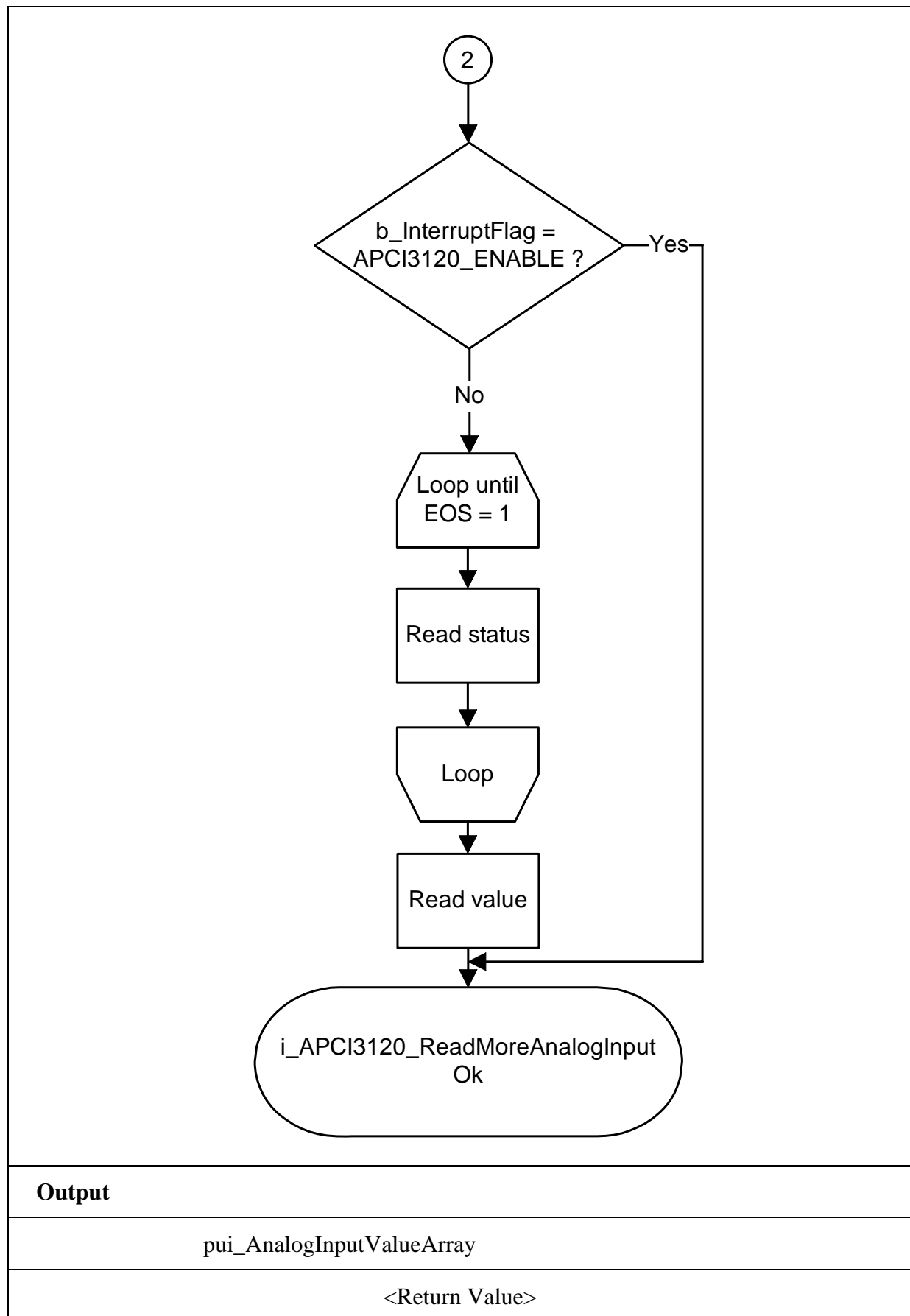
i_ReturnValue = i_APCI3120_ReadMoreAnalogInput
                                                         (b_BoardHandle,
                                                         16,
                                                         b_ChannelArray,
                                                         b_GainArray,
                                                         b_PolarityArray,
                                                         APCI3120_DISABLE,
                                                         ui_AnalogInputValue);

```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The size of the scan list is wrong
- 3: Wrong parameter detected in table "pb_ChannelArray"
- 4: Wrong parameter detected in table "pb_GainArray"
- 5: Wrong parameter detected in table "pb_PolarityArray"
- 6: Selected conversion time is wrong
- 7: Wrong parameter entered for *b_InterruptFlag*, or the user interrupt routine has not been installed.
See function "i_APCI3120_SetBoardIntRoutine".





3.4 Cyclic conversion of analog input channels

i

IMPORTANT!

If you use Visual Basic 5 and want to make an acquisition with DMA, make sure that the function `i_APCI3120_SetBoardIntRoutineWin32` is initialised .

1) `i_APCI3120_InitAnalogInputAcquisition (...)`

Syntax:

```
<Return value> = i_APCI3120_InitAnalogInputAcquisition
                    (BYTE      b_BoardHandle,
                     BYTE      b_SequenzArraySize,
                     PBYTE     pb_ChannelArray,
                     PBYTE     pb_GainArray,
                     PBYTE     pb_PolarityArray,
                     BYTE      b_AcquisitionMode,
                     BYTE      b_ExternTrigger,
                     UINT       ui_AcquisitionTiming,
                     LONG       l_DelayTiming,
                     ULONG      ul_NumberOfAcquisition,
                     BYTE      b_DMAUsed,
                     BYTE      b_AcquisitionCycle)
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of the board
BYTE	b_SequenzArraySize	Size of the scan lists (1 up to 16 elements)
PBYTE	pb_ChannelArray	Scan list for the analog input channels. See table 3-3.
PBYTE	pb_GainArray	Scan list for gain. See table 3-4.
PBYTE	pb_PolarityArray	Scan list for the input voltage range See table 3-5.
BYTE	b_AcquisitionMode	Two conversion cycles are possible: - APCI3120_SIMPLE_MODUS: A conversion occurs every <i>ui_AcquisitionTiming</i> (time interval) See example 2. - APCI3120_DELAY_MODUS: Both times are used in this mode <i>ui_AcquisitionTiming</i> and <i>l_DelayTiming</i> . Conversions occur every <i>ui_AcquisitionTiming</i> (time interval) until all analog input channels have been acquired (determined by <i>b_SequenzArraySize</i>) Example 3 step 1. Afterwards there is a waiting time of <i>l_DelayTiming</i> . Step 2 of example 3 The two steps are repeated . See example 3. - APCI3120_DELAY_1_MODUS: See example 4

BYTE	b_ExternTrigger	<p>Two modes are possible:</p> <ul style="list-style-type: none"> - APCI3120_DISABLE: the cyclic conversion is initialised after calling the function „i_APCI3120_StartAnalogInputAcquisition“. - APCI3120_ENABLE: the cyclic conversion is initialised after calling the function „i_APCI3120_StartAnalogInputAcquisition“. <p>A logic „1“ on the digital input 1 starts the conversion. The conversion is stopped by calling „i_APCI3120_StopAnalogInputAcquisition“.</p>
UINT	ui_AcquisitionTiming	Time interval in μs between 2 conversions of successive input channels. (from 10 μs to 32767 μs)
LONG	l_DelayTiming	Waiting time in μs between two conversion cycles (from 100 μs to 3276750 μs) This parameter is only relevant if you use the mode APCI3120_DELAY_MODUS or APCI3120_DELAY_1_MODUS.
ULONG	ul_NumberOfAcquisition	<p>If you use DMA, this parameter determines how many conversions have to occur</p> <p>Under Dos: 1 to 32767 UnterWIN32: 1 to 2^{32}</p> <p>If you do not use DMA, this parameter determines how many acquisition cycles must be performed.</p> <p>WARNING: If you work in DMA CONTINUOUS with an acquisition time of 10 μs and under VB 5.0, the minimum value is 16.</p>
BYTE	b_DMAUsed	<p>Determines if DMA should be used or not.</p> <ul style="list-style-type: none"> - APCI3120_DMA_USED: All the conversion values are saved in the DMA buffer. An interrupt is generated, when <i>ui_NumberOfAcquisition</i> conversions have been completed. See function "i_APCI3120_SetBoardIntRoutine". - APCI3120_DMA_NOT_USED: An interrupt is generated when the conversion of the last channel group has been completed. You obtain the analog value through the user interrupt routine. See function "i_APCI3120_SetBoardIntRoutine".

BYTE *b_AcquisitionCycle* Defines the type of DMA conversion.

- APCI3120_CONTINUOUS:
An interrupt is generated each time, when a DMA conversion cycle is completed. A new DMA conversion cycle is started.
- APCI3120_SINGLE:
The DMA conversion cycle is carried out only once: i.e. you receive one single interrupt at the end of the first DMA conversion cycle. See example 1.

- Output:

No output signal has occurred

Task:

This function initialises a cyclic conversion.

The priority of the analog input channels is set through the scan list.

The scan list allows to set the input voltage range and the gain for each analog input channel. See example 1.

The DMA option (APCI3120_DMA_USED) allows to acquire in the background analog values at high frequencies.

An interrupt is generated at the end of conversion. A '2' is passed through the parameter *b_InterruptMask* in your interrupt routine. The DMA buffer is returned through the parameter *pui_AnalogInputValue*.

See function "i_APCI3120_SetBoardIntRoutineXXX".

Do the following:

- set the priority of the analog input channels through the scan list.
- enter the mode through the parameter *b_AcquisitionMode*.
- enter the time between 2 conversions through the parameter *ui_AcquisitionTiming*.
- enter the waiting time between 2 conversion cycles through the parameter *l_DelayTiming*, if you work in mode APCI3120_DELAY_MODUS.
- enter the delay time between the first conversion of the first cycle and the first conversion of the next cycle through the parameter *l_DelayTiming*, if you work in mode APCI3120_DELAY_1_MODUS.
- determine if you want to use DMA (parameter *b_DMAUsed*)
- enter the number of acquisitions through parameter *ul_NumberOfAcquisition*
- enter the DMA conversion cycle through parameter *b_AcquisitionCycle*, if DMA is used.

Examples:**Example 1: Scan lists**

Selecting the analog input channels, gain and input voltage range

```

b_RamArraySize           = 16
pb_ChannelArray [0]      = APCI3120_CHANNEL_0
pb_ChannelArray [1]      = APCI3120_CHANNEL_1
pb_ChannelArray [2]      = APCI3120_CHANNEL_2
pb_ChannelArray [3]      = APCI3120_CHANNEL_3
pb_ChannelArray [4]      = APCI3120_CHANNEL_2
pb_ChannelArray [5]      = APCI3120_CHANNEL_1
pb_ChannelArray [6]      = APCI3120_CHANNEL_0
pb_ChannelArray [7]      = APCI3120_CHANNEL_4
pb_ChannelArray [8]      = APCI3120_CHANNEL_5
pb_ChannelArray [9]      = APCI3120_CHANNEL_6
pb_ChannelArray [10]     = APCI3120_CHANNEL_7
pb_ChannelArray [11]     = APCI3120_CHANNEL_8
pb_ChannelArray [12]     = APCI3120_CHANNEL_9
pb_ChannelArray [13]     = APCI3120_CHANNEL_10
pb_ChannelArray [14]     = APCI3120_CHANNEL_11
pb_ChannelArray [15]     = APCI3120_CHANNEL_12

pb_GainArray [0]         = APCI3120_1_GAIN
pb_PolarityArray [0]     = APCI3120_UNIPOLAR
pb_GainArray [1]         = APCI3120_1_GAIN
pb_PolarityArray [1]     = APCI3120_UNIPOLAR
pb_GainArray [2]         = APCI3120_1_GAIN
pb_PolarityArray [2]     = APCI3120_UNIPOLAR
pb_GainArray [3]         = APCI3120_1_GAIN
pb_PolarityArray [3]     = APCI3120_UNIPOLAR
pb_GainArray [4]         = APCI3120_5_GAIN
pb_PolarityArray [4]     = APCI3120_BIPOLAR
pb_GainArray [5]         = APCI3120_5_GAIN
pb_PolarityArray [5]     = APCI3120_BIPOLAR
pb_GainArray [6]         = APCI3120_5_GAIN
pb_PolarityArray [6]     = APCI3120_BIPOLAR
pb_GainArray [7]         = APCI3120_10_GAIN
pb_PolarityArray [7]     = APCI3120_UNIPOLAR
pb_GainArray [8]         = APCI3120_10_GAIN
pb_PolarityArray [8]     = APCI3120_UNIPOLAR
pb_GainArray [9]         = APCI3120_10_GAIN
pb_PolarityArray [9]     = APCI3120_BIPOLAR
pb_GainArray [10]        = APCI3120_1_GAIN
pb_PolarityArray [10]    = APCI3120_UNIPOLAR
pb_GainArray [11]        = APCI3120_1_GAIN
pb_PolarityArray [11]    = APCI3120_UNIPOLAR
pb_GainArray [12]        = APCI3120_2_GAIN
pb_PolarityArray [12]    = APCI3120_UNIPOLAR
pb_GainArray [13]        = APCI3120_2_GAIN
pb_PolarityArray [13]    = APCI3120_UNIPOLAR
pb_GainArray [14]        = APCI3120_2_GAIN
pb_PolarityArray [14]    = APCI3120_UNIPOLAR
pb_GainArray [15]        = APCI3120_1_GAIN
pb_PolarityArray [15]    = APCI3120_UNIPOLAR

```

In this example the priority is set as follows:

Element	Analog input channel	Input voltage range
1	0	0-10 V
2	1	0-10 V
3	2	0-10 V
4	3	0-10 V
5	2	± 2 V
6	1	± 2 V
7	0	± 2 V
8	4	0-1 V
9	5	0-1 V
10	6	± 1 V
11	7	0-10 V
12	8	0-10 V
13	9	0-5 V
14	10	0-5 V
15	11	0-5 V
16	12	0-10 V

Example 2: Cyclic conversion without DMA without external trigger

```

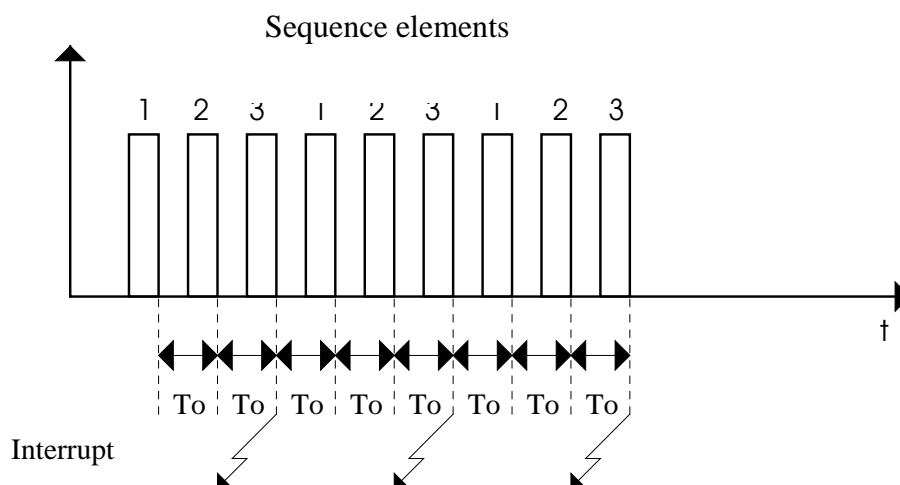
i_APCI3120_InitAnalogInputAcquisition
(b_BoardHandle,
3,
pb_ChannelArray,
pb_GainArray,
pb_PolarityArray,
APCI3120_SIMPLE_MODUS,
APCI3120_DISABLE,
T0,
0,
3,
APCI3120_DMA_NOT_USED,
APCI3120_SINGLE)

```

```

- b_AcquisitionMode      = APCI3120_SIMPLE_MODUS
- b_ExternTrigger        = APCI3120_DISABLE
- ui_AcquisitionTiming   = T0
- b_DMAUsed              = APCI3120_DMA_NOT_USED
- ui_NumberOfAquisition = 3

```



Example 3: Cyclic conversion with DMA without external trigger

```

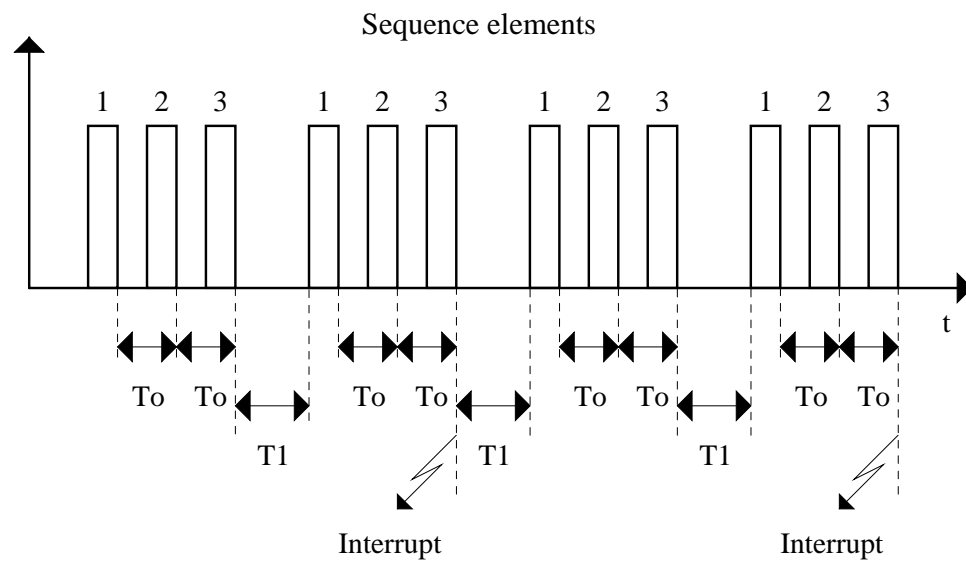
i_APCI3120_InitAnalogInputAcquisition
(b_BoardHandle,
3,
pb_ChannelArray,
pb_GainArray,
pb_PolarityArray,
APCI3120_DELAY_MODUS,
APCI3120_DISABLE,
T0,
T1,
6,
APCI3120_DMA_USED,
APCI3120_CONTINUOUS)

```

```

- b_AcquisitionMode      = APCI3120_DELAY_MODUS
- b_ExternTrigger        = APCI3120_DISABLE
- ui_NumberOfAcquisition = 6
- ui_AcquisitionTiming   = T0
- l_DelayTiming          = T1
- b_DMAUsed              = APCI3120_DMA_USED
- b_AcquisitionCycle     = APCI3120_CONTINUOUS
- b_ExternTrigger        = APCI3120_DISABLE

```



Example 4: Cyclic conversion with DMA without external trigger

```

i_APCI3120_InitAnalogInputAcquisition
(b_BoardHandle,
3,
pb_ChannelArray,
pb_GainArray,
pb_PolarityArray,
APCI3120_DELAY_1_MODUS,
APCI3120_DISABLE,
T0,
T1,
6,
APCI3120_DMA_USED,
APCI3120_CONTINUOUS)

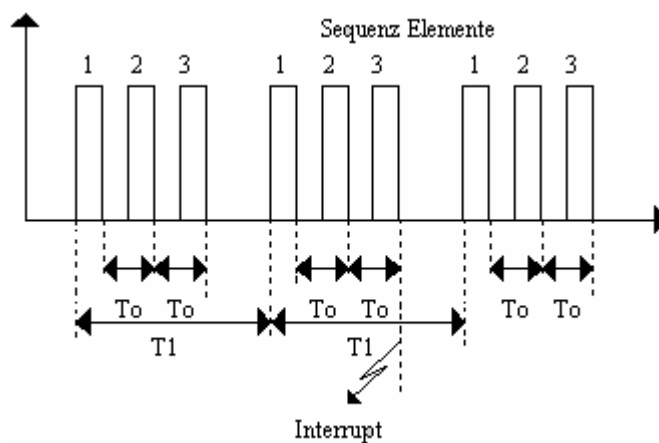
```

```

- b_AcquisitionMode      = APCI3120_DELAY_1_MODUS
- b_ExternTrigger        = APCI3120_DISABLE
- ui_NumberOfAcquisition = 6
- ui_AcquisitionTiming   = T0
- l_DelayTiming          = T1
- b_DMAUsed              = APCI3120_DMA_USED
- b_AcquisitionCycle     = APCI3120_CONTINUOUS
- b_ExternTrigger        = APCI3120_DISABLE

```

Sequence elements



Calling convention:ANSI C :

```

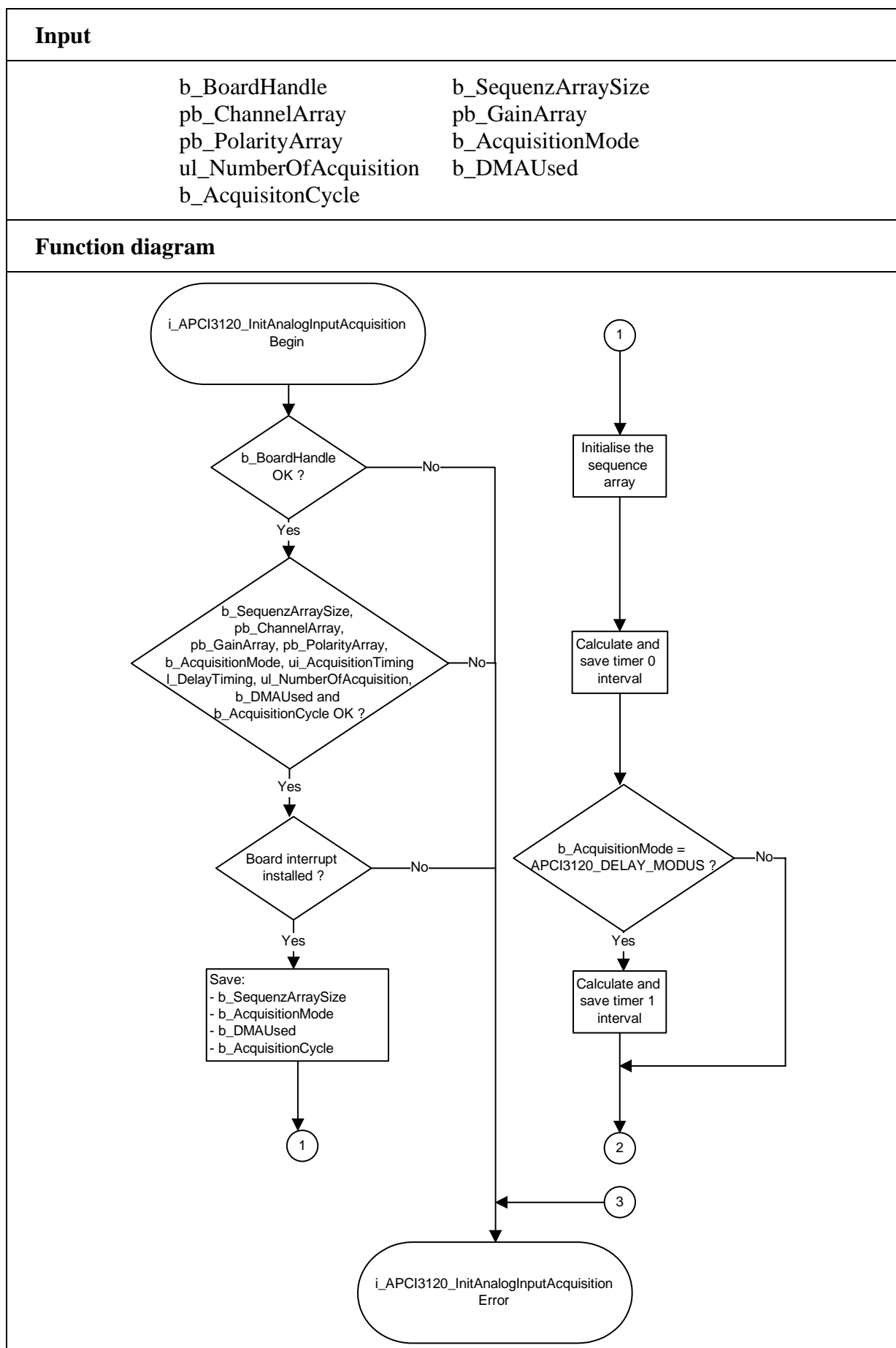
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned int  ui_AnalogInputValue;
unsigned char b_ChannelArray    [16];
unsigned char b_GainArray       [16];
unsigned char b_PolarityArray   [16];

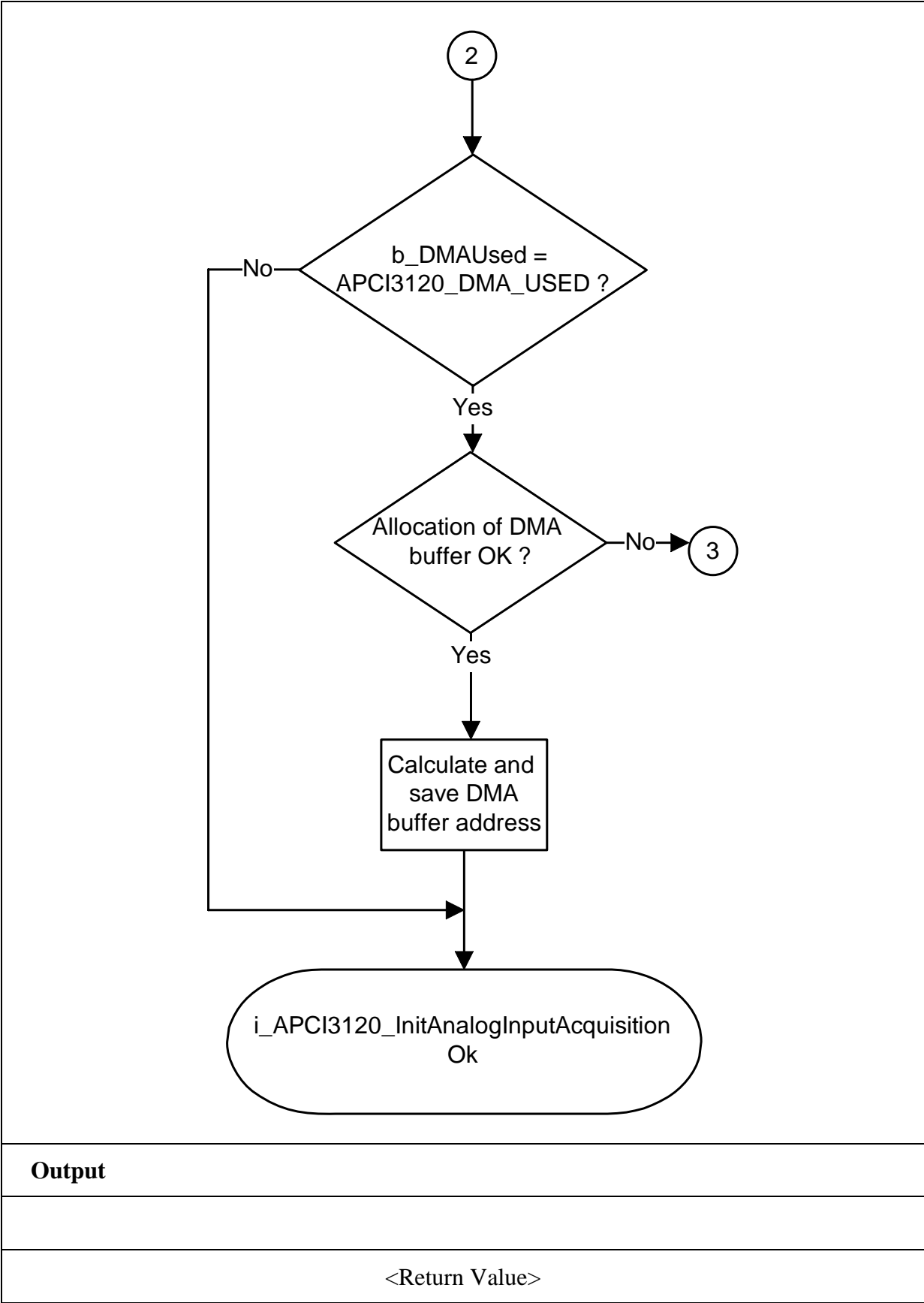
i_ReturnValue = i_APCI3120_InitAnalogInputAcquisition
                (b_BoardHandle,
                 16,
                 b_ChannelArray,
                 b_GainArray,
                 b_PolarityArray,
                 APCI3120_DELAY_MODUS,
                 APCI3120_DISABLE,
                 100,
                 3000,
                 1000,
                 DMA_NOT_USED,
                 APCI3120_SINGLE);

```

Return value:

- 0 : No error
- 1 : The handle parameter of the board is wrong
- 2 : User interrupt routine has not been installed.
See function "i_APCI3120_SetBoardIntRoutine"
- 3 : Size of the scan list is wrong
- 4 : Wrong parameter detected in table "pb_ChannelArray"
- 5 : Wrong parameter detected in table "pb_GainArray"
- 6 : Wrong parameter detected in table "pb_PolarityArray"
- 7 : Waiting time between two conversion cycles is too long
- 8 : The selected time for *ui_AcquisitionTiming* or *l_DelayTiming* is wrong
- 9: Parameter *b_DMAUsed* is wrong
- 10: Parametered running time of the DMA conversion cycle is wrong
(APCI3120_CONTINUOUS or APCI3120_SINGLE)
- 11: Parametered conversion cycle is wrong
(APCI3120_SIMPLE_MODUS or APCI3120_DELAY_MODUS)
- 12: Not enough memory available.
- 13: The parameter of the external trigger is wrong .
- 14: The number of acquisitions is wrong.





2) i APCI3120 StartAnalogInputAcquisition (...)

Syntax:

<Return value> = i_APCI3120_StartAnalogInputAcquisition
(BYTE b BoardHandle)

Parameters:

- Input:

BYTE	b BoardHandle	Handle of the board
------	---------------	---------------------

- Output:

No output signal has occurred

Task:

Starts the cyclic conversion. It has been previously initialised with function „i_APCI3120_InitAnalogInputAcquisition“.

Calling convention:

ANSI C :

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

```
i_ReturnValue = i_APCI3120_StartAnalogInputAcquisition
                                     (b_BoardHandle);
```

Return value:

0: No error

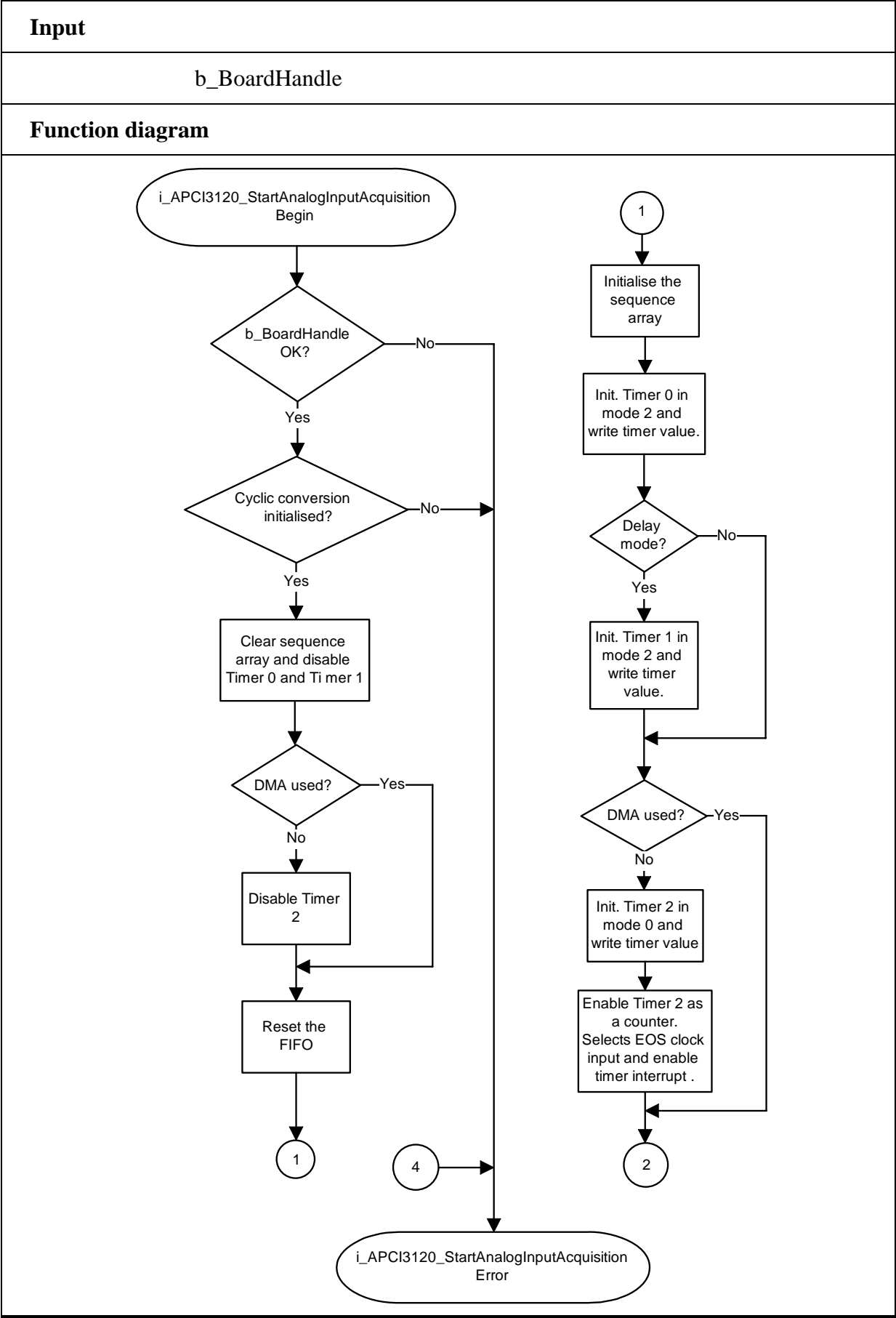
-1: The handle parameter of the board is wrong

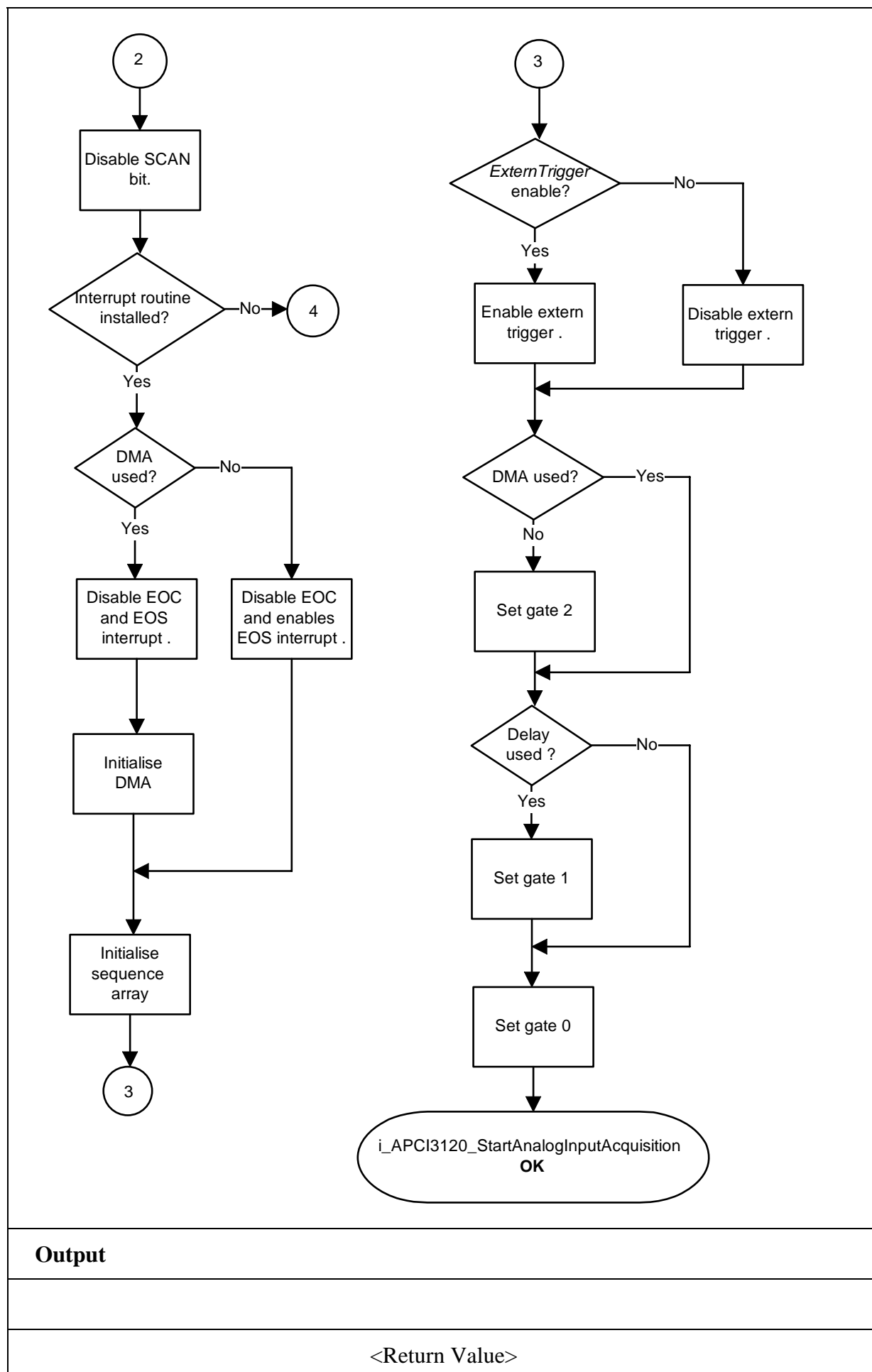
-2: The cyclic conversion has not been initialised.

Please use function "i_APCI3120_InitAnalogInputAcquisition"

-3: The user interrupt routine is not installed

See the function "i_APCI3120_InitAnalogInputAcquisition"





3) i_APCI3120_StopAnalogInputAcquisition (...)

Syntax:

< Return value > = i_APCI3120_StopAnalogInputAcquisition
(BYTE b_BoardHandle)

Parameters:**- Input:**

BYTE b_BoardHandle Handle of the board

- Output:

No output signal has occurred

Task:

Stops the cyclic conversion. It has been previously started with function „i_APCI3120_StartAnalogInputAcquisition“

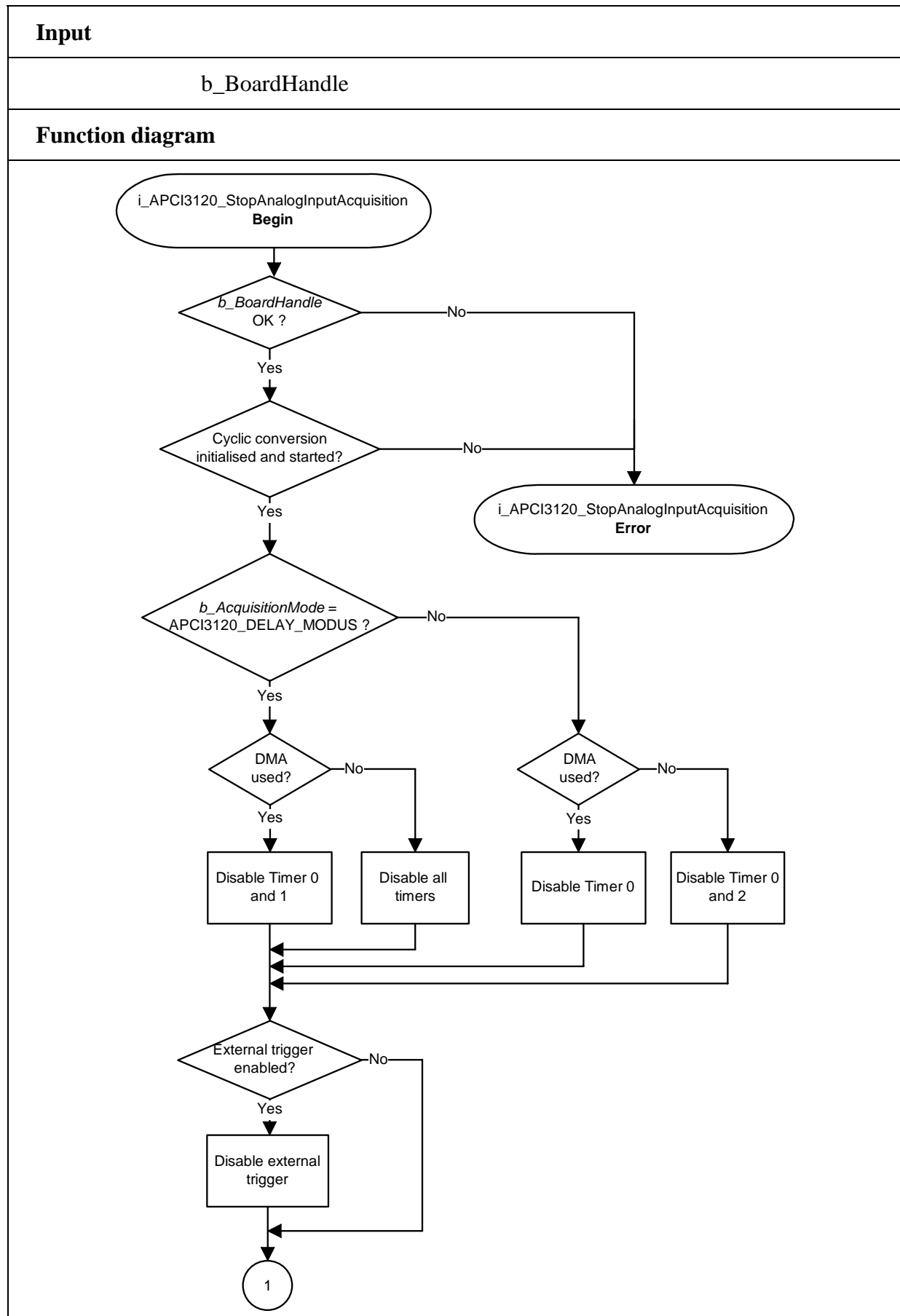
Calling convention:ANSI C:

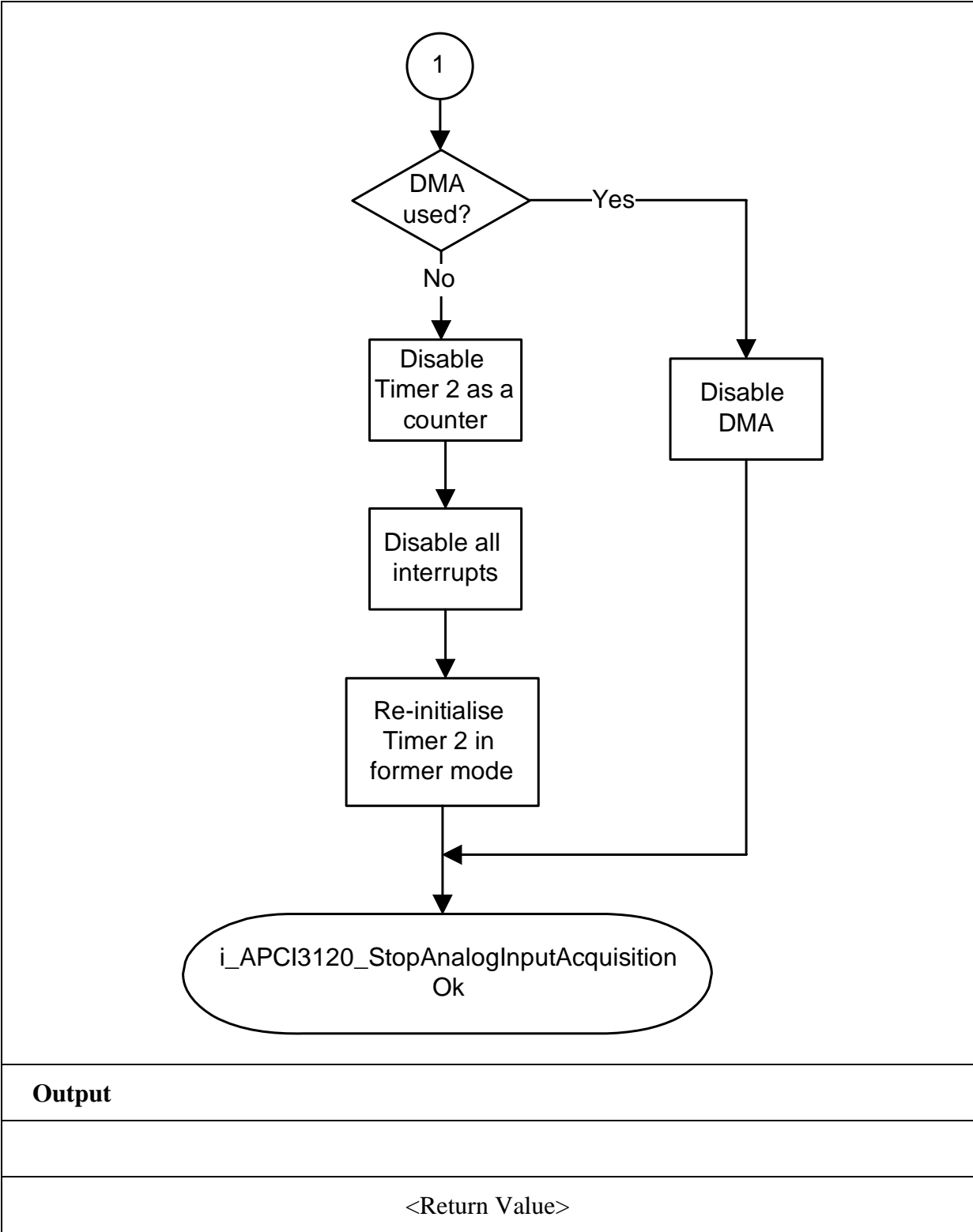
```
int                      i_ReturnValue;  
unsigned char    b_BoardHandle;
```

```
i_ReturnValue = i_APCI3120_StopAnalogInputAcquisition (b_BoardHandle);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The cyclic conversion has not been started.
Please use function "i_APCI3120_StartAnalogInputAcquisition"





4) i_APCI3120_ClearAnalogInputAcquisition(...)

Syntax:

< Return value > = i_APCI3120_ClearAnalogInputAcquisition
(BYTE b_BoardHandle)

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of the board
------	---------------	---------------------

- Output:

No output signal has occurred

Task:

Releases the DMA buffer.

Calling convention:

ANSI C :

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

```
i_ReturnValue = i_APCI3120_ClearAnalogInputAcquisition
                                     (b_BoardHandle);
```

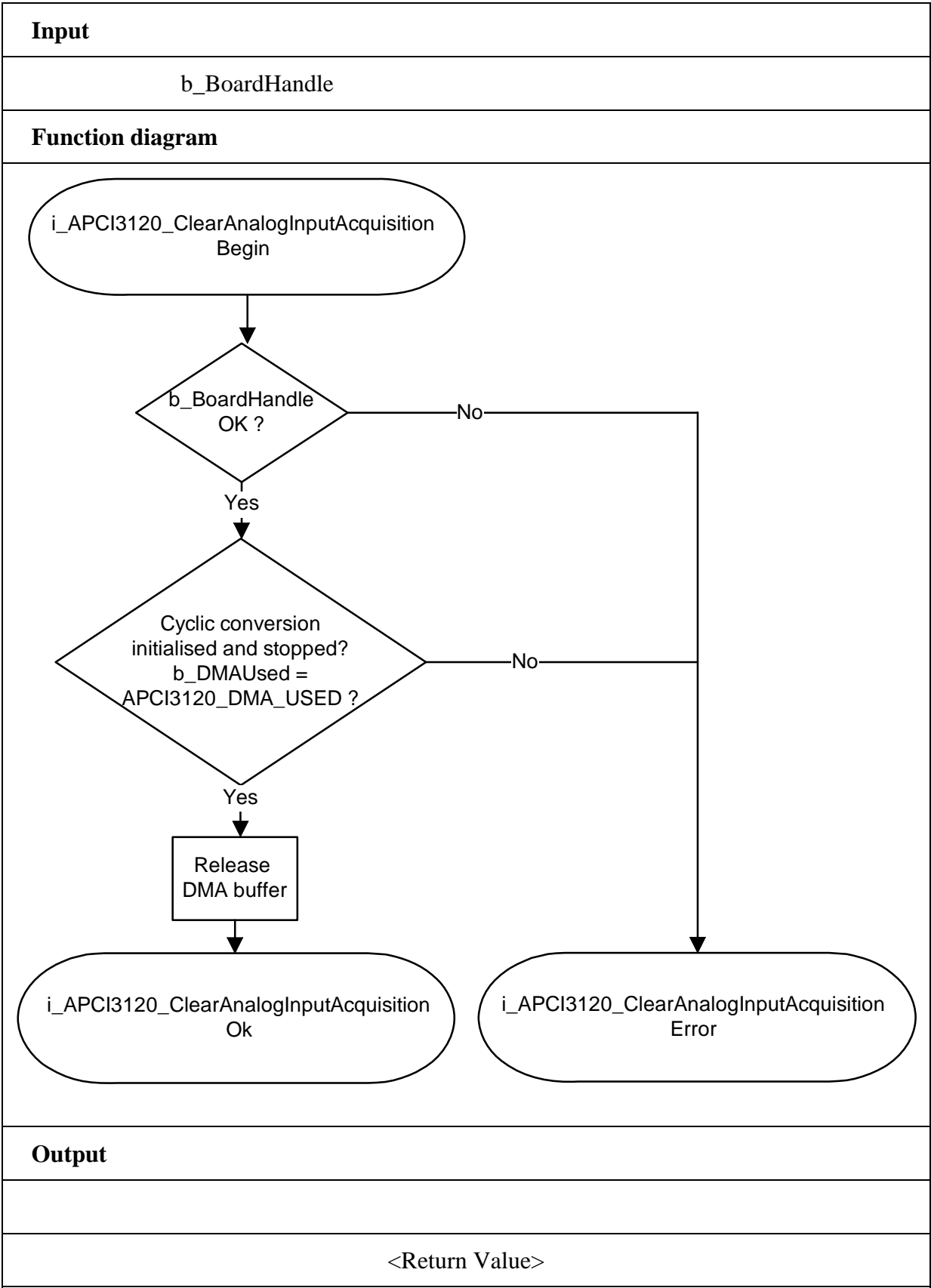
Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: The cyclic conversion has not been initialised.

Please use the function "i_APCI3120_InitAnalogInputAcquisition"



3.5 Analog output channels

1) i_APCI3120_Write1AnalogValue (...)

Syntax:

```
<Return value> = i_APCI3120_Write1AnalogValue
                    (BYTE      b_BoardHandle,
                     BYTE      b_ChannelNbr,
                     BYTE      b_Polarity,
                     UINT       ui_ValueToWrite)
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of the board
BYTE	b_ChannelNbr	Number of the analog output channel
BYTE	b_Polarity	Selection of the output voltage range.
UINT	ui_ValueToWrite	Analog output value to write in UNIPOLAR_MODE: 0 to 8192 in BIPOLAR_MODE: 0 to 16383

- Output:

No output signal has occurred.

Task:

Writes an analog value (*ui_ValueToWrite*) on the analog output channel *b_ChannelNbr*.

Calling convention:

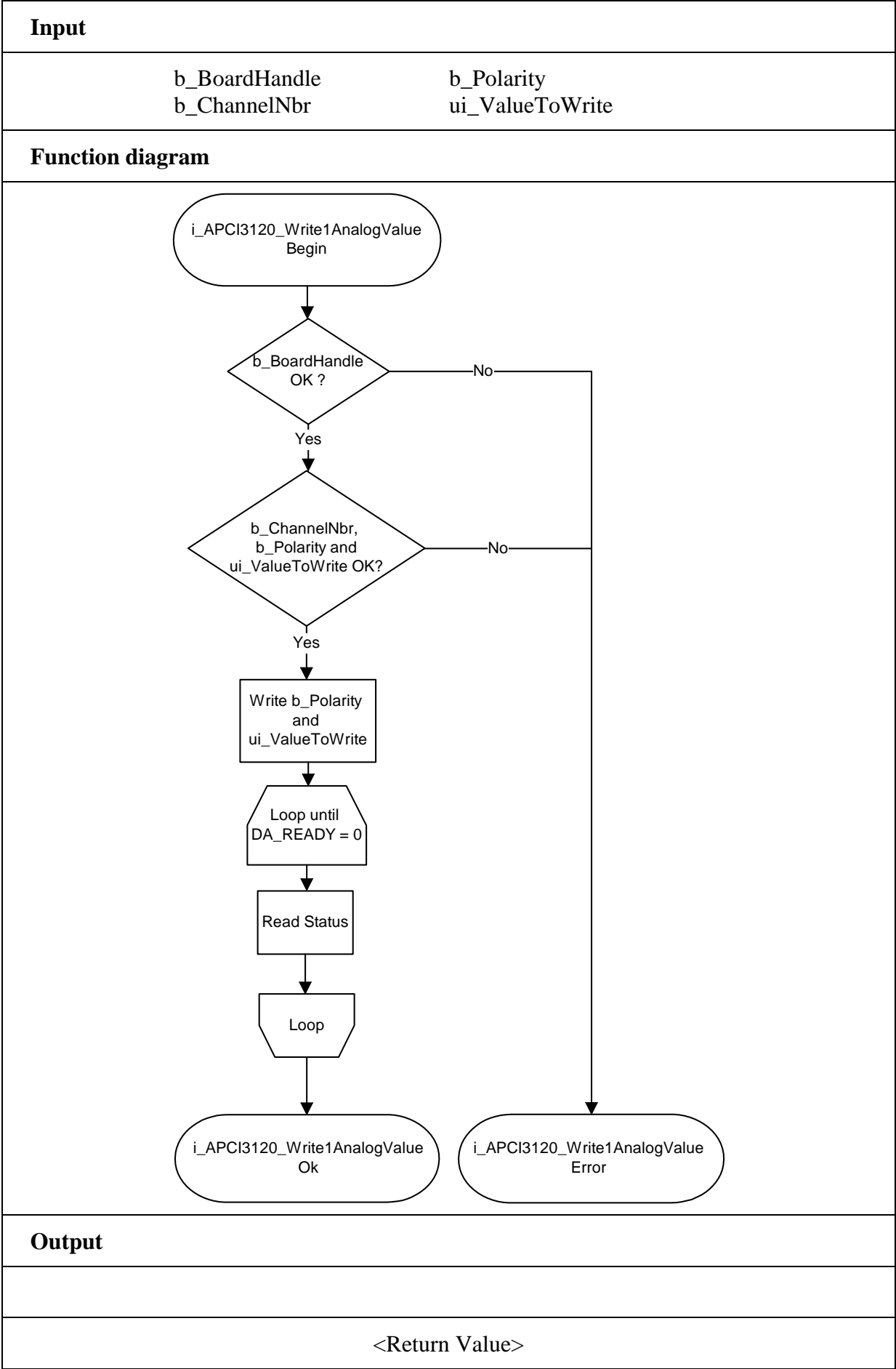
ANSI C:

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

```
i_ReturnValue = i_APCI3120_Write1AnalogValue (b_BoardHandle,
                                                1,
                                                APCI3120_UNIPOLAR
                                                4095);
```

Return value:

0: No error
 -1: The handle parameter of the board is wrong
 -2: Number of the analog output channel is wrong.
 -3: The output voltage range selected is wrong. See table 3-6
 -4: Output value too high



2) i_APCI3120_WriteMoreAnalogValue (...)**Syntax:**

```
<Return value> = i_APCI3120_WriteMoreAnalogValue
                    (BYTE      b_BoardHandle,
                     BYTE      b_FirstChannelNbr,
                     BYTE      b_NbrOfChannel,
                     PBYTE     pb_PolarityArray,
                     PUINT     pui_ValueArray)
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of the board
BYTE	b_FirstChannelNbr	Number of the first analog output channel (0 to 7)
BYTE	b_NbrOfChannel	Number of analog output channels you wish to write on (1 to 8)
PBYTE	pb_PolarityArray	Scan list for the input voltage range See table 3-6.
PUINT	pui_ValueArray	Table of analog output values on which you want to write

- Output:

No output signal has occurred.

Task:

Writes several analog values on several analog output channels.

The variable *b_FirstChannelNbr* defines the first analog output channel.

The variable *b_NbrOfChannel* defines the number of analog output channels .

Table 3-6: Selection of the output voltage range

pb_PolarityArray or b_Polarity Parameter	Voltage range	Define decimal value
APCI3120_UNIPOLAR	0-10V	128
APCI3120_BIPOLAR	±10V	0

Example:

Parameter

```

b_FirstChannelNbr  = 2
b_NbrOfChannel     = 3

pui_ValueArray [0] = 0
pui_ValueArray [1] = 8192
pui_ValueArray [2] = 8192
pb_PolarityArray[0] = APCI3120_UNIPOLAR
pb_PolarityArray[1] = APCI3120_BIPOLAR
pb_PolarityArray[2] = APCI3120_UNIPOLAR

```

The value 0 (0V) is written in the buffer of analog output 2
 The value 8192 (0V) is written in the buffer of analog output 3
 The value 8192 (10V) is written in the buffer of analog output 4.

Calling convention:ANSI C:

```

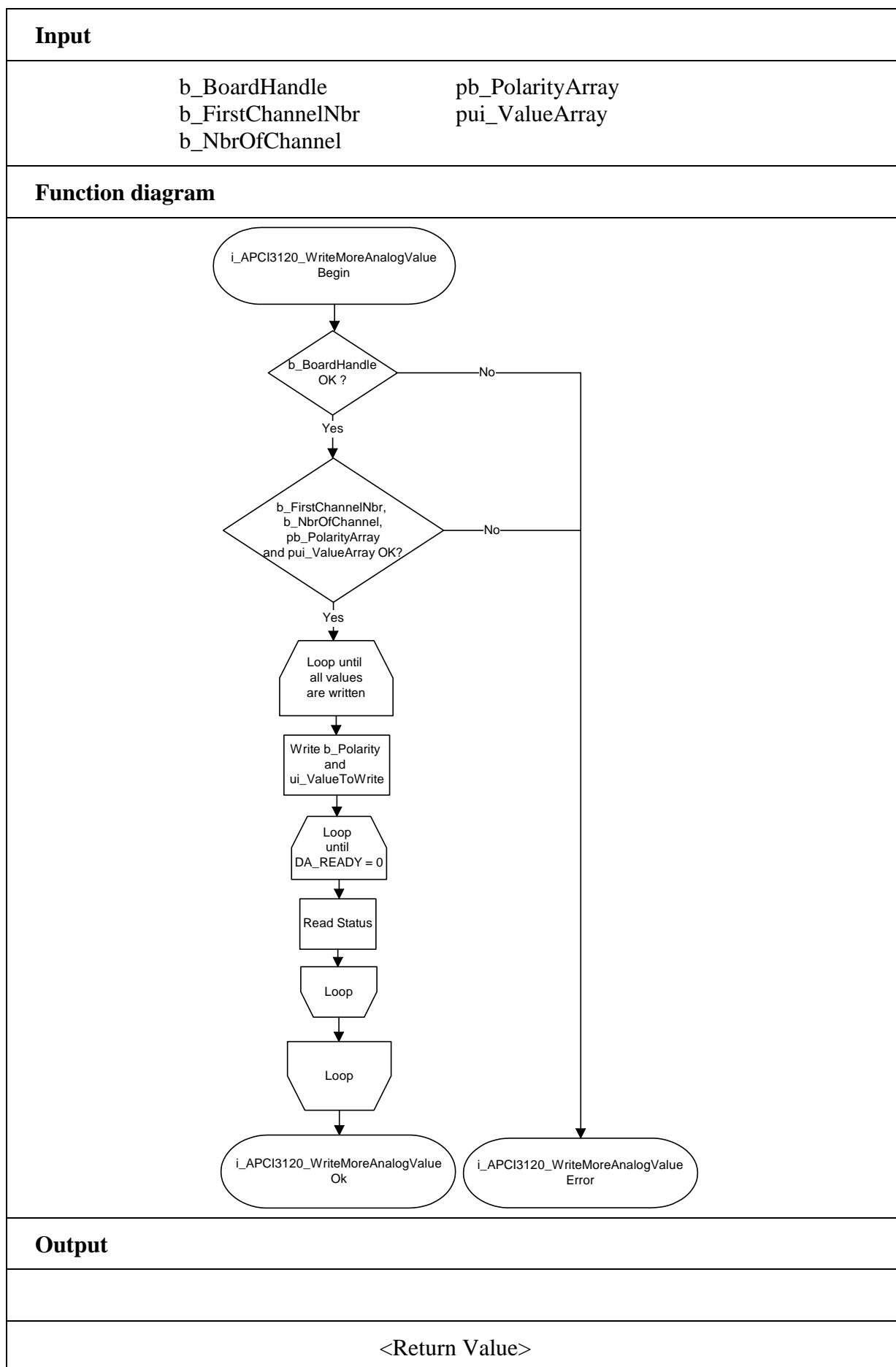
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned char b_PolarityArray [8];
unsigned int  ui_ValueArray  [8];

i_ReturnValue = i_APCI3120_WriteMoreAnalogValue
                (b_BoardHandle,
                 0,
                 8,
                 b_PolarityArray,
                 ui_ValueArray);

```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: Number of the analog output channel is wrong
- 3: The number of analog output channels you wish to write on is wrong
 See function "i_APCI3120_SetBoardInformation"
- 4: One or several polarity selections are wrong.
- 5: One or several output value are too high.



3.6 Timer

1) i_APCI3120_InitTimerWatchdog (...)

Syntax:

```
<Return value> = i_APCI3120_InitTimerWatchdog
                    (BYTE  b_BoardHandle,
                     BYTE  b_TimerMode,
                     LONG  l_DelayValue,
                     BYTE  b_InterruptFlag)
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of the board
BYTE	b_TimerMode	Defines the mode of the timer
		- APCI3120_TIMER: The timer/watchdog is used as an edge generator.
		- APCI3120_WATCHDOG: The timer/watchdog is used as a watchdog for the analog output channels.
LONG	l_DelayValue	Time interval (watchdog time) of the timer
		Timer_Mode: 100 μ s to 838.8 s
		Watchdog_Mode: 50 μ s to 838.8 s
BYTE	b_InterruptFlag	APCI3120_ENABLE:
		<u>Timer</u> : an interrupt is generated at the end of each time interval
		<u>Watchdog</u> : an interrupt is generated when the watchdog has run down
		APCI3120_DISABLE: No interrupt is generated.

- Output:

No output signal has occurred

Task:

Initialises the timer as an edge generator or as a watchdog for the analog output channels.

Calling convention:

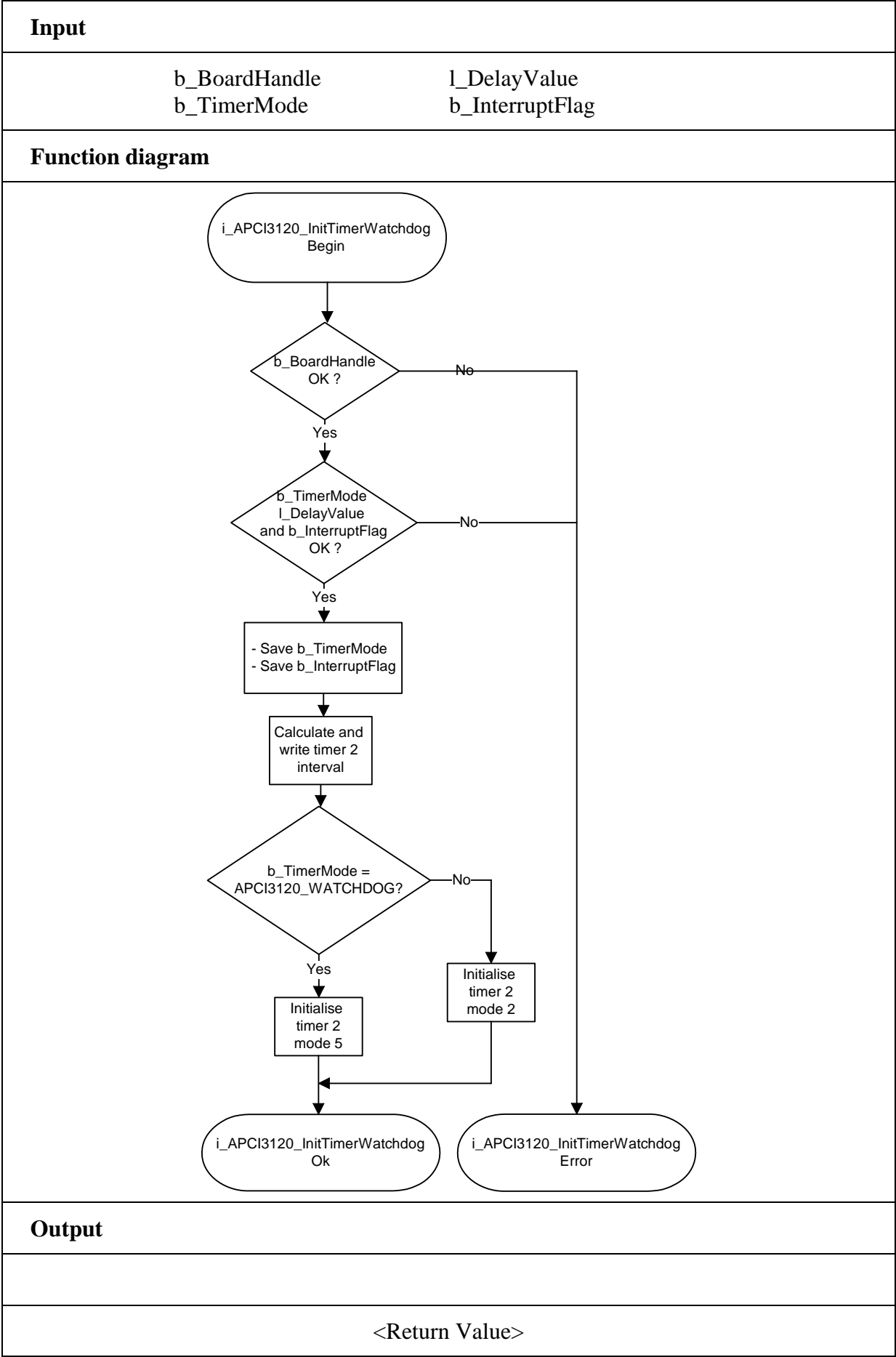
ANSI C :

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

```
i_ReturnValue = i_APCI3120_InitTimerWatchdog (b_BoardHandle,
                                                APCI3120_UNIPOLAR,
                                                1000,
                                                APCI3120_DISABLE);
```

Return value:

- 0: No error.
- 1: The handle parameter of the board is wrong.
- 2: The mode parametered for the timer is wrong.
- 3: The user interrupt routine has not been installed
See function "i_APCI3120_SetBoardIntRoutine"
- 4: The interrupt parameter is wrong
- 5: Time selection is wrong



2) i_APCI3120_StartTimerWatchdog (...)**Syntax:**

<Return value> = i_APCI3120_StartTimerWatchdog (BYTE b_BoardHandle)

Parameters:**- Input:**

BYTE b_BoardHandle Handle of the board

- Output:

No output signal has occurred.

Task:

Starts the timer/watchdog.

Calling convention:

ANSI C :

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

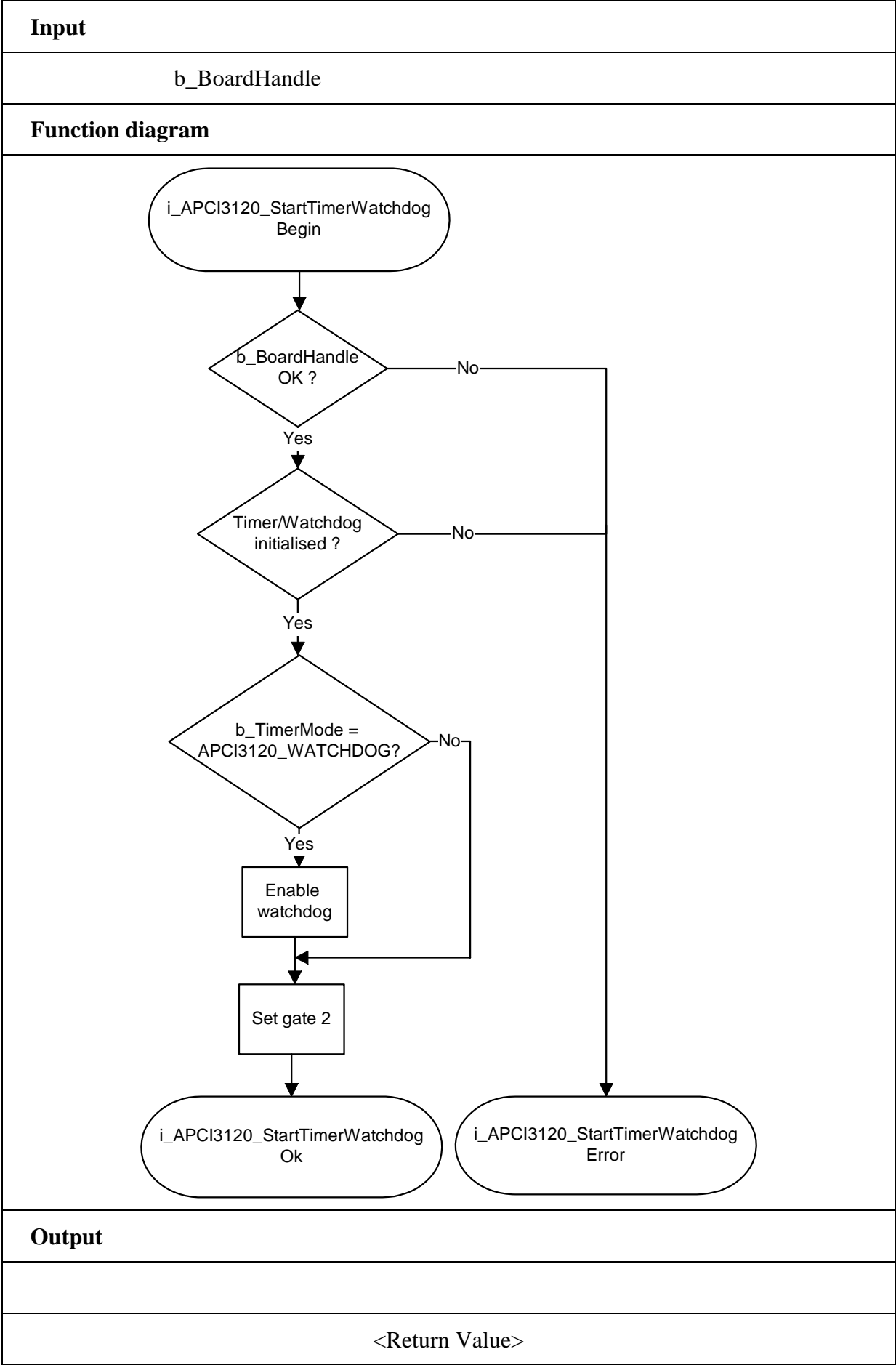
```
i_ReturnValue = i_APCI3120_StartTimerWatchdog (b_BoardHandle);
```

Return Value:

0: No error.

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

-2: Timer/watchdog has not been initialised.



3) i_APCI3120_StopTimerWatchdog (...)

Syntax:

<Return value> = i_APCI3120_StopTimerWatchdog
(BYTE b_BoardHandle)

Parameters:**- Input:**

BYTE b_BoardHandle Handle of the board **xPCI-3120**

- Output:

No output signal has occurred.

Task:

Stops the timer/watchdog.

Calling convention:

ANSI C:

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

```
i_ReturnValue = i_APCI3120_StopTimerWatchdog (b_BoardHandle);
```

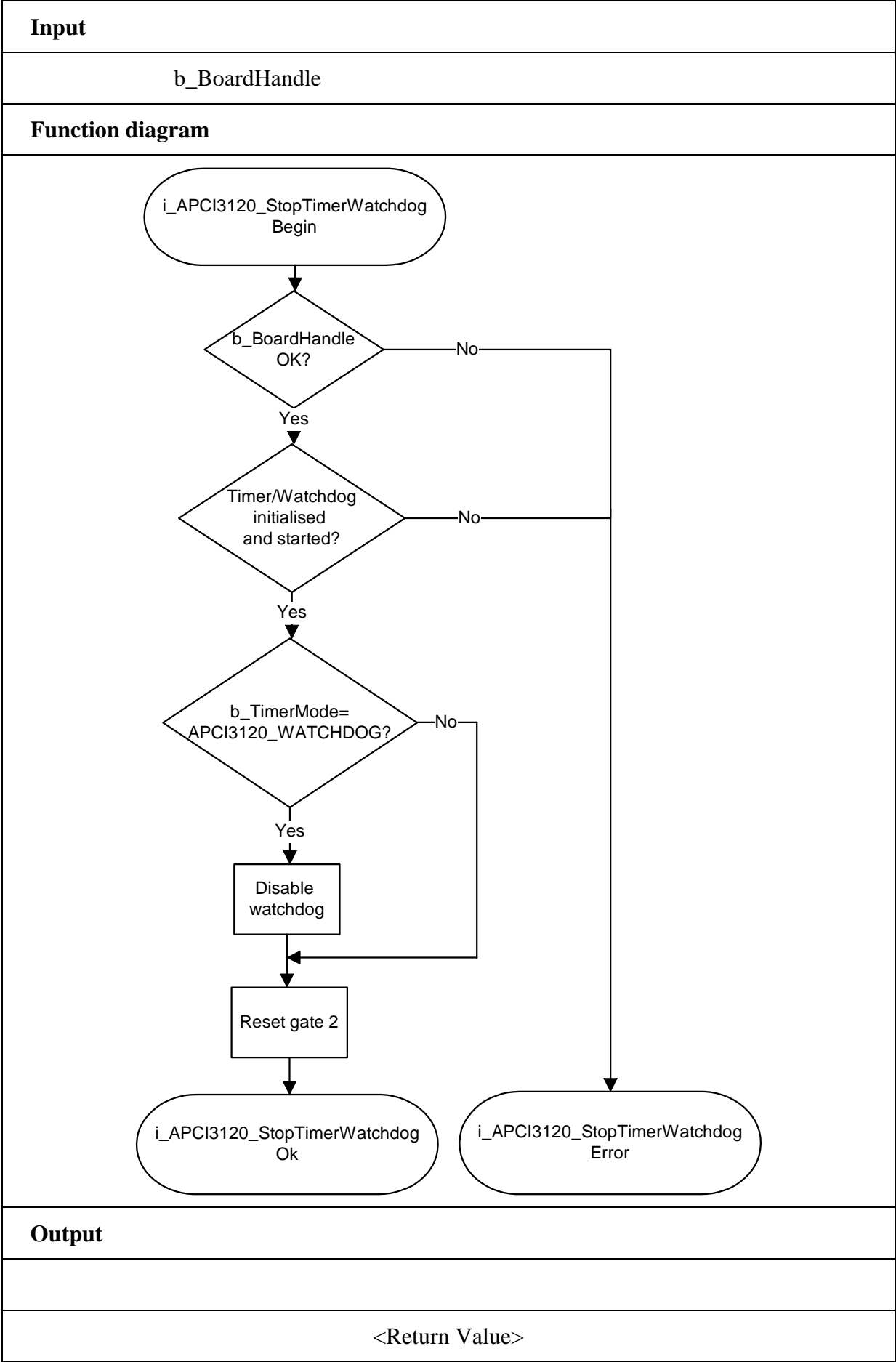
Return Value:

0: No error.

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

-2: Timer/watchdog has not been initialised.

-3: Timer/watchdog has not been started.



4) i_APCI3120_ReadTimer (...)

Syntax:

<Return value> = i_APCI3120_ReadTimer
(BYTE b_BoardHandle
LONG pl_ReadValue)

Parameters:

- Input:

BYTE	b BoardHandle	Handle of the board xPCI-3120
------	---------------	--------------------------------------

- Output:

PLONG	pl_ReadValue	Current timer value (from 0 to FFFFFFF Hex)
-------	--------------	--

Task:

Reads the current value of the timer.

Calling convention:

ANSI C :

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

```
i_ReturnValue = i_APCI3120_ReadTimer(b_BoardHandle,
&l_ReadValue);
```

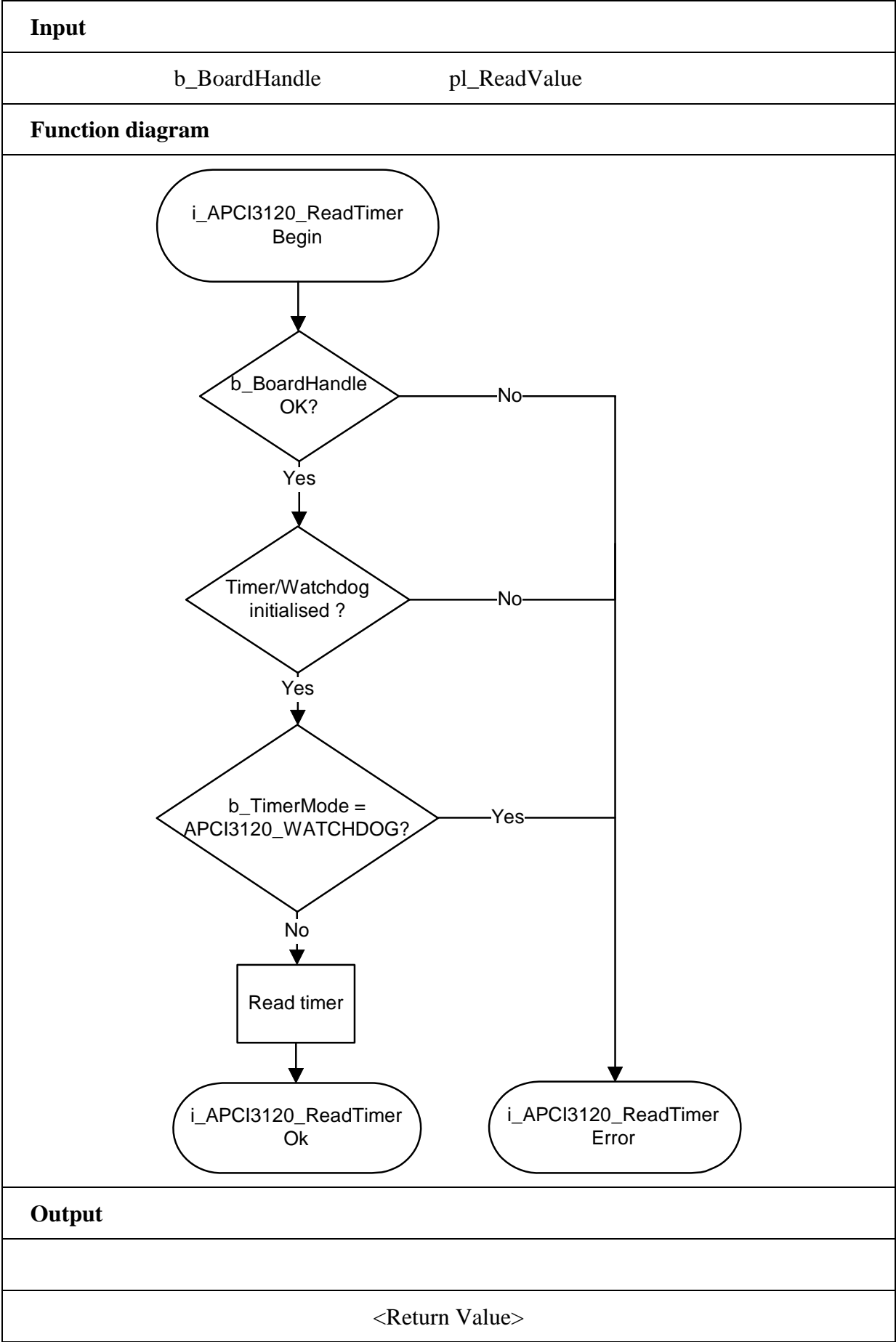
Return value:

0: No error.

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

-2: Timer/watchdog has not been initialised.

-3: Timer/watchdog has been initialised as a watchdog.



5) i_APCI3120_WriteTimer (...)

Syntax:

<Return value> = i_APCI3120_WriteTimer	(BYTE	b_BoardHandle
	LONG	1 WriteValue)

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of the board xPCI-3120
LONG	l_WriteValue	New timer value (from 0 to FFFFFFF Hex)

- Output:

No output signal has occurred

Task:

Writes a new value in the timer.

Calling convention:

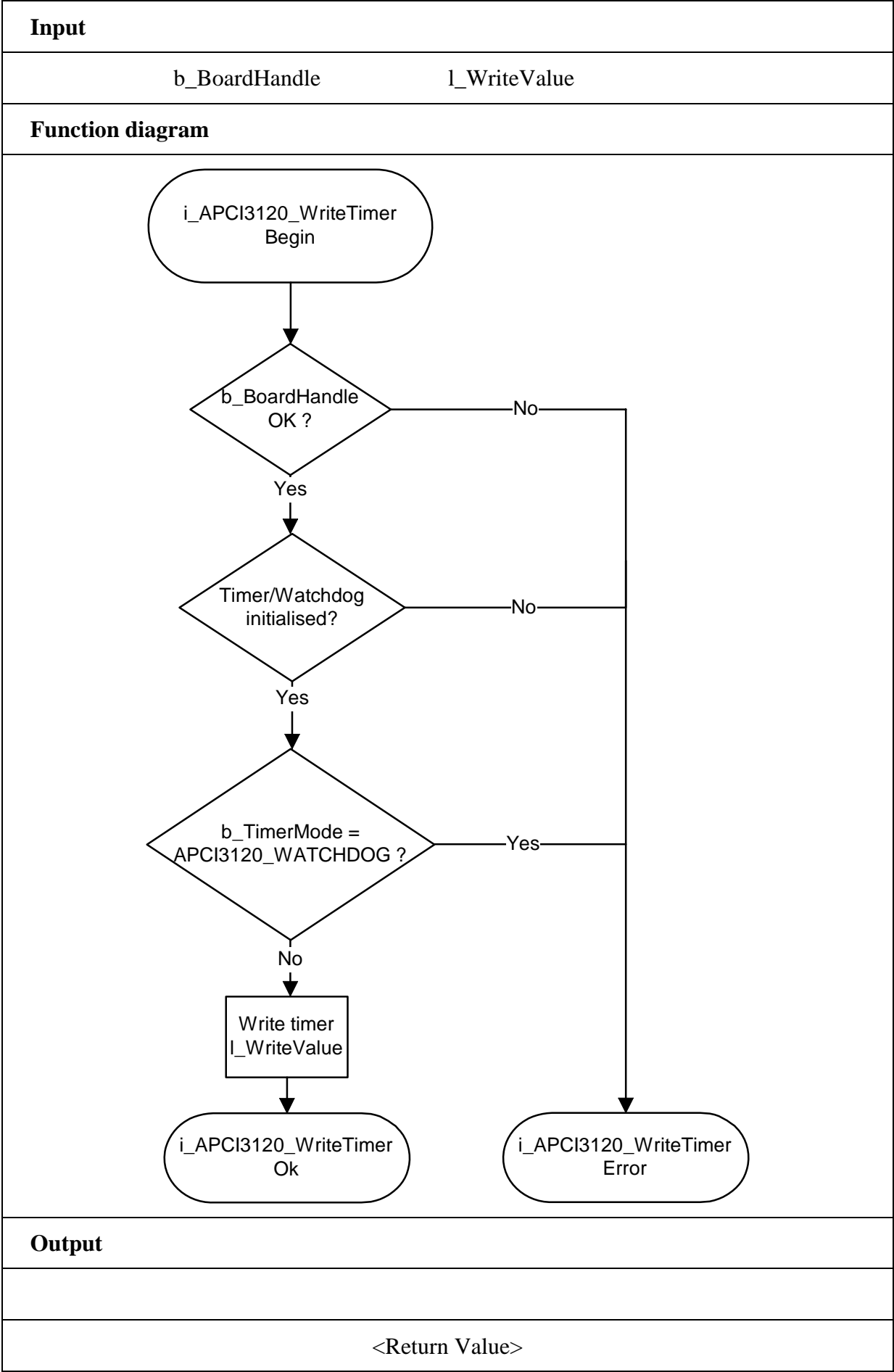
ANSI C :

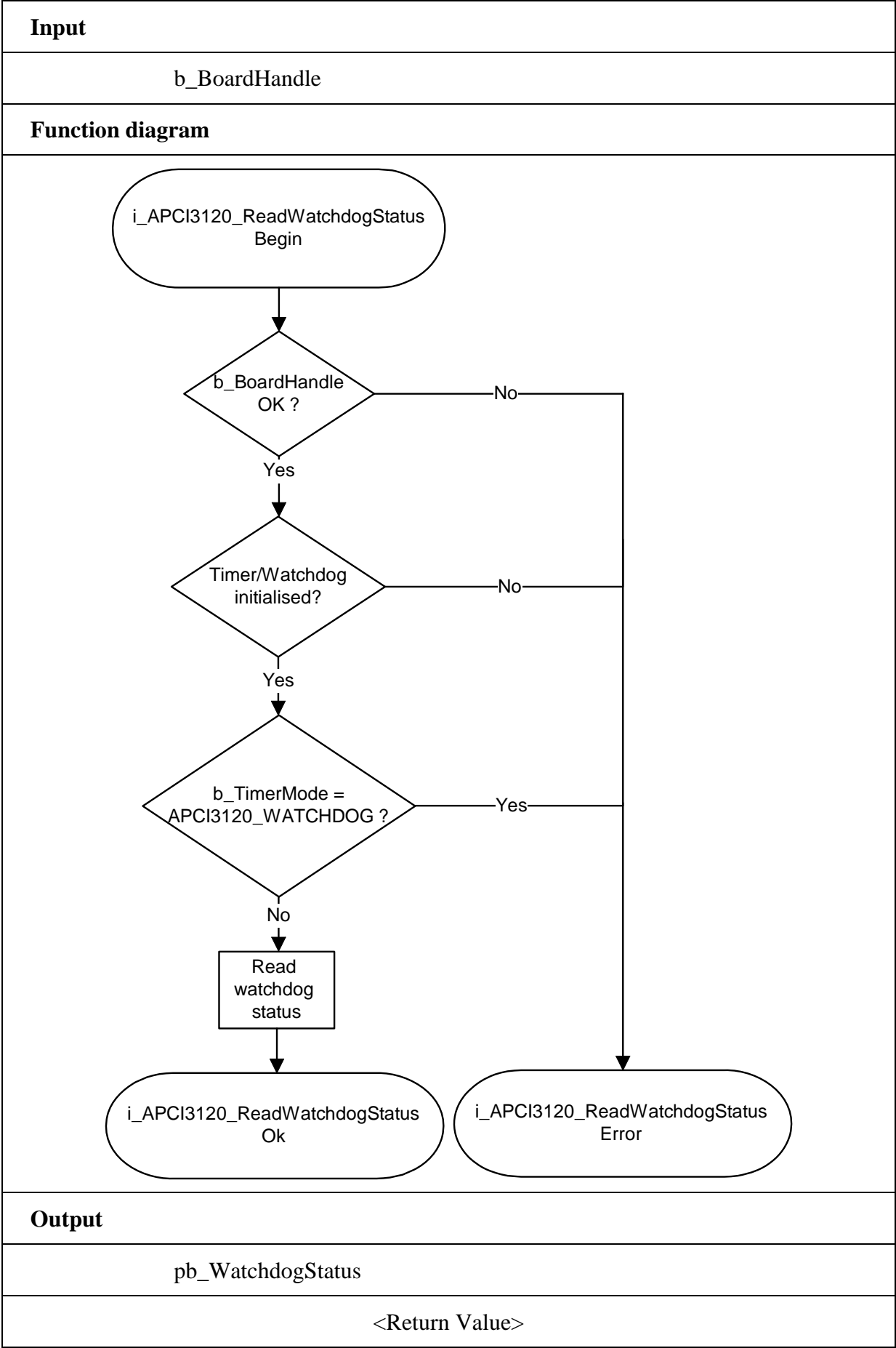
```
int      i_ReturnValue;
unsigned char b_BoardHandle;
```

```
i_ReturnValue = i_APCI3120_WriteTimer(b_BoardHandle,
1000);
```

Return value:

- 0: No error.
- 1: The handle parameter of the board is wrong.
- 2: Timer/watchdog has not been initialised.
- 3: Timer/watchdog has been initialised as a watchdog.





3.7 Digital input channels

1) i_APCI3120_Read1DigitalInput (...)

Syntax:

```
<Return value> = i_APCI3120_Read1DigitalInput
                    (BYTE  b_BoardHandle,
                     BYTE  b_Channel,
                     PBYTE pb_ChannelValue)
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of the board
BYTE	b_Channel	Number of the input channel to be read (1 to 4) .

- Output:

PBYTE	pb_ChannelValue	State of the digital input channel 0 -> Low 1 -> High
-------	-----------------	---

Task:

Indicates the state of an input channel. Enter the input channel to be read with the variable b_Channel (1 to 4). A value is returned with the variable pb_ChannelValue:
0 (Low), 1 (High) .

Calling convention:

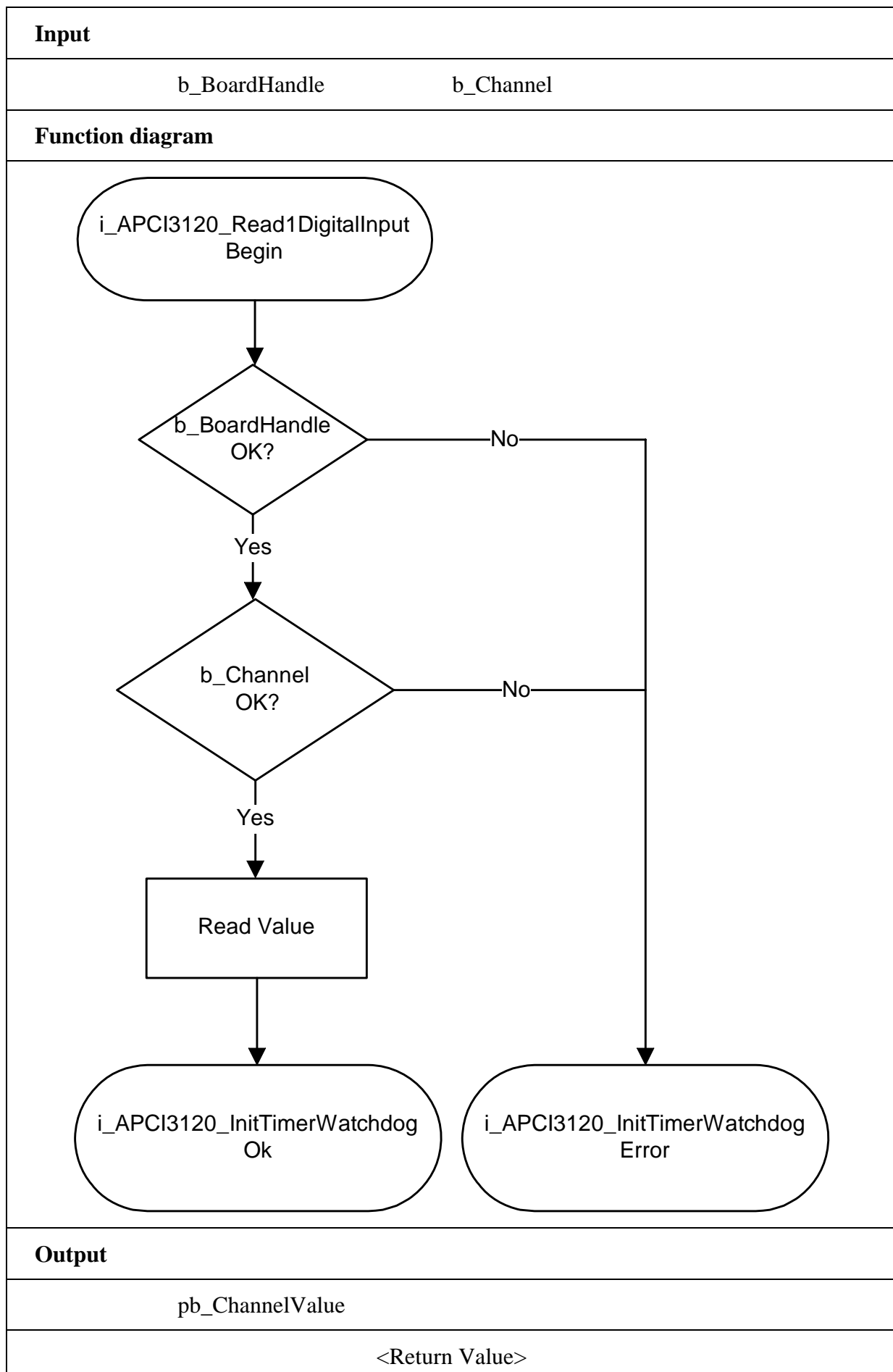
ANSI C :

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

```
i_ReturnValue = i_APCI3120_Read1DigitalInput (b_BoardHandle,
                                                1,
                                                &pb_ChannelValue);
```

Return value:

0: No error.
-1: The handle parameter of the board is wrong.
-2: The input number is not between 1 and 4.



2) i_APCI3120_Read4DigitalInput (...)**Syntax:**

```
<Return value> = i_APCI3120_Read4DigitalInput
                    (BYTE  b_BoardHandle,
                     PBYTE pb_PortValue)
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of the board
------	---------------	---------------------

- Output:

PBYTE	pb_PortValue	State of the digital input port (0 to 15)
-------	--------------	--

Task:

Indicates the state of the port. A value is returned with the variable pb_PortValue.

Calling convention:

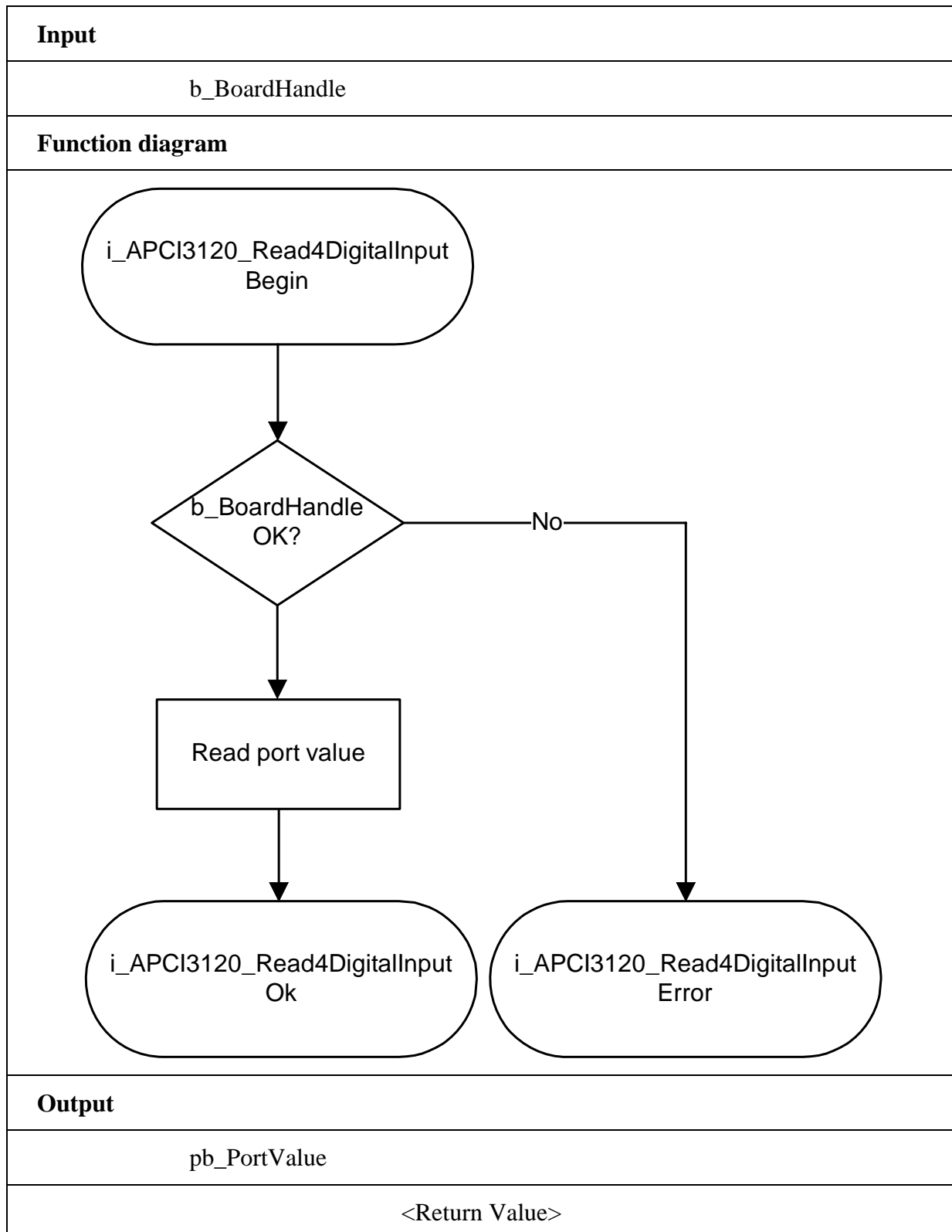
ANSI C:

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

```
i_ReturnValue = i_APCI3120_Read4DigitalInput (b_BoardHandle,
                                              &pb_PortValue);
```

Return value:

0: No error.
-1: The handle parameter of the board is wrong.



3.8 Digital output channels

1) i_APCI3120_Set1DigitalOutputOn (...)

Syntax:

```
<Return value> = i_APCI3120_Set1DigitalOutputOn
                    (BYTE  b_BoardHandle,
                     BYTE  b_Channel)
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of the board
BYTE	b_Channel	Number of the output channel to be set (1 to 4) .

- Output:

No output signal has occurred

Task:

Sets the output channel which has been passed with the parameter b_Channel. Setting an output channel means setting an output channel to „High“.

Calling convention:

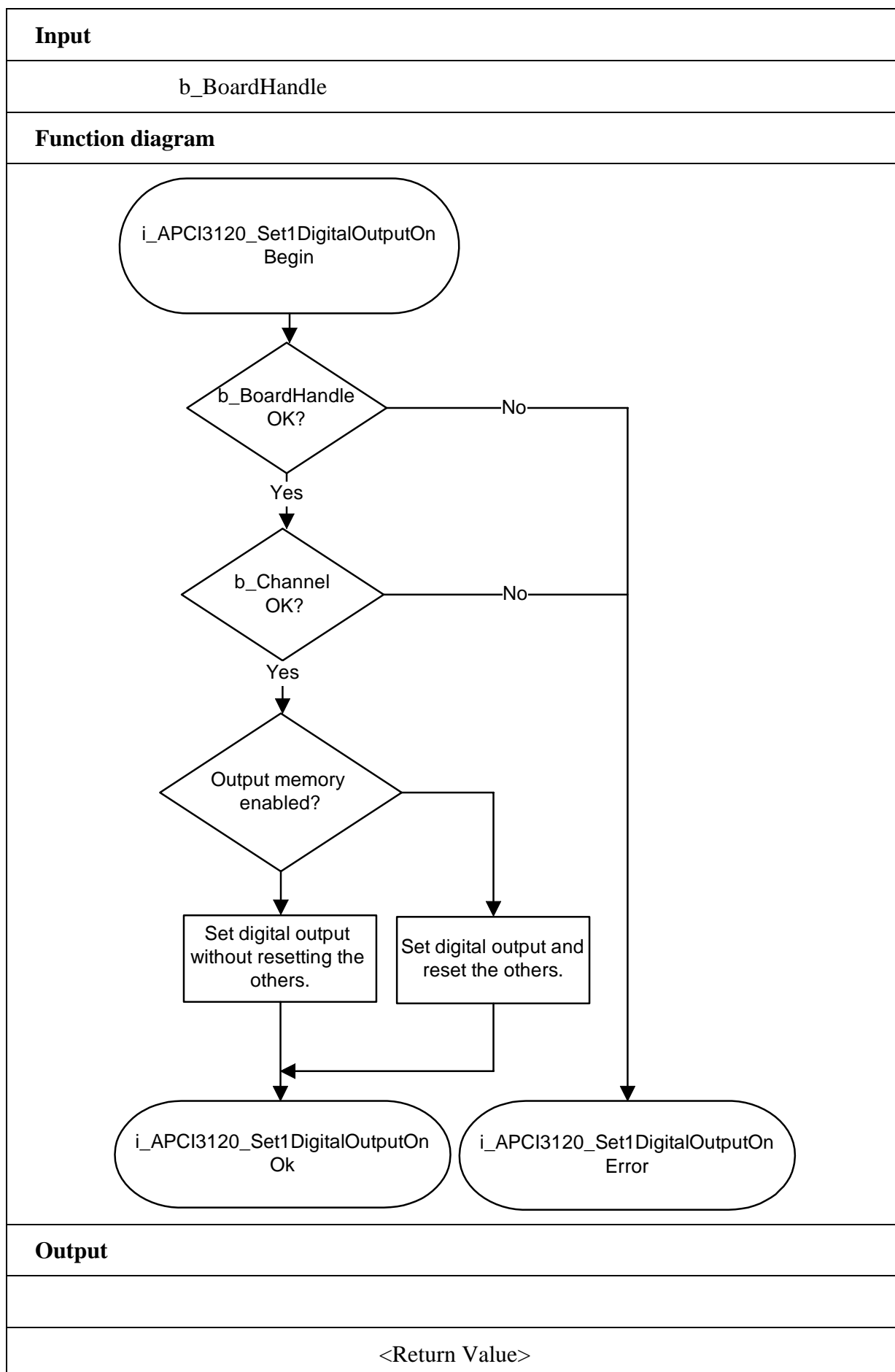
ANSI C:

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

```
i_ReturnValue = i_APCI3120_Set1DigitalOutputOn (b_BoardHandle,
                                                1);
```

Return value:

0: No error.
 -1: The handle parameter of the board is wrong.
 -2: The input number is not between 1 and 4.



2) i_APCI3120_Set1DigitalOutputOff (...)**Syntax:**

```
<Return value> = i_APCI3120_Set1DigitalOutputOff
                    (BYTE  b_BoardHandle,
                     BYTE  b_Channel)
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of the board
BYTE	b_Channel	The number of the output channel to reset (1 to 4) .

- Output:

No output signal has occurred

Task:

Resets the output channel which has been passed with the parameter b_Channel. Resetting an output channel means setting an output channel to „Low“.

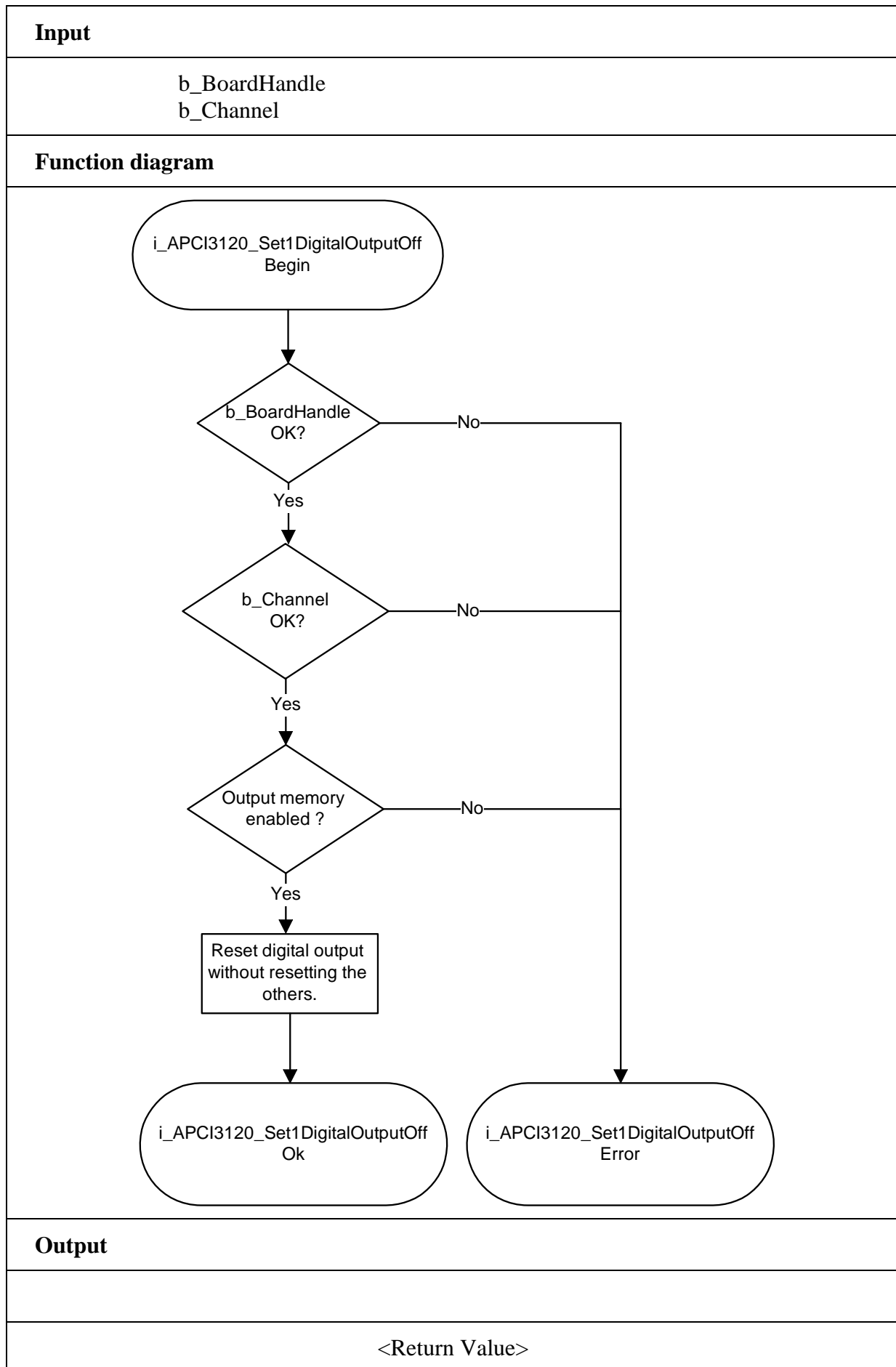
Calling convention:ANSI C:

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

```
i_ReturnValue = i_APCI3120_Set1DigitalOutputOff    (b_BoardHandle,
                                                    1);
```

Return value:

- 0: No error.
- 1: The handle parameter of the board is wrong.
- 2: The output number is not between 1 and 4.
- 3: Digital output memory OFF.
Use previously the function „i_APCI3120_SetOutputMemoryOn“.



3) i_APCI3120_Set4DigitalOutputOn (...)**Syntax:**

```
<Return value> = i_APCI3120_Set4DigitalOutputOn
                    (BYTE  b_BoardHandle,
                     BYTE  b_Value)
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of the board
BYTE	b_Value	Output value (0 to 15)

- Output:

No output signal has occurred

Task:

Sets one or several output channels of a port. Setting an output channel means setting an output channel to „High“. If you have switched OFF the digital output memory (OFF), all other output channels are set to „0“.

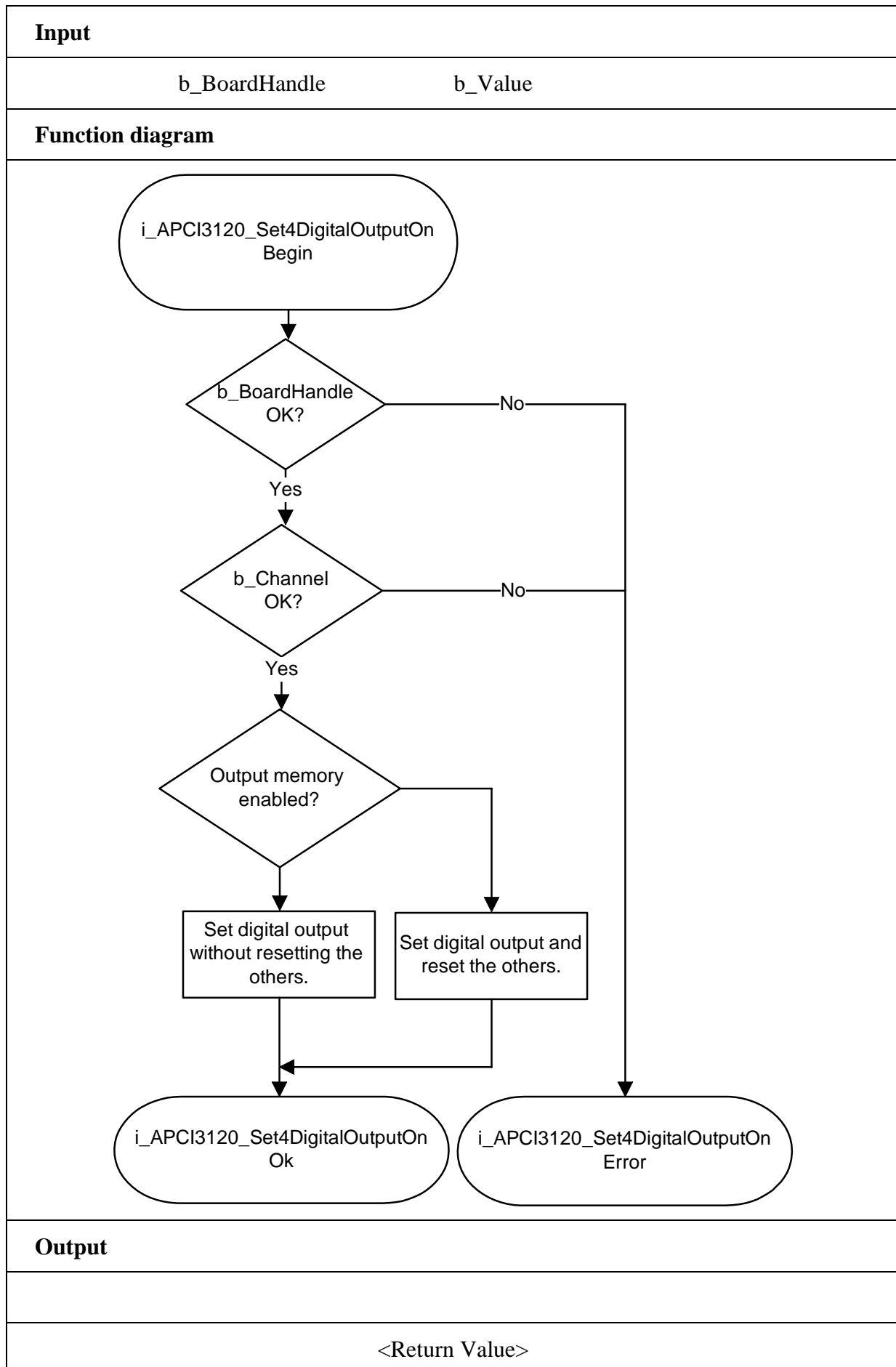
Calling convention:ANSI C :

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

```
i_ReturnValue = i_APCI3120_Set4DigitalOutputOn (b_BoardHandle,
                                                1);
```

Return value:

0: No error.
-1: The handle parameter of the board is wrong.



4) i_APCI3120_Set4DigitalOutputOff (...)**Syntax:**

```
<Return value> = i_APCI3120_Set4DigitalOutputOff
                    (BYTE  b_BoardHandle,
                     BYTE  b_Value)
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of the board
BYTE	b_Value	Output value (0 to 15)

- Output:

No output signal has occurred

Task:

Resets one or several output channels of one port. Resetting means setting to „Low“.

Calling convention:

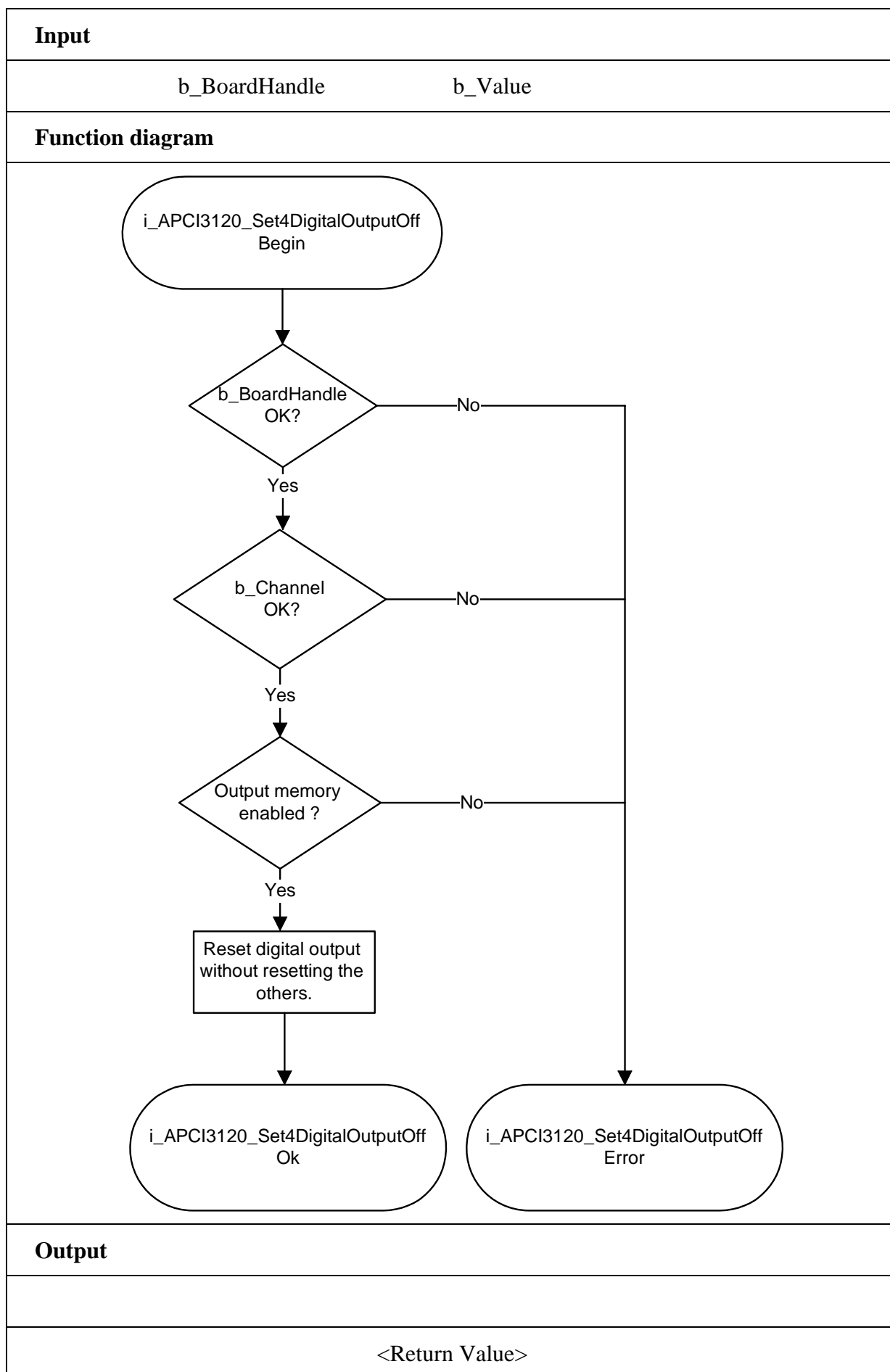
ANSI C :

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

```
i_ReturnValue = i_APCI3120_Set4DigitalOutputOff
                (b_BoardHandle,
                 1);
```

Return value:

- 0: No error.
- 1: The handle parameter of the board is wrong.
- 2: The digital output memory is OFF.
Please use previously the function „i_APCI3120_SetOutputMemoryOn“.



5) i_APCI3120_SetOutputMemoryOn (...)

Syntax:

<Return value> = i_APCI3120_SetOutputMemoryOn
(BYTE b_BoardHandle)

Parameters:

- Input:

BYTE b_BoardHandle Handle of the board

- Output:

No output signal has occurred

Task:

Activates the digital output memory. After calling up this function, the output channels you have previously activated with the function „i_APCI3120_SetXDigitalOutputOn“ are not reset. You can reset them with the function „i_APCI3120_SetXDigitalOutputOff“.

Calling convention:

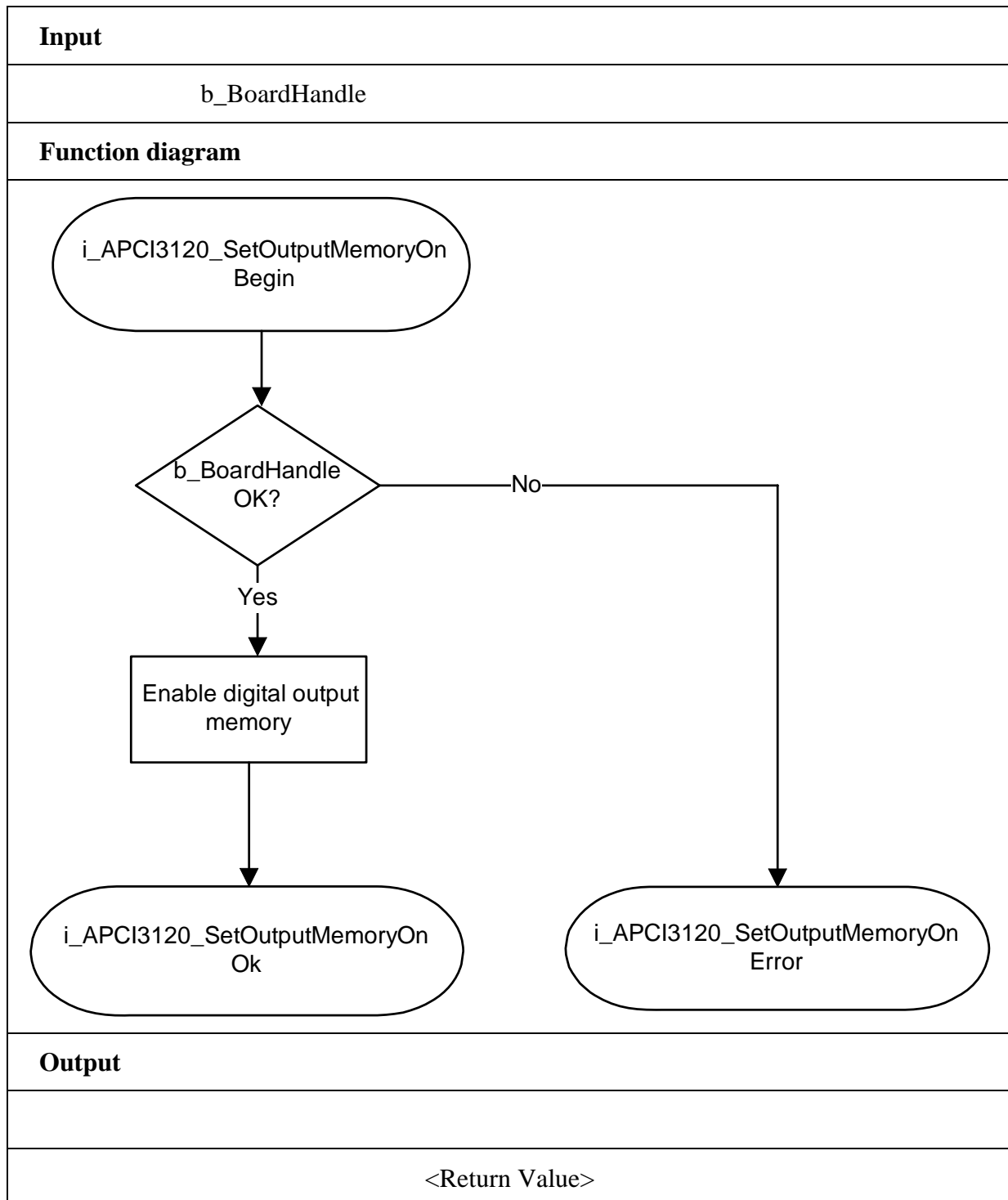
ANSI C:

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

```
i_ReturnValue = i_APCI3120_SetOutputMemoryOn (b_BoardHandle);
```

Return value:

0: No error.
-1: The handle parameter of the board is wrong.



6) i_APCI3120_SetOutputMemoryOff (...)**Syntax:**

<Return value> = i_APCI3120_SetOutputMemoryOff
(BYTE b_BoardHandle)

Parameters:**- Input:**

BYTE b_BoardHandle Handle of the board

- Output:

No output signal has occurred

Task:

Deactivates the digital output memory.

Calling convention:ANSI C:

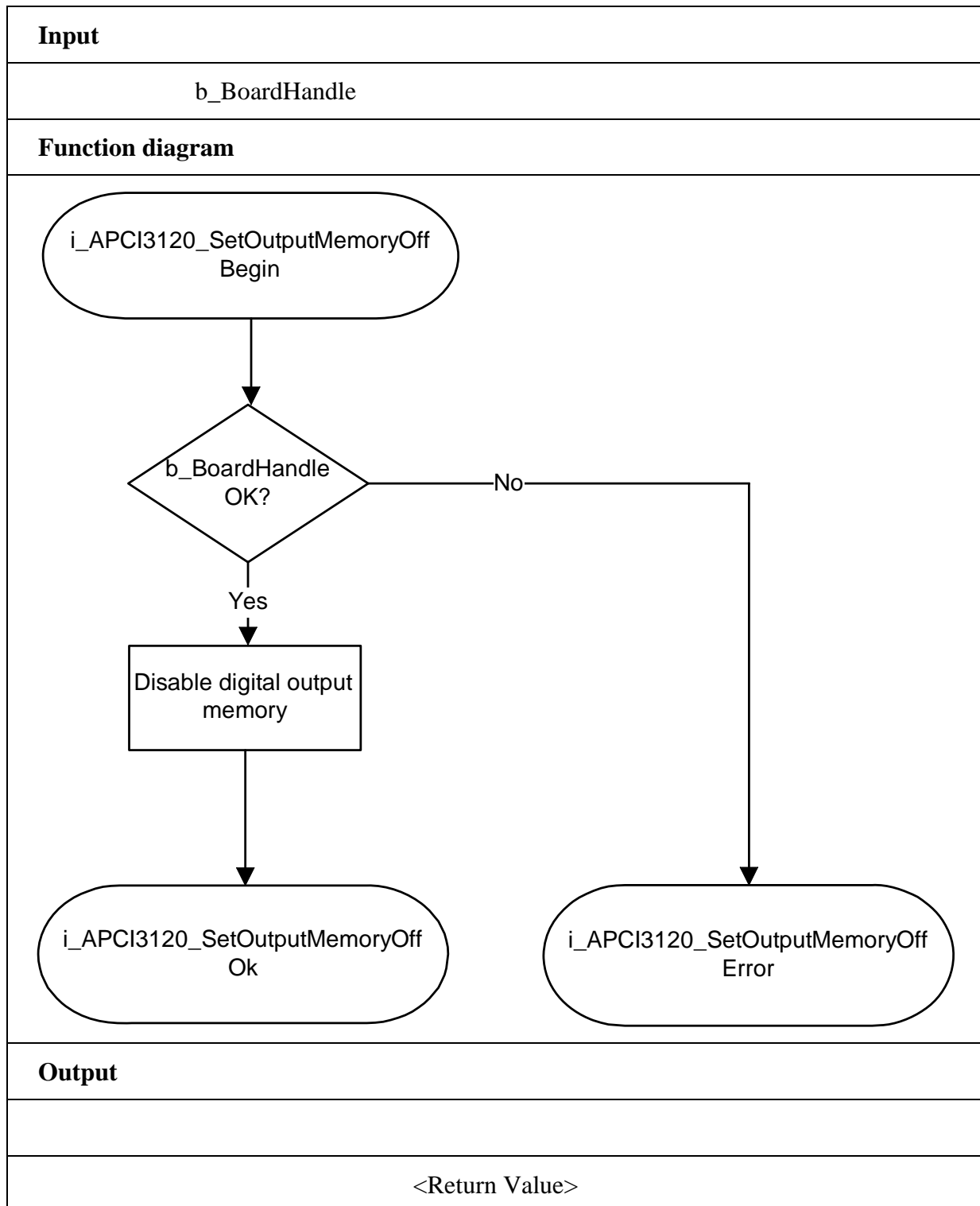
```
int                      i_ReturnValue;  
unsigned char    b_BoardHandle;
```

```
i_ReturnValue = i_APCI3120_SetOutputMemoryOff            (b_BoardHandle);
```

Return value:

0: No error.

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



3.9 Functions to be used in Kernel mode

1) i_APCI3120_KRNL_Read1DigitalInput (...)

Syntax:

```
<Return value> = i_APCI3120_KRNL_Read1DigitalInput
                    (UINT  ui_Address,
                     BYTE  b_Channel,
                     PBYTE pb_ChannelValue)
```

Parameters:

- Input:

UINT	ui_Address	Address of the board xPCI-3120
BYTE	b_Channel	The number of the input channel to be read (1 to 4) .

- Output:

PBYTE	pb_ChannelValue	Status of the digital input channel
		0 -> Low
		1 -> High

Task:

Indicates the status of an input channel. The variable b_Channel passes the input channel to be read (1 to 4). A value is returned with the variable pb_ChannelValue: 0 (Low) or 1 (High).

Calling convention:

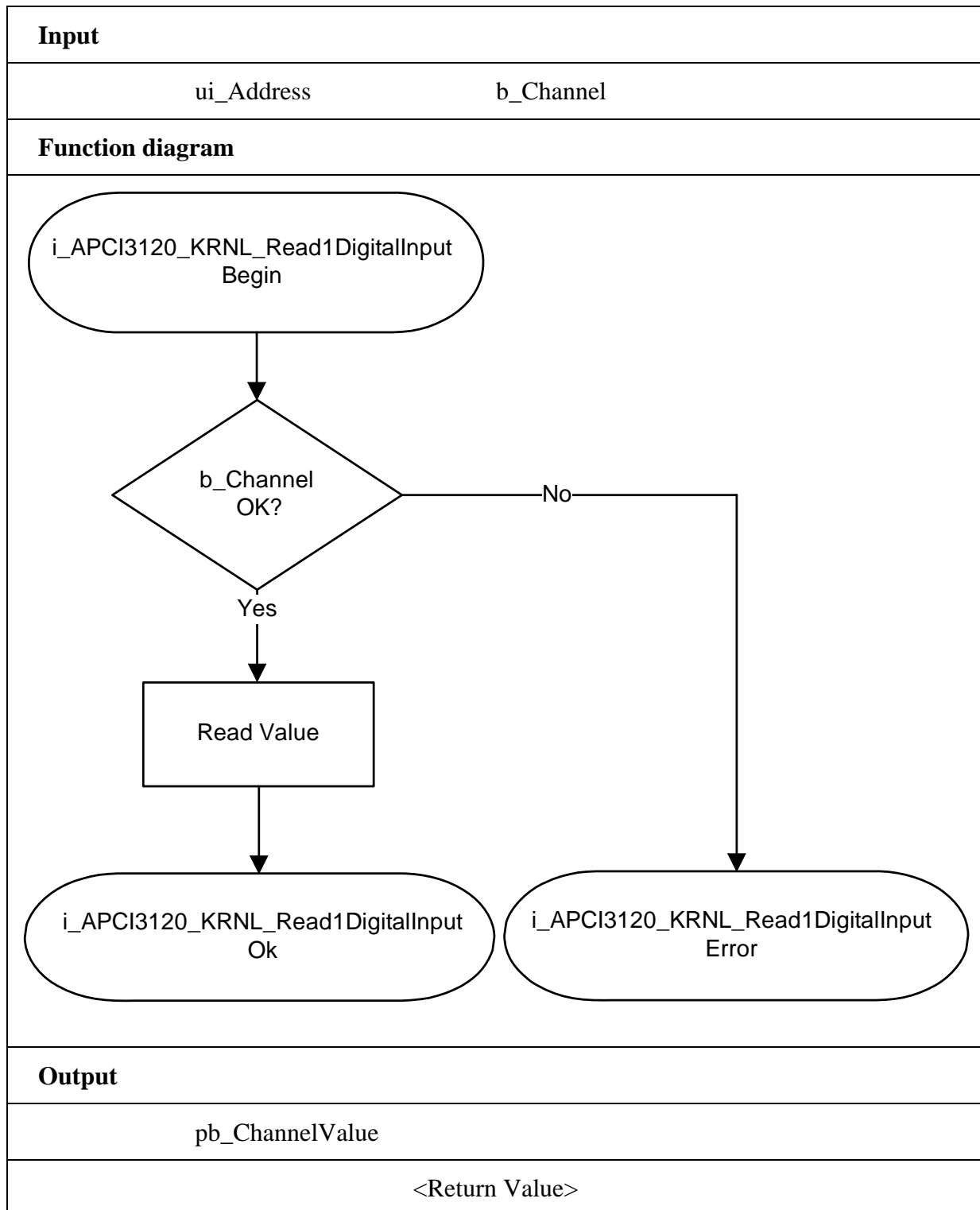
ANSI C :

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

```
i_ReturnValue = i_APCI3120_KRNL_Read1DigitalInput
                (0x390,
                 1,
                 &pb_ChannelValue);
```

Return value:

0: No error.
 -2: The input number is not between 1 and 4.



2) i_APCI3120_KRNL_Read4DigitalInput (...)**Syntax:**

```
<Return value> = i_APCI3120_KRNL_Read4DigitalInput
                    (UINT ui_Address,
                     PBYTE pb_PortValue)
```

Parameters:**- Input:**

UINT	ui_Address	Address of the board xPCI-3120
------	------------	---------------------------------------

- Output:

PBYTE	pb_PortValue	State of the digital input port (0 to 15)
-------	--------------	--

Task:

Indicates the state of the port. A value is returned with the variable pb_PortValue.

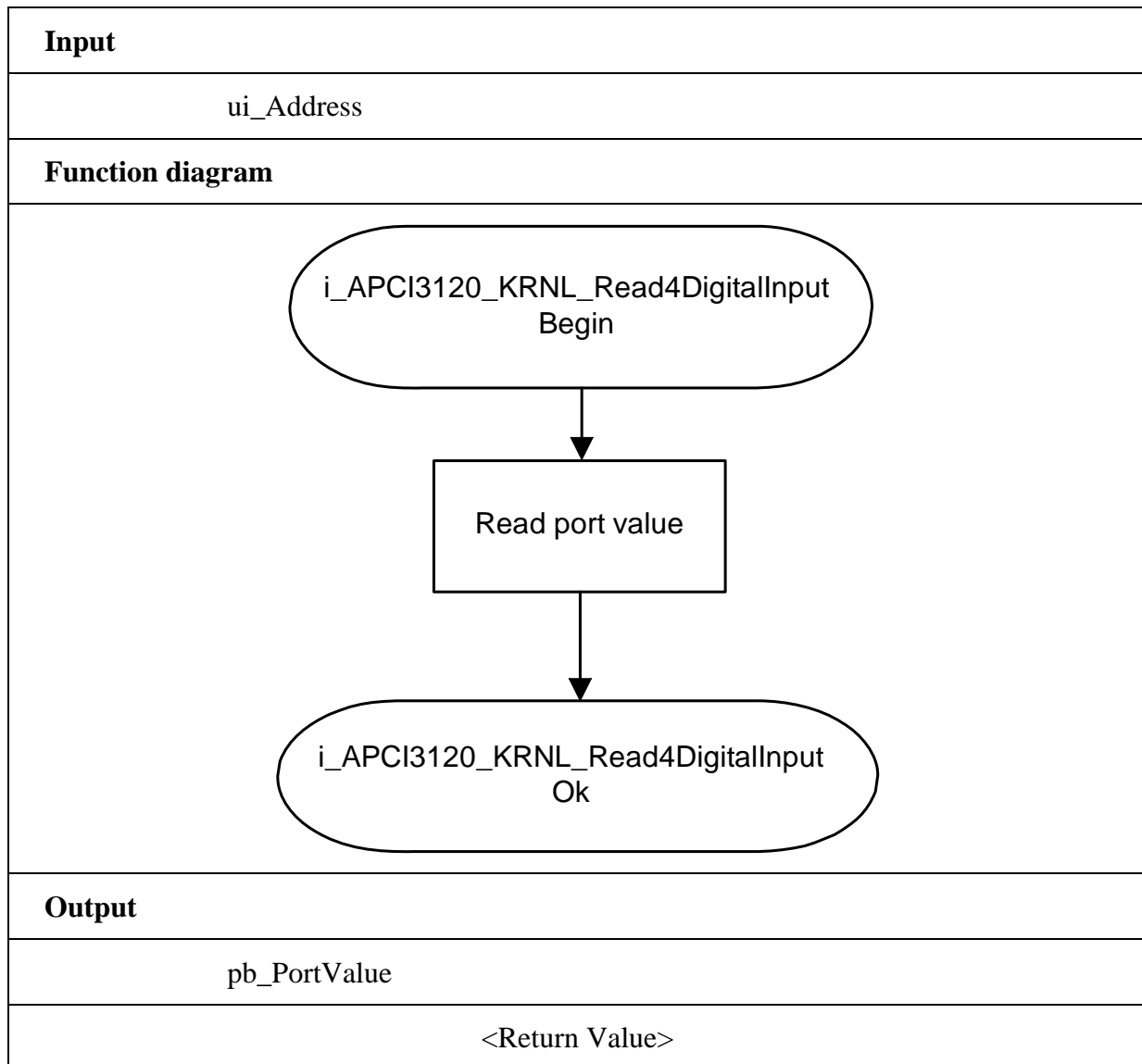
Calling convention:ANSI C:

```
int          i_ReturnValue;
unsigned char pb_PortValue;

i_ReturnValue = i_APCI3120_Read4DigitalInput
                (0x390,
                 &pb_PortValue);
```

Return value:

0: No error.



3) i_APCI3120_KRNL_Set1DigitalOutputOn (...)**Syntax:**

```
<Return value> = i_APCI3120_KRNL_Set1DigitalOutputOn
                    (UINT ui_Address,
                     BYTE  b_Channel)
```

Parameters:**- Input:**

UINT	ui_Address	Address of the board xPCI-3120
BYTE	b_Channel	Number of the output channel to be set (1 to 4).

- Output:

No output signal has occurred.

Task:

Sets the output channel which has been passed with the parameter b_Channel. Setting an output channel means setting an output channel to „High“.

Calling convention:ANSI C:

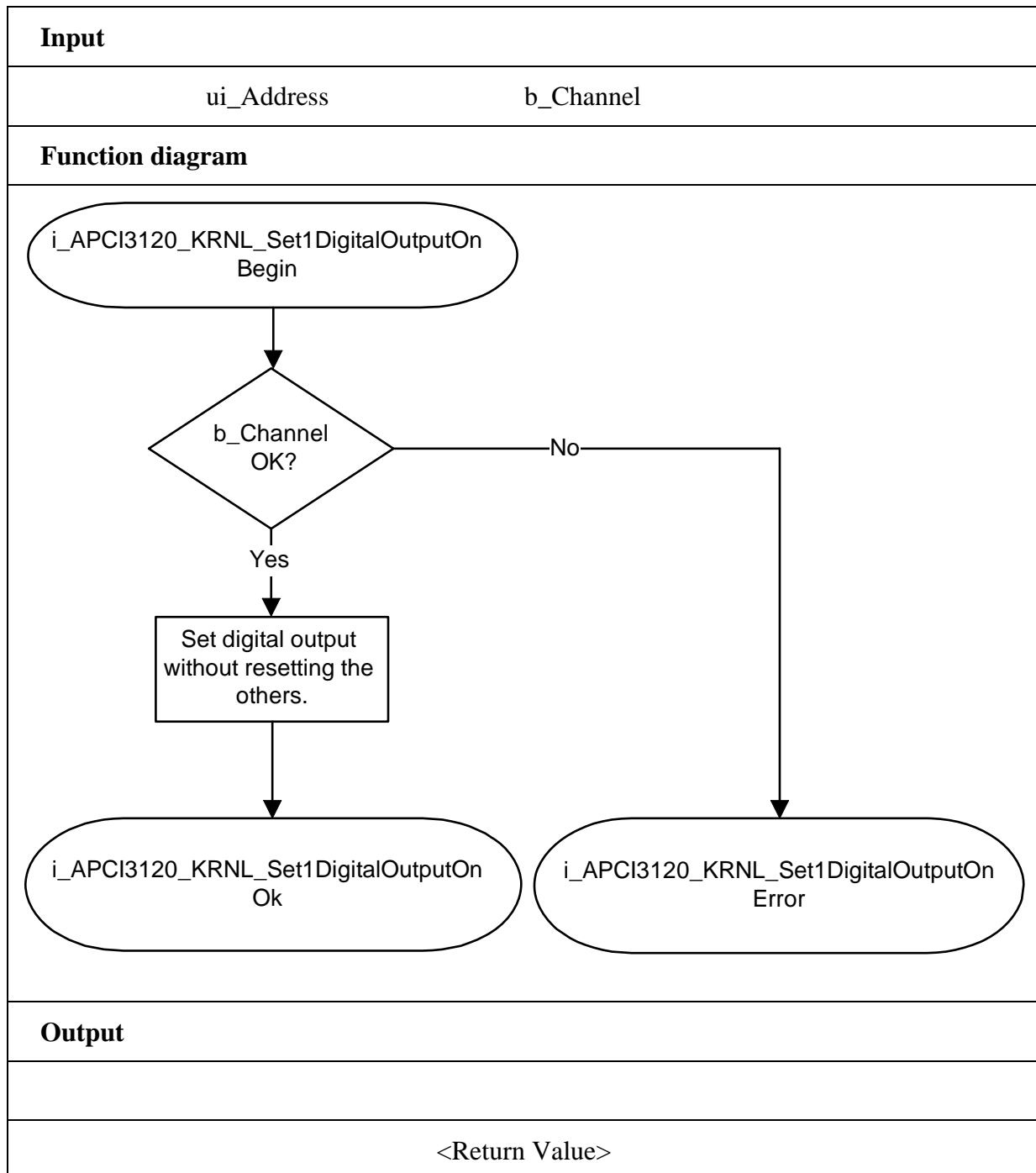
```
int      i_ReturnValue;
```

```
i_ReturnValue = i_APCI3120_KRNL_Set1DigitalOutputOn
                (0x390,
                 1);
```

Return value:

0: No error.

-2: The input number is not between 1 and 4.



4) i_APCI3120_KRNL_Set4DigitalOutputOn (...)**Syntax:**

```
<Return value> = i_APCI3120_KRNL_Set4DigitalOutputOn
                    (UINT  ui_Address,
                     BYTE  b_Value)
```

Parameters:**- Input:**

UINT	ui_Address	Address of the board xPCI-3120
BYTE	b_Value	Output value (0 to 15)

- Output:

No output signal has occurred.

Task:

Sets one or several output channels of a port. Setting an output channel means setting an output channel to „High“. The other output channels are set to „0“.

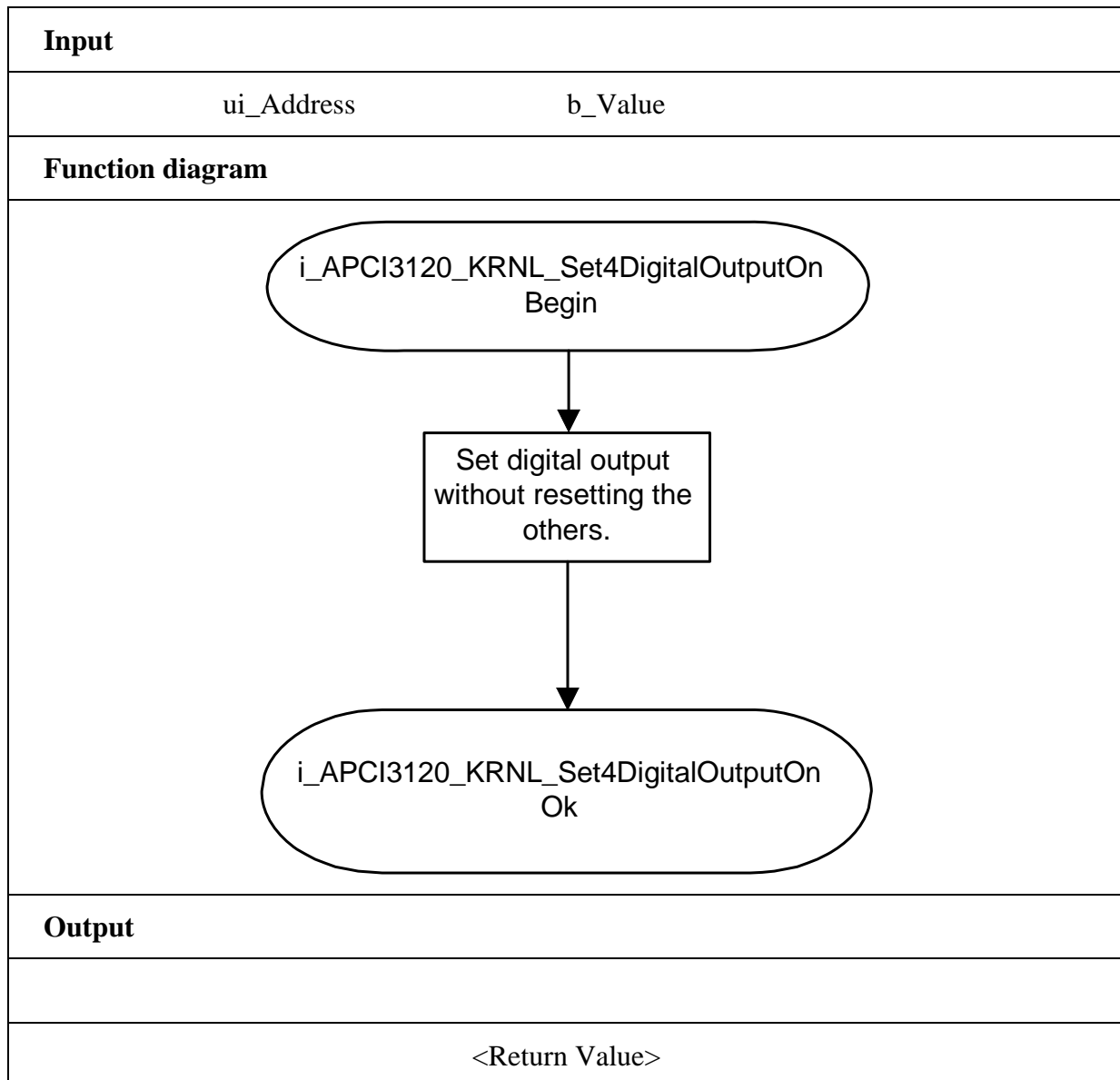
Calling convention:ANSI C:

```
int      i_ReturnValue;
```

```
i_ReturnValue = i_APCI3120_KRNL_Set4DigitalOutputOn    (0x390,
                                                         1);
```

Return value:

0: No error.



5) i_APCI3120_KRNL_Write1AnalogValue (...)**Syntax:**

```
<Return value> = i_APCI3120_KRNL_Write1AnalogValue
                    (UINT      ui_Address,
                     BYTE      b_ChannelNbr,
                     BYTE      b_Polarity,
                     UINT      ui_ValueToWrite)
```

Parameters:**- Input:**

UINT	ui_Address	Address of the board xPCI-3120
BYTE	b_ChannelNbr	Number of the analog output channel (0 to 7)
BYTE	b_Polarity	Selection of the output voltage range. See table 3-8.
UINT	ui_ValueToWrite	Analog output value to write in UNIPOLAR_MODE: 0 to 8192 in BIPOLAR_MODE: 0 to 16383

- Output:

No output signal has occurred.

Task:

Writes an analog value (*ui_ValueToWrite*) on the analog output channel *b_ChannelNbr*.

Calling convention:ANSI C:

```
int      i_ReturnValue;
```

```
i_ReturnValue = i_APCI3120_KRNL_Write1AnalogValue
                (0x390,
                 1,
                 APCI3120_UNIPOLAR);
```

Return value:

0: No error

-3: The output voltage range selected is wrong. See table 3-8

-4: Output value too high

