DIN EN ISO 9001:2000
certified

ADDI-DATA®

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

Technical description

ADDIALOG APCI-/CPCI-3001

Standard software

Edition: 04.01-07/2005

Tables

1 INTRODUCTION

IMPORTANT!
Note the following conventions in the text:

Function: "i_PCI3001_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

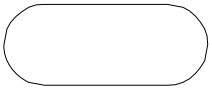
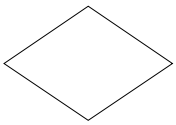
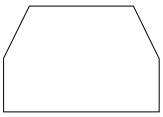
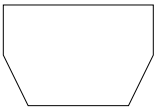
Define name	Decimal value	Hexadecimal value
DLL_COMPILER_C	0	0
DLL_COMPILER_PASCAL	1	1
DLL_COMPILER_VB	2	2
DLL_LABVIEW	3	3
DLL_COMPILER_VB_5	4	4
PCI3001_DISABLE	0	0
PCI3001_ENABLE	1	1
PCI3001_CHANNEL0	0	0
PCI3001_CHANNEL1	1	1
PCI3001_CHANNEL2	2	2
PCI3001_CHANNEL3	3	3
PCI3001_CHANNEL4	4	4
PCI3001_CHANNEL5	5	5
PCI3001_CHANNEL6	6	6
PCI3001_CHANNEL7	7	7
PCI3001_CHANNEL8	8	8
PCI3001_CHANNEL9	9	9
PCI3001_CHANNEL10	10	A
PCI3001_CHANNEL11	11	B
PCI3001_CHANNEL12	12	C

Define name	Decimal value	Hexadecimal value
PCI3001_CHANNEL13	13	D
PCI3001_CHANNEL14	14	E
PCI3001_CHANNEL15	15	F
PCI3001_1_GAIN	0	0
PCI3001_2_GAIN	16	10
PCI3001_5_GAIN	32	20
PCI3001_10_GAIN	48	30
PCI3001_UNIPOLAR	128	80
PCI3001_BIPOLAR	0	0
PCI3001_SIMPLE_MODUS	0	0
PCI3001_DELAY_MODUS	1	1
PCI3001_DELAY_1_MODUS	2	2
PCI3001_DMA_NOT_USED	0	0
PCI3001_DMA_USED	1	1
PCI3001_SINGLE	0	0
PCI3001_CONTINUOUS	1	1
PCI3001_ASYNCHRONOUS_MODE	0	0
PCI3001_SYNCHRONOUS_MODE	1	1

2 DIN 66001- GRAPHICAL SYMBOLS

All the API software functions necessary to the operation of the board **APCI-/CPCI-3001** are listed in the following chapter (Device dirver).

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)
	**Terminator** (eg. Beginning or end of a sequence, origin or place of data)
	**Decision** **Selection unit** (eg.: switch)
	**Loop limit** Beginning
	**Loop limit** End

Remark: The **CPCI-3001** board is compatible with the **APCI-3001** 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-3001.**

3 SOFTWARE FUNCTIONS (API)

3.1 Initialisation

1) i_PCI3001_InitCompiler (..)

Syntax:
<Return value> = i_PCI3001_InitCompiler (BYTE b_CompilerDefine)

Parameter:
- 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 NT or Windows 95. - 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.

IMPORTANT!
This function is only available with a Windows environment.

Calling convention:
ANSI C :

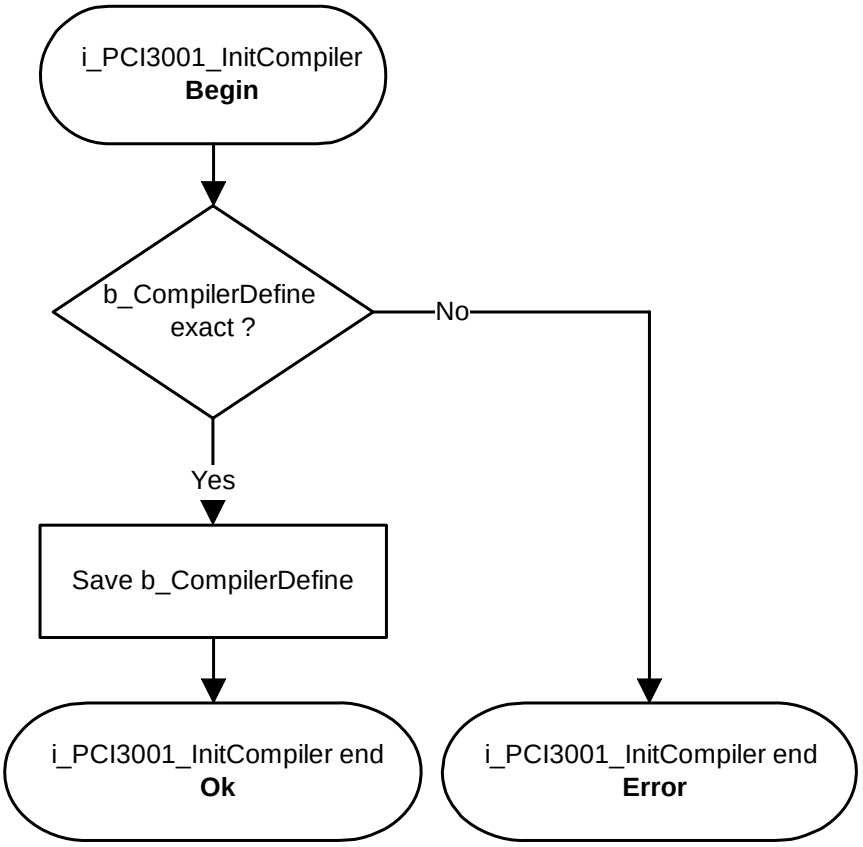
```
int        i_ReturnValue;

i_ReturnValue = i_PCI3001_InitCompiler (DLL_COMPILER_C);
```

Return value:
0: No error
-1: Compiler parameter is wrong

Input

b_CompilerDefine

Function diagram



Output

<Return Value>

2) i_PCI3001_CheckAndGetPCISlotNumber (...)

Syntax:
<Return value> = i_PCI3001_CheckAndGetPCISlotNumber
(PBYTE pb_SlotNumberArray)

Parameter:
- Input:
No input signal has occurred.
- Output:

PBYTE	pb_SlotNumberArray	Slot number list. Pointer of one byte array .

Task:
Checks all **xPCI-3001** and returns the slot number of each **xPCI-3001** board.
Each *pb_SlotNumberArray* contains the slot number (1 to 10) of 1 **xPCI-3001**.

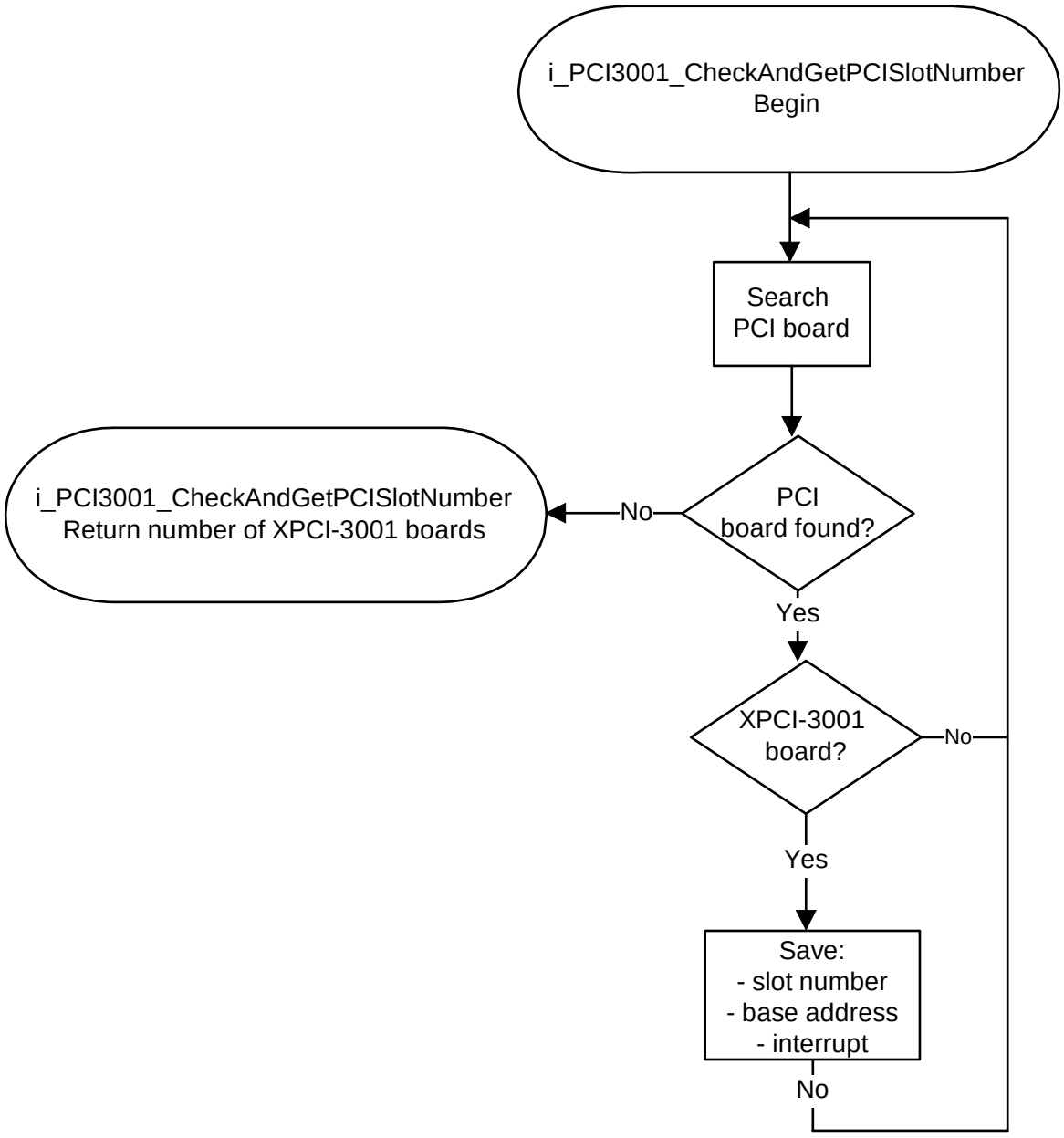
Calling convention:
ANSI C :

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

i_ReturnValue = i_PCI3001_CheckAndGetPCISlotNumber
                                        (b_SlotNumberArray);
```

Return value:
Returns the number of **xPCI-3001** found in your PC.
If the return values equal 0, no **xPCI-3001** was found in your PC.

Input

Function diagram



Output

pb_SlotNumberArray

<Return Value>

3) i_PCI3001_SetBoardInformation (...)

Syntax:
<Return value> = i_PCI3001_SetBoardInformation
(BYTE b_SlotNumber,
BYTE b_AnalogInputChannelNbr,
PBYTE pb_BoardHandle)

Parameter:
- Input:

BYTE	b_SlotNumber	Slot number of the board
BYTE	b_AnalogInputChannelNbr	Number of analog inputs (0 to 16)

- Output:

PBYTE	pb_BoardHandle	Handle[1] of the **xPCI-3001** to use the functions

Task:
Verifies if board **xPCI-3001** is present. Stores the following information:
- slot number,
- the number of analog inputs

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_PCI3001_SetBoardInformation (1, 16, &b_BoardHandle);
```

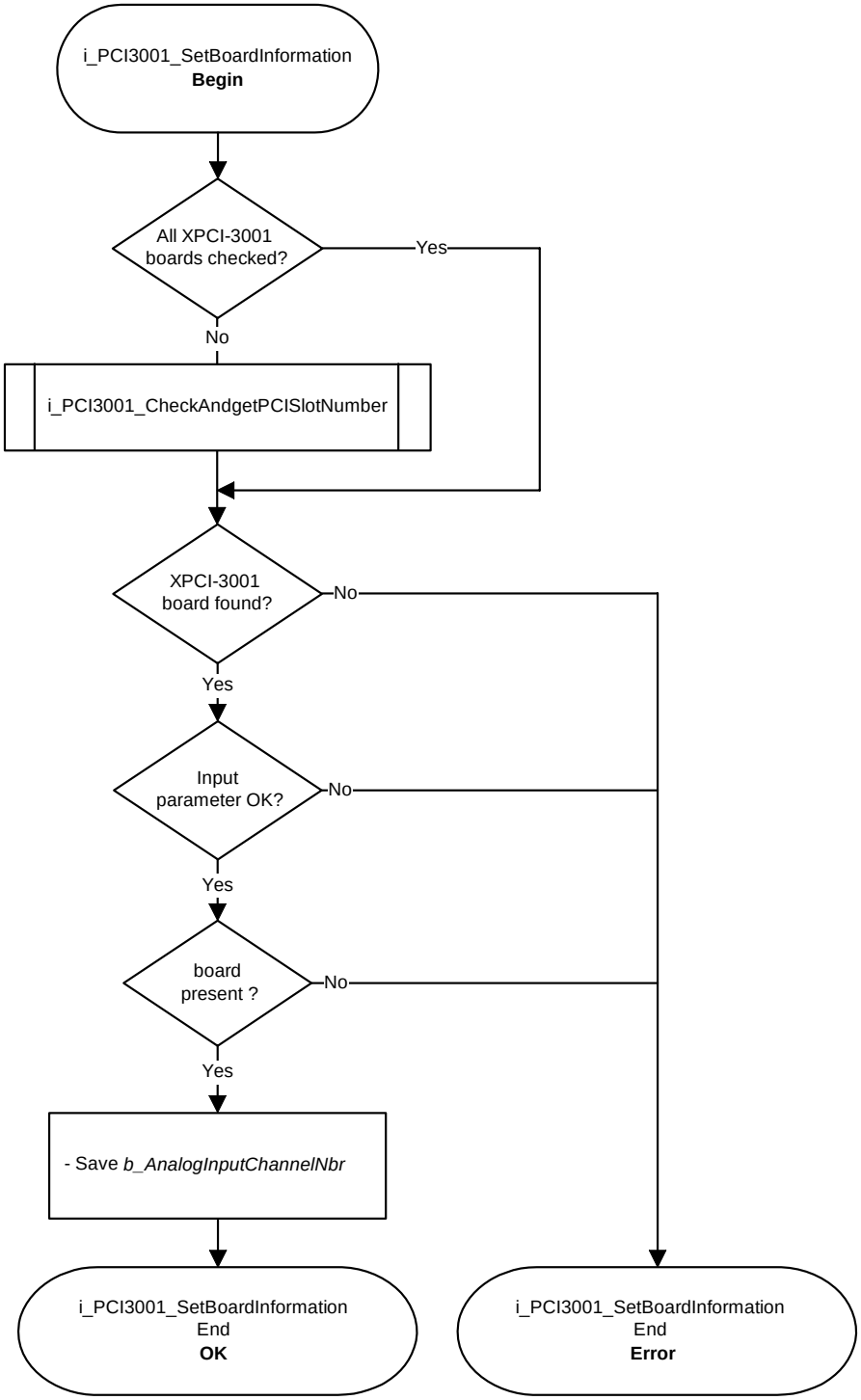
Return value:
0: No error
-1: Slot number not available
-2: Board not present
-3: Number for the analog inputs is wrong
-4: No handle is available for the board (up to 10 handles can be used)

[1] Identification number of the board

Input

b_SlotNumber b_AnalogInputChannelNbr

Function diagram



Output

pb_BoardHandle

<Return Value>

4) i_PCI3001_GetHardwareInformation (...)

Syntax:

```
<Return value> =  i_PCI3001_GetHardwareInformation
                                                (BYTE    b_BoardHandle,
                                                 PUINT   pui_BaseAddress,
                                                 PBYTE   pb_InterruptNbr,
                                                 PBYTE   pb_SlotNumber)
```

Parameter:

- Input:

BYTE	b_BoardHandle	Handle of the board

- Output:

PUINT	pui_BaseAddress	**xPCI-3001** base address
PBYTE	pb_InterruptNbr	**xPCI-3001** interrupt channel.
PBYTE	pb_SlotNumber	**xPCI-3001** slot number

Task:

Returns the base address, the interrupt and slot number of the **xPCI-3001**.

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_PCI3001_GetHardwareInformation    (b_BoardHandle,
                                                     &ui_BaseAddress,
                                                     &b_InterruptNbr,
                                                     &b_SlotNumber);
```

Return value:

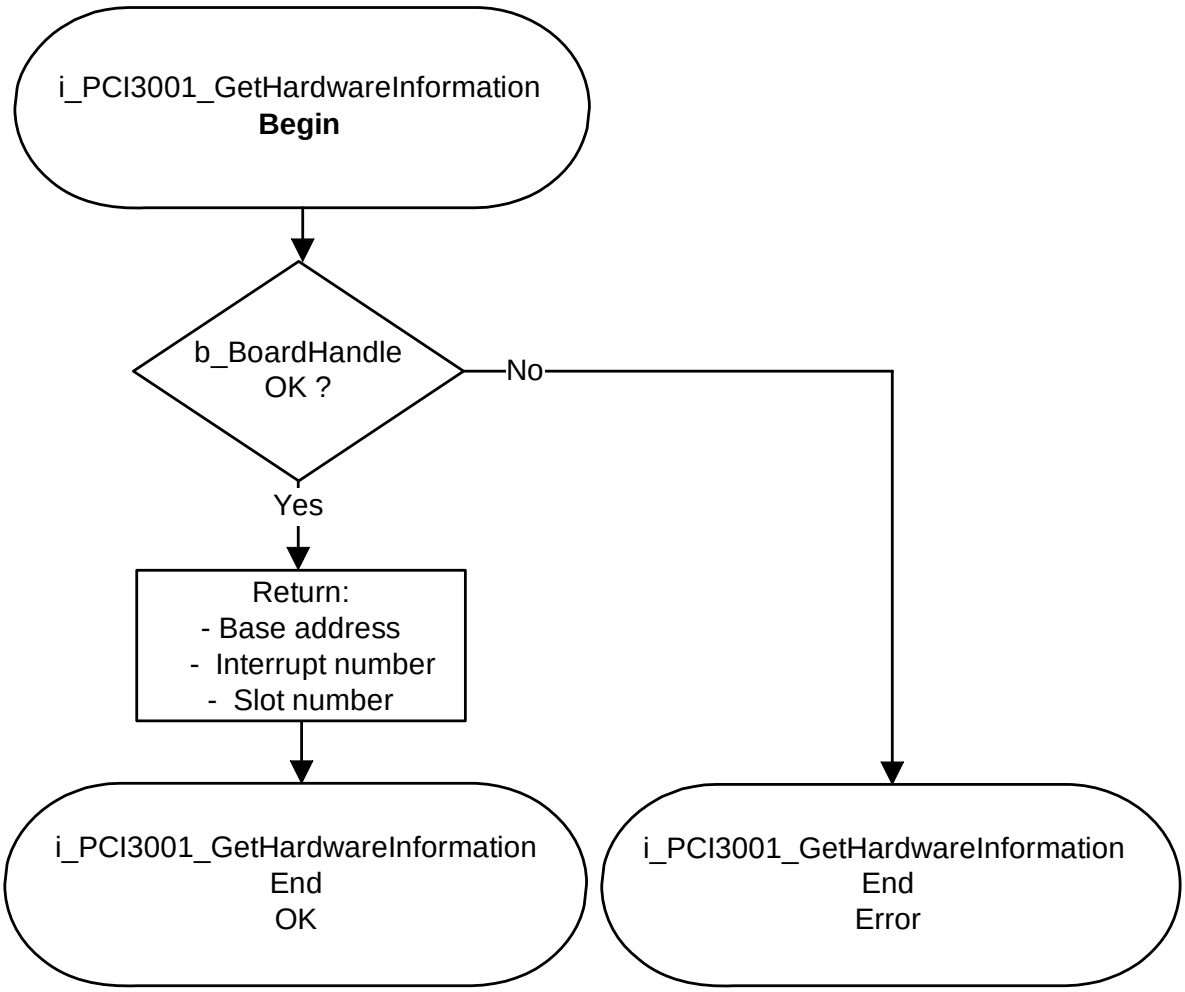
0: No error

-1: The handle parameter of the board is wrong

Input

b_BoardHandle

Function diagram



Output

pui_BaseAddress pb_InterruptNbr
pb_SlotNumber

<Return Value>

5) i_PCI3001_CloseBoardHandle (..)

IMPORTANT!
Call up this function each time you want to leave the user program!

Syntax:
<Return value> = i_PCI3001_CloseBoardHandle (BYTE b_BoardHandle)

Parameter:
- 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 :

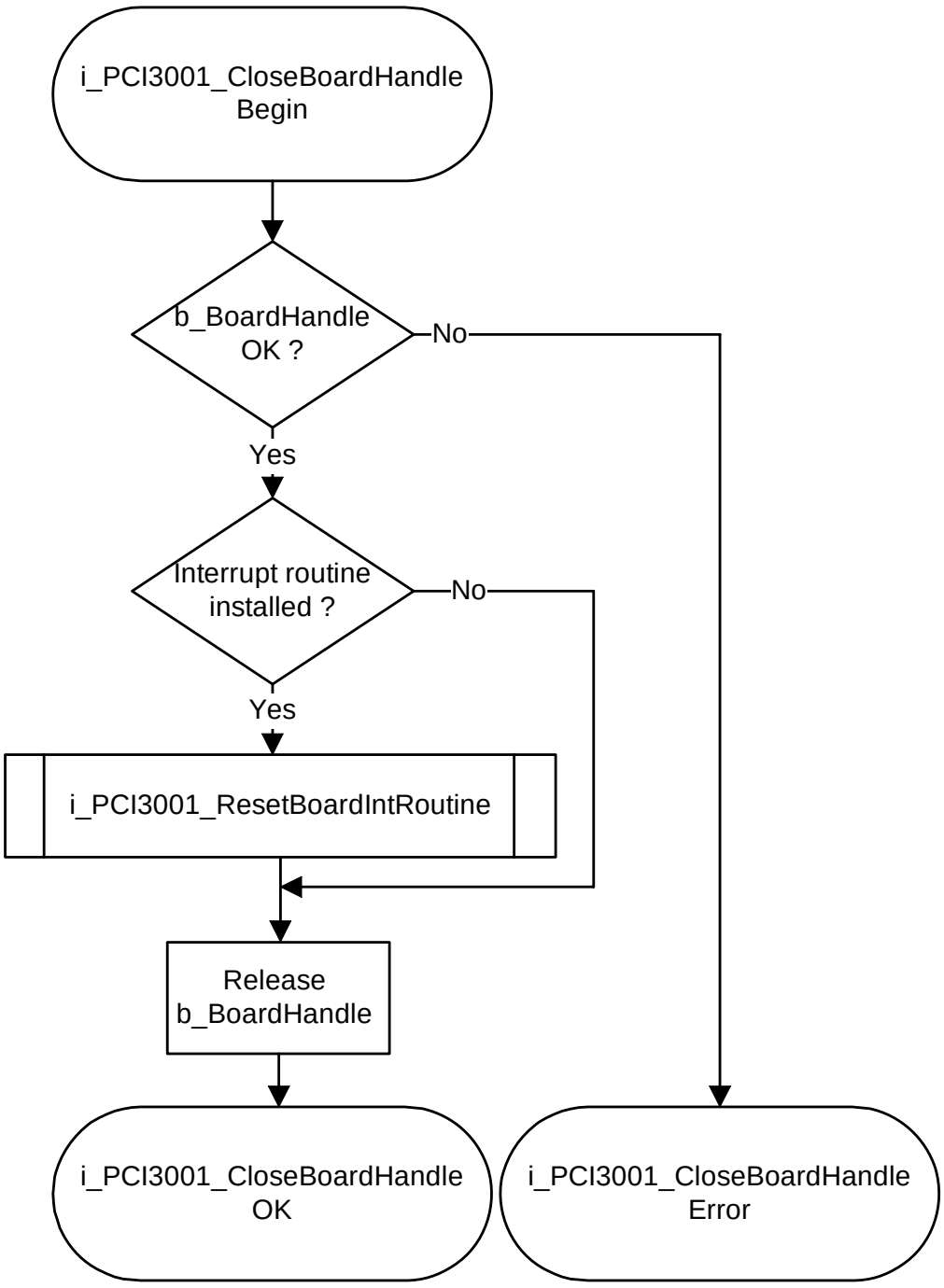
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PCI3001_CloseBoardHandle      (b_BoardHandle);
```

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

Input

b_BoardHandle

Function diagram



Output

<Return Value>

3.2 Interrupt

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

1) i_PCI3001_SetBoardIntRoutineDos (..)

Syntax:
<Return value> = i_PCI3001_SetBoardIntRoutineDos
(BYTE b_BoardHandle,
VOID v_FunctionName (BYTE b_BoardHandle,
BYTE b_InterruptMask,
PUINT pui_AnalogInputValue))

Parameter:
- 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 for each **xPCI-3001** on which you want to enable an interrupt action. It installs one user interrupt function for all boards on which you have enabled the interrupt.

First calling (first board):
- the user interrupt routine is installed
- interrupts are enabled.

If you operate several boards **xPCI-3001** which have to react to interrupts, call up the function as often as you operate boards **xPCI-3001**. The variable *v_FunctionName* is only relevant **for the first calling**.

From the second callup 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*)

v_FunctionName	Name of the user interrupt routine
b_BoardHandle	Handle of the **xPCI-3001** which has generated the interrupt
b_InterruptMask	Mask of the events which have generated the interrupt
pui_AnalogInputValue	The values of the analog inputs and of the DMA buffer are returned.

Table 3-1: Values returned for the analog inputs

Event	Variable name	Variable content
b_InterruptMask = 1	*pui_AnalogInputValue [0]*	Number of the last analog input
	pui_AnalogInputValue [1]	Value of the analog input
b_InterruptMask = 2	*pui_AnalogInputValue [0]*	Number of analog inputs
	pui_AnalogInputValue [1] up to *pui_AnalogInputValue [pui_AnalogInputValue[0]]* See example 1	Values of the analog inputs
b_InterruptMask = 4	*pui_AnalogInputValue [0]*	Number of analog inputs
	pui_AnalogInputValue [1] up to *pui_AnalogInputValue [pui_AnalogInputValue[0]]* See example 1	Values of the analog inputs
b_InterruptMask = 8	*pui_AnalogInputValue*	Values of the DMA buffer

Example 1 pui_AnalogInputValue [0] = 3

3	= Number of analog inputs
1	= Analog value 0
2	= Analog value 1
3	= Analog value 2

Table 3-2 : Interrupt mask

Mask	Meaning
0000 000**1**	End of Conversion (EOC)
0000 00**1**0	Conversion driven by timer is completed
0000 0**1**00	Conversion of a group of channels is completed (EOS)
0000 **1**000	DMA conversion cycle is completed
000**1** 0000	Timer 2 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_PCI3001_SetBoardIntRoutineDos  (b_BoardHandle,
                                                  v_FunctionName);
```

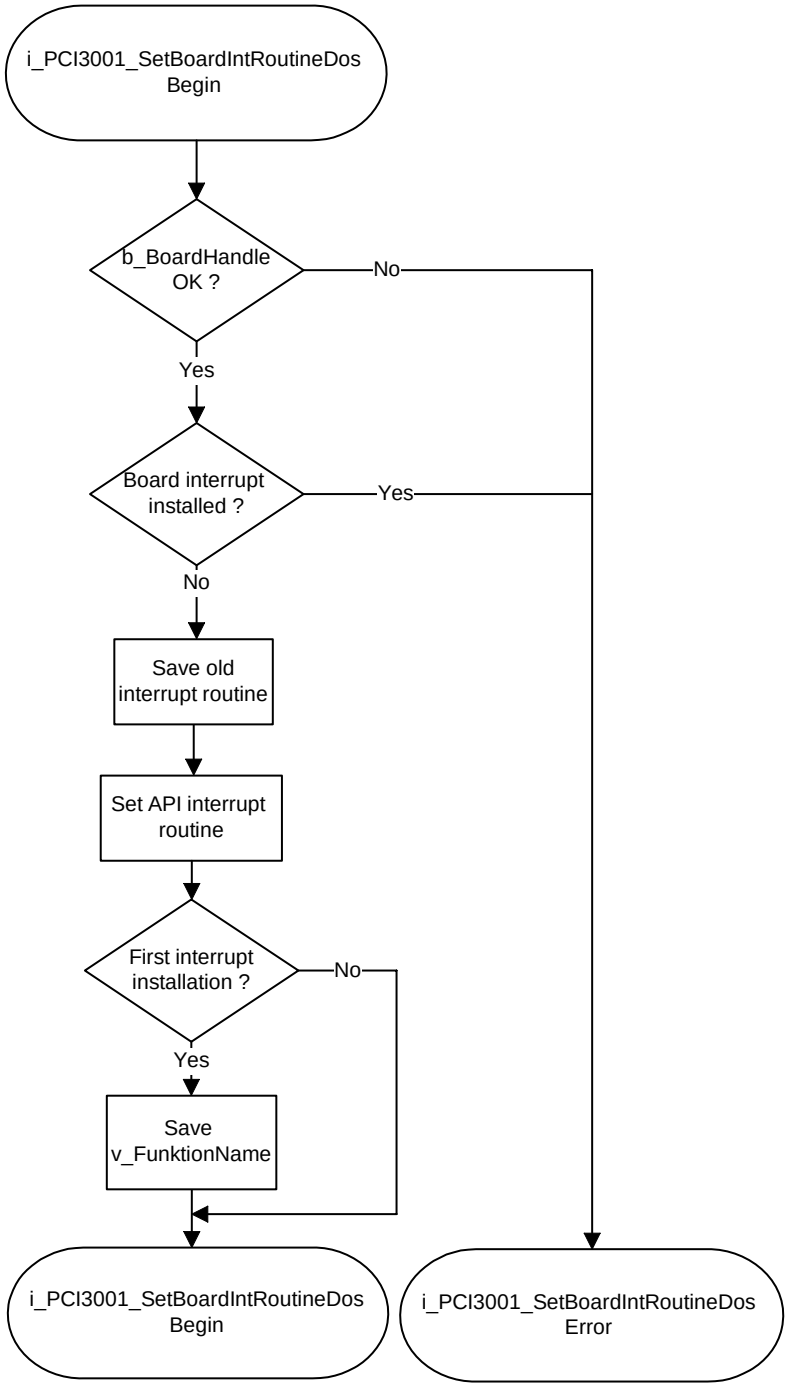
Return value:

0: No error
-1: The handle parameter of the board is wrong
-2: Interrupt already installed

Input

b_BoardHandle v_FunctionName

Function diagram



Output

<Return Value>

IMPORTANT!
This function is only available for Visual Basic DOS.

2) i_PCI3001_SetBoardIntRoutineVBDos (..)

Syntax:
<Return value> = i_PCI3001_SetBoardIntRoutineVBDos
(BYTE b_BoardHandle)

Parameter:
- Input:
BYTE b_BoardHandle Handle of the board
- Output:
No output signal has occurred.

Task:
This function must be up called for each **xPCI-3001** on which you want to enable an interrupt action. If an interrupt occurs, a Visual Basic event is generated. Plese refer to the calling convention.

From the first callup of the function:
- interrupts are enabled for the selected board.

If you operate several boards **xPCI-3001** which have to react to interrupts, call up the function as often as you operate boards **xPCI-3001**.

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_PCI3001_TestInterrupt"

This function tests the interrupt of board **xPCI-3001**. It is used to obtain the values of *b_BoardHandle , b_InterruptMask, 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_PCI3001_TestInterrupt    (i_BoardHandle, _
                                            i_InterruptMask,  _
                                            l_AnalogInputValue (0))
.
.
.
Return

ON UEVENT GOSUB IntLabel
UEVENT ON
i_ReturnValue = i_PCI3001_SetBoardIntRoutineVBDos     (b_BoardHandle)
```

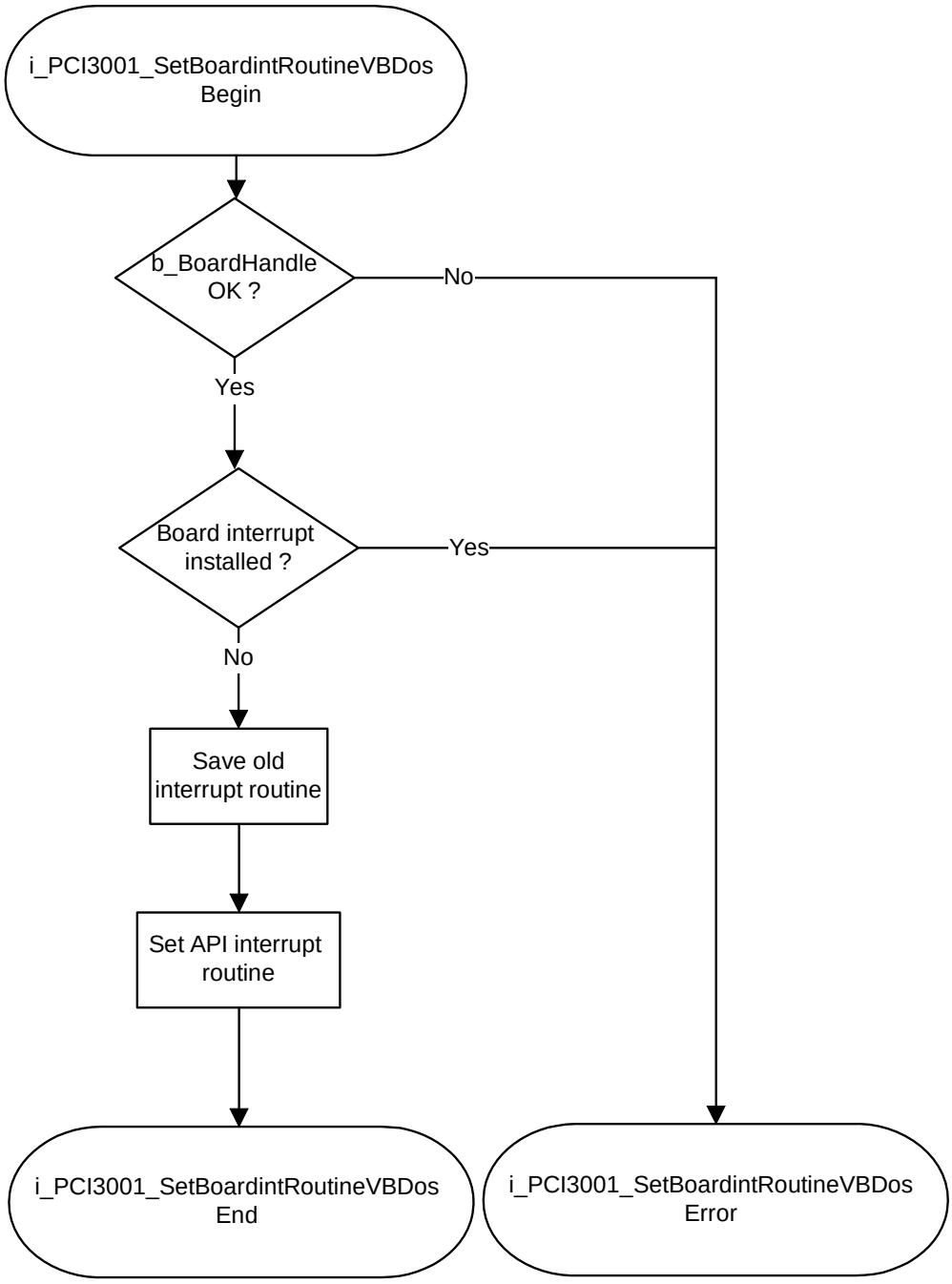
Return value:

0: No error
-1: The handle parameter of the board is wrong
-2: Interrupt already installed

Input

b_BoardHandle

Function diagram



Output

<Return Value>

IMPORTANT!
This function is only available for Windows 3.1 and Windows 3.11.

3) i_PCI3001_SetBoardIntRoutineWin16 (..)

Syntax:
<Return value> = i_PCI3001_SetBoardIntRoutineWin16
(BYTE b_BoardHandle,
VOID v_FunctionName (BYTE b_BoardHandle,
BYTE b_InterruptMask,
PUINT pui_AnalogInputValue))

Parameter:
- 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-3001** on which you want to enable an interrupt action. It installs one user interrupt function for all boards on which you have enabled the interrupt.

First callup (first board):
- the user interrupt routine is installed
- interrupts are enabled.

If you operate several boards **xPCI-3001** which have to react to interrupts, call up the function as often as you operate boards **xPCI-3001**. The variable *v_FunctionName* is only relevant **for the first calling**.

From the second callup 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*)

v_FunctionName	Name of the user interrupt routine
b_BoardHandle	Handle of the **xPCI-3001** which has generated the interrupt
b_InterruptMask	Mask of the events which have generated the interrupt
pui_AnalogInputValue	The values of the analog inputs and of the DMA buffer are returned.

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

IMPORTANT!
If you use Visual Basic for Windows, the following parameter has no meaning. Please use the "i_PCI3001_TestInterrupt" function.

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_PCI3001_SetBoardIntRoutineWin16 (b_BoardHandle,
                                                   v_FunctionName);
```

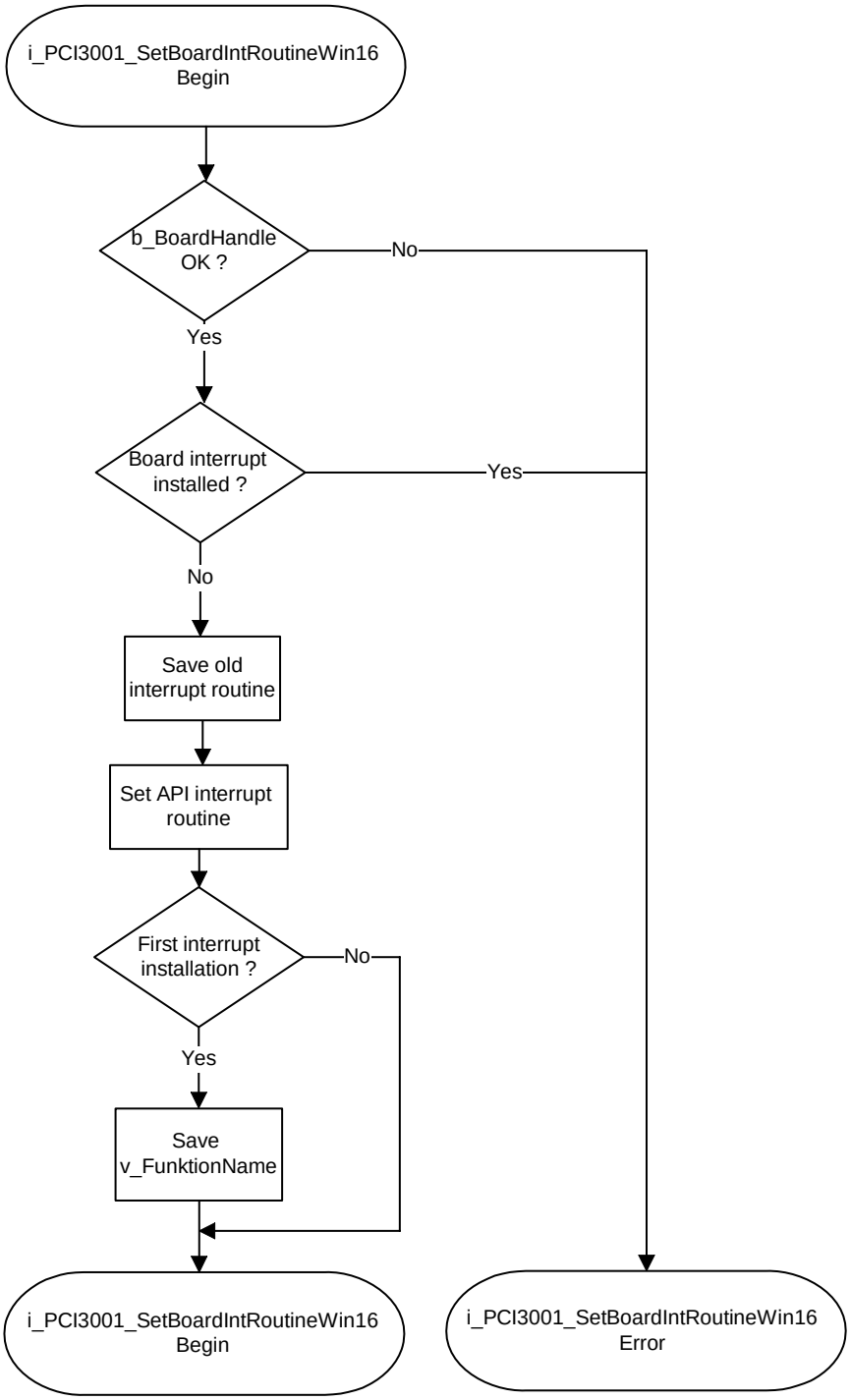
Return value:

0: No error
-1: The handle parameter of the board is wrong
-2: Interrupt already installed

Input

b_BoardHandle v_FonctionName

Function diagram



Output

<Return Value>

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

4) i_PCI3001_SetBoardIntRoutineWin32 (..)

Syntax:

```
<Return value> = i_PCI3001_SetBoardIntRoutineWin32
                 (BYTE       b_BoardHandle,
                  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))
```

Parameter:

- Input:

BYTE	b_BoardHandle	Handle of the board
BYTE	b_UserCallingMode	PCI3001_SYNCHRONOUS_MODE : The user routine is directly called by the driver interrupt routine. PCI3001_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 values allocated by the dynamic array (only used in acquisition with DMA). Other compilers: Determines the size in bytes of the user shared memory. Only used if you have selected the PCI3001_SYNCHRONOUS_MODE mode.

IMPORTANT!
The size of the User Shared Memory is limited to 63 MB. It could cause problems if more memory is required.

- Output:

VOID **	ppv_UserSharedMemory	With Visual Basic 5.0: pointer on a dynamic array. (Only used in acquisition with DMA. See Function i_PCI3001_InitAnalogInputAcquisition) Other compilers: User shared memory address Only used if you have selected the PCI3001_SYNCHRONOUS_MODE mode.

Task:
If you use Visual Basic 5.0:
- you can only use the asynchronous mode.
- the parameters ppv_UserSharedMemory and ul_UserSharedMemorySize are used in acquisition with DMA: when you want to start an analog acquisition with DMA you have to pass a pointer on a dynamic array and set its size. This pointer is used to store the analog value. The size corresponds to the number of acquisitions you want to make before an interrupt occurs. If you do not use the DMA mode, enter 0 for both parameters. Important : The size allocated to the dynamic array must at least equal the number of acquisitions.

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.

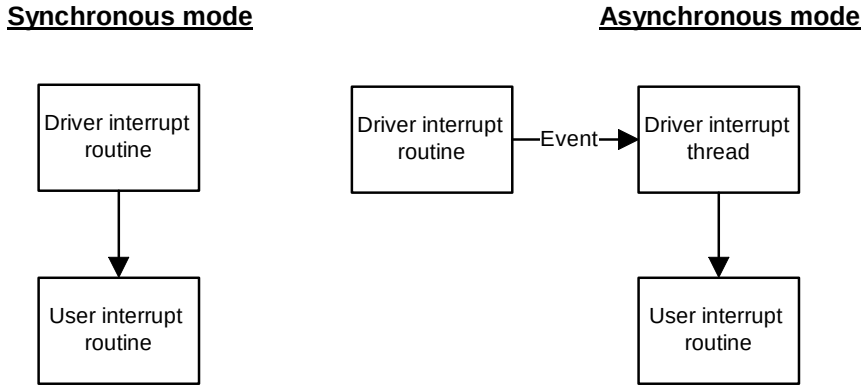
This function must be called up for each **xPCI-3001** on which an interrupt is to be enabled. It installs one user interrupt function for all boards on which you have enabled the interrupt.

First calling (first board):
- the user interrupt routine is installed
- interrupts are enabled
- user shared memory is allocated if PCI3001_SYNCHRONOUS_MODE has been selected.

If you operate several boards **xPCI-3001** which have to react to interrupts, call up the function as often as you operate on boards **xPCI-3001**. 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 up as follows:
- directly by 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
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 up Windows API functions.
	The user routine cannot call up functions which give access to global variables. The user can still use a shared memory.
	The user routine can only call up xPCI-3001 driver functions with the following extension: "i_PCI3001_KRNL_XXXX"
	This mode is not available for Visual Basic

	ASYNCHRONOUS MODE
ADVANTAGE	The user can debug the user interrupt routine provided it is not programmed in Visual Basic 5
	The user routine can call up Windows API functions.
	The user routine can call up functions which give access to global variables.
	The user routine can call up all xPCI-3001 driver functions with the following extension: "i_PCI3001_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 PCI3001_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    pui_AnalogInputValue,
                         BYTE     b_UserCallingMode,
                         VOID *   pv_UserSharedMemory)
```

v_FunctionName	Name of the user interrupt routine
b_BoardHandle	Handle of the **xPCI-3001** which has generated the interrupt
b_InterruptMask	Mask of the events which have generated the interrupt
pui_AnalogInputValue	The values of the analog inputs and of the DMA buffer are returned.
b_UserCallingMode	PCI3001_SYNCHRONOUS_MODE: User routine is directly called by driver interrupt routine. PCI3001_ASYNCHRONOUS_MODE: User routine is called by driver interrupt thread
pv_UserSharedMemory	Pointer of the user shared memory.

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

IMPORTANT!
If you use Visual Basic 4 the following parameters have no signification. Please use the "i_PCI3001_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_PCI3001_SetBoardIntRoutineWin32
                                    (b_BoardHandle,
                                     PCI3001_SYNCHRONOUS_MODE,
                                     sizeof (str_UserStruct),
                                     (void **) &ps_UserSharedMemory,
                                     v_FunctionName);
```

IMPORTANT!
Please read the following if you use Visual Basic 5.0.

Visual Basic 5: Sample Acquisition with DMA

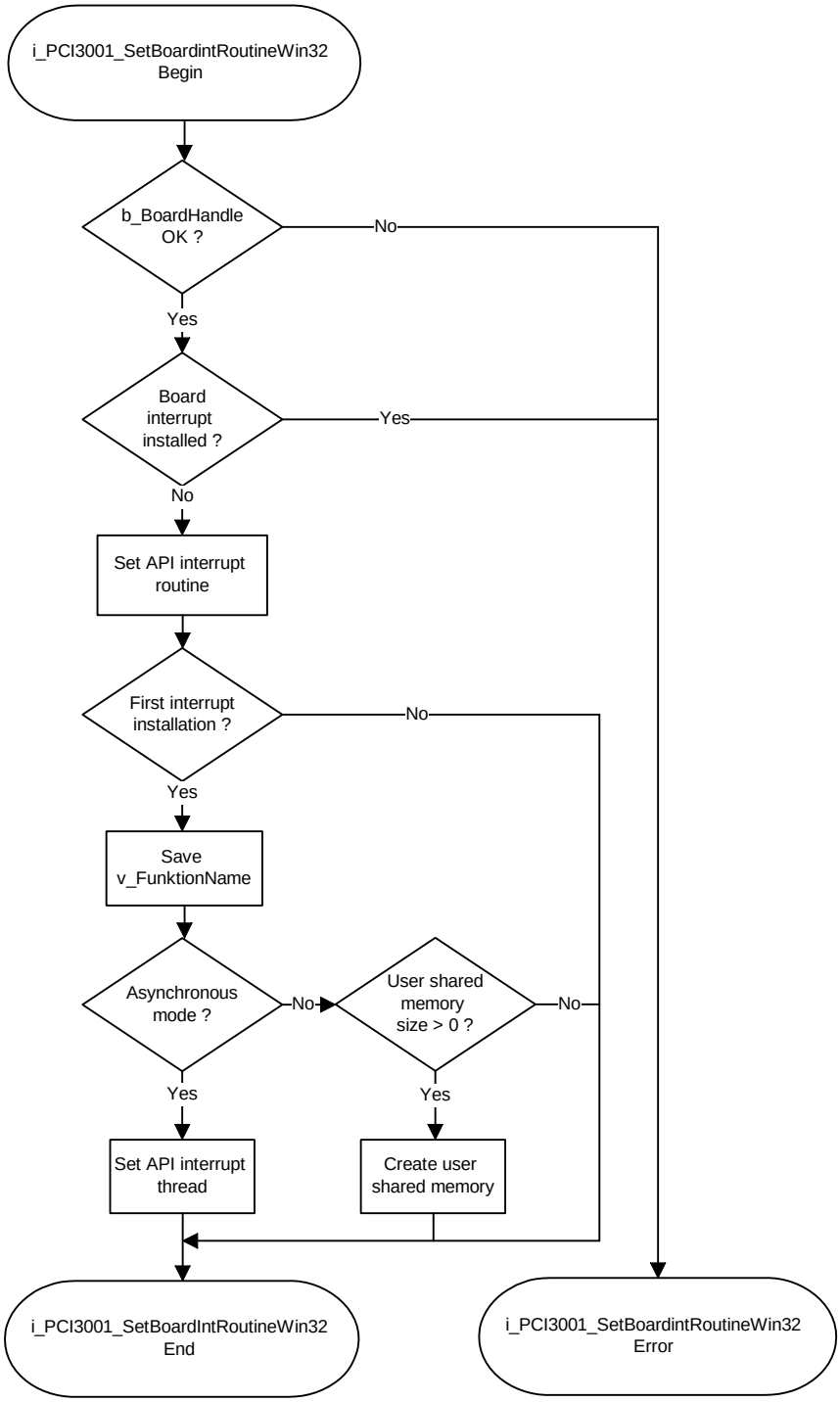
```
Sub     v_FunctionName
                    (ByVal i_BoardHandle        As Integer,
                     ByVal i_InterruptMask      As Integer,
                     l_AnalogInputValue         As INTEGER_ARRAY_16,
                     b_UserCallingMode          As Integer,
                     l_UserSharedMemory         As Long)
End Sub

Dim i_ReturnValue   As Integer
Dim i_BoardHandle As Integer
Global DynamicArray() As Integer
ReDim DynamicArray(NUMBER_OF_VALUE) As Integer
...
i_ReturnValue = i_PCI3001_SetBoardIntRoutineWin32
                                (i_BoardHandle,
                                 PCI3001_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 calling mode selected for the user interrupt routine is wrong
-4: No memory avaible for the user shared memory

Input

b_BoardHandle	v_FunctionName
b_UserCallingMode	ul_UserSharedMemorySize

Function diagram



Output

ppv_UserSharedMemory

<Return Value>

5) i_PCI3001_TestInterrupt (..)

Syntax:

```
<Return value> =  i_PCI3001_TestInterrupt (PBYTE    pb_BoardHandle,
                                           PBYTE    pb_InterruptMask,
                                           PUINT    pui_AnalogInputValue)
```

Parameter:

- Input:

No input signal occurs.

- Output:

PBYTE	pb_BoardHandle	Handle of the board **xPCI-3001** 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 inputs and of the DMA buffer are returned.

Task:

Verifies if a board **xPCI-3001** has generated an interrupt. If yes, the function returns the board handle and the interrupt source.

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

Calling convention:

ANSI C :

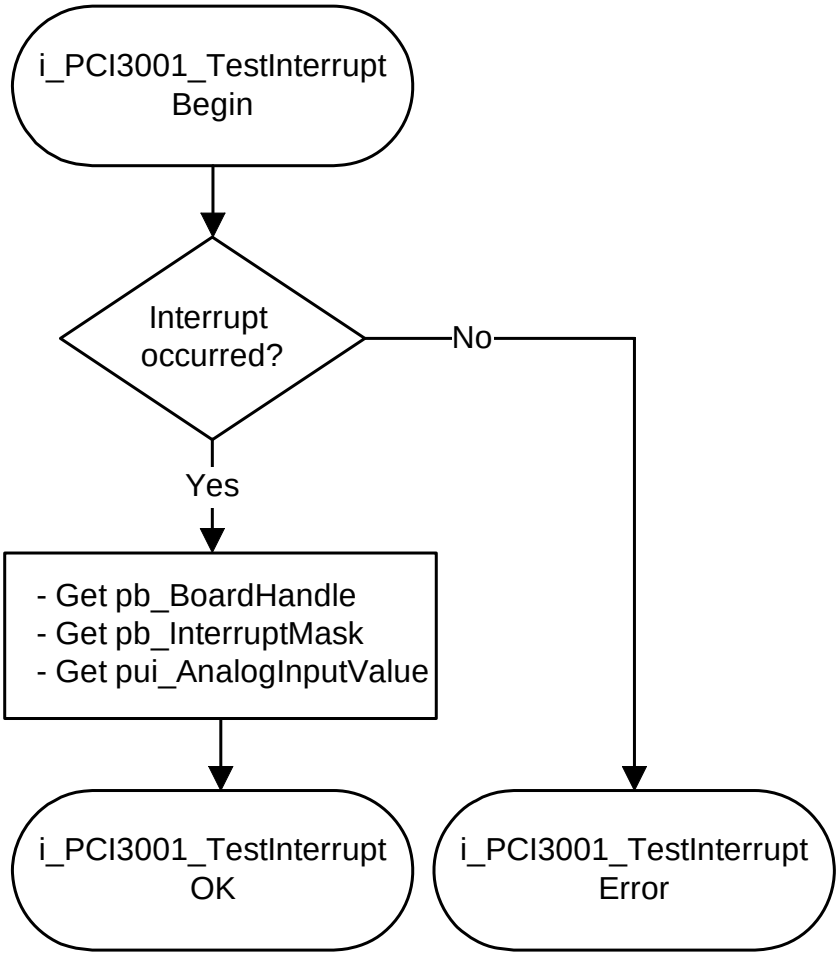
```
int             i_ReturnValue;
unsigned char   b_BoardHandle,b_InterruptMask;
unsigned int    ui_AnalogInputValue [XX];

i_ReturnValue = i_PCI3001_TestInterrupt        (&b_BoardHandle,
                                                &b_InterruptMask,
                                                ui_AnalogInputValue);
```

Return value:

-1 No interrupt
> 0 IRQ number

Input

Function diagram



Output

pb_BoardHandle pb_InterruptMask
pui_AnalogInputValue

<Return Value>

6) i_PCI3001_ResetBoardIntRoutine (..)

Syntax:
<Return value> = i_PCI3001_ResetBoardIntRoutine (BYTE b_BoardHandle)

Parameter:
- Input:
BYTE b_BoardHandle Handle of the board
- Output:
No output signal has occurred.

Task:
Stops the interrupt management of board **xPCI-3001.**
Deinstalls the user interrupt routine if the interrupt management of all boards **xPCI-3001** is stopped.

Calling convention:
ANSI C :

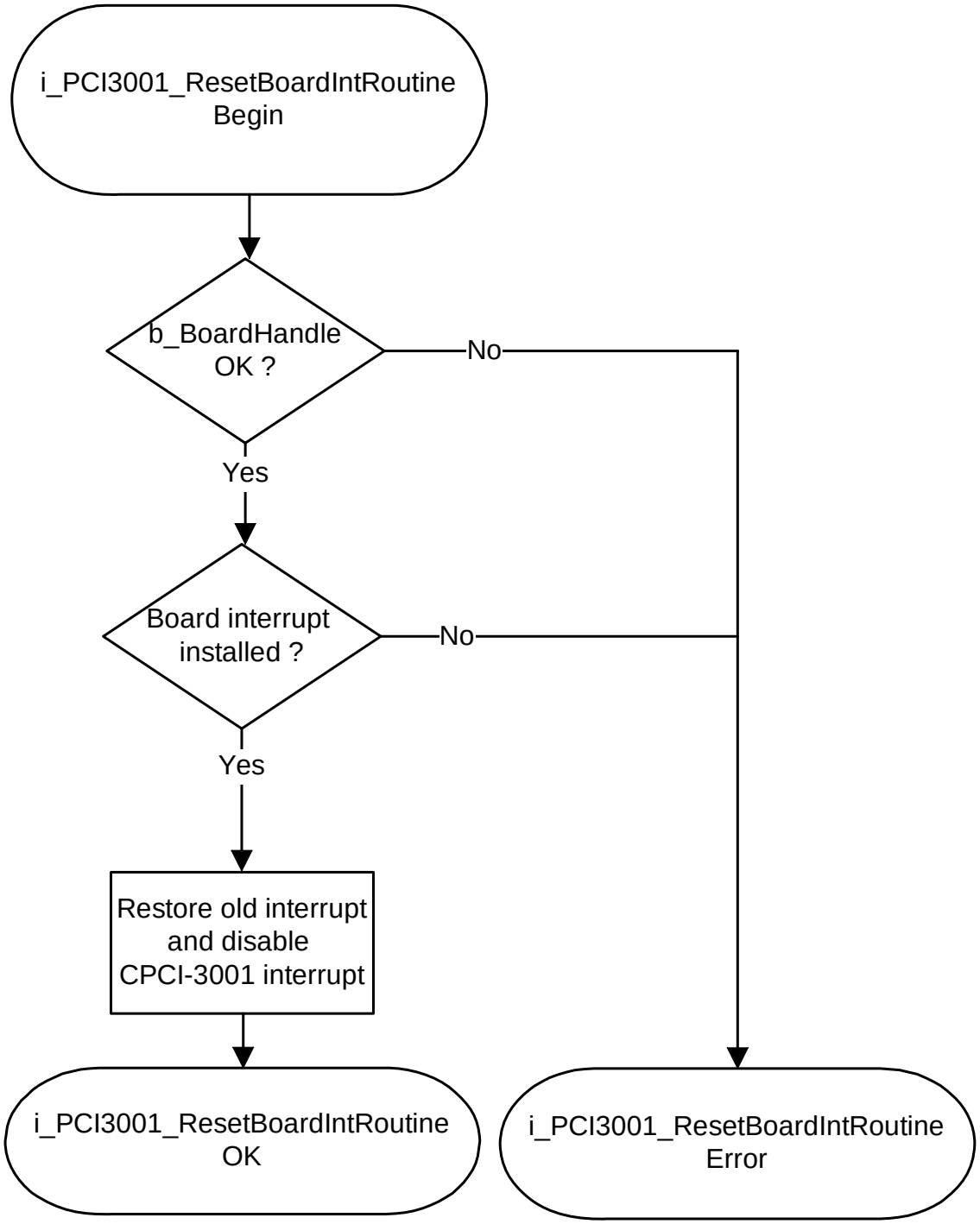
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PCI3001_ResetBoardIntRoutine    (b_BoardHandle);
```

Return value:
0: No error
-1: The handle parameter of the board is wrong
-2: No interrupt function initialised with function
"i_PCI3001_SetBoardIntRoutineXXX"

Input

b_BoardHandle

Function diagram



Output

<Return Value>

3.3 Direct conversion of analog input channels

1) i_PCI3001_Read1AnalogInput (...)

Syntax:

```
<Return value> =  i_PCI3001_Read1AnalogInput  (BYTE     b_BoardHandle,
                                               BYTE     b_Channel,
                                               BYTE     b_Gain,
                                               BYTE     b_Polarity,
                                               UINT     ui_ConvertTiming
                                               BYTE     b_InterruptFlag,
                                               PUINT    pui_AnalogInputValue)
```

Parameter:

- Input:

BYTE	b_BoardHandle	Handle of the board
BYTE	b_Channel	Number of the analog input to be read (see table 3-3). 0 to 15.
BYTE	b_Gain	Gain selection. See table 3-4 Gain 1: 0 Gain 2: 16 Gain 5: 32 Gain 10: 48
BYTE	b_Polarity	Selection of the input voltage range of the analog input to be converted. 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	PCI3001_ENABLE: An interrupt is generated at EOC. See "i_PCI3001_SetBoardIntRoutine". PCI3001_DISABLE: No interrupt is generated at EOC. The analog value is located in parameter *pui_AnalogInputValue*.

- Output:

PUINT	pui_AnalogInputValue	The analog value is returned (0 to 4095)

Task:

Reads the current value of the analog input *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_PCI3001_Read1AnalogInput      (b_BoardHandle,
                                                 PCI3001_CHANNEL_1,
                                                 PCI3001_1_GAIN,
                                                 PCI3001_UNIPOLAR,
                                                 10,
                                                 PCI3001_DISABLE,
                                                 & ui_AnalogInputValue);
```

Return value:

0: No error
-1: The handle parameter of the board is wrong
-2: The number of the analog input is wrong. See table 3-3
-3: The gain selected is wrong. See table 3-4.
-4: The input voltage range selected 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_PCI3001_SetBoardIntRoutine".
-7: Time has run down.

Table 3-3: Selection of the analog inputs

pb_ChannelArray Parameter	Analoger Eingang
PCI3001_CHANNEL_0	0
PCI3001_CHANNEL_1	1
PCI3001_CHANNEL_2	2
PCI3001_CHANNEL_3	3
PCI3001_CHANNEL_4	4
PCI3001_CHANNEL_5	5
PCI3001_CHANNEL_6	6
PCI3001_CHANNEL_7	7
PCI3001_CHANNEL_8	8
PCI3001_CHANNEL_9	9
PCI3001_CHANNEL_10	10
PCI3001_CHANNEL_11	11
PCI3001_CHANNEL_12	12
PCI3001_CHANNEL_13	13
PCI3001_CHANNEL_14	14
PCI3001_CHANNEL_15	15

Table 3-4: Gain selection

pb_GainArray Parameter	Verstärkung
PCI3001_1_GAIN	1
PCI3001_2_GAIN	2
PCI3001_5_GAIN	5
PCI3001_10_GAIN	10

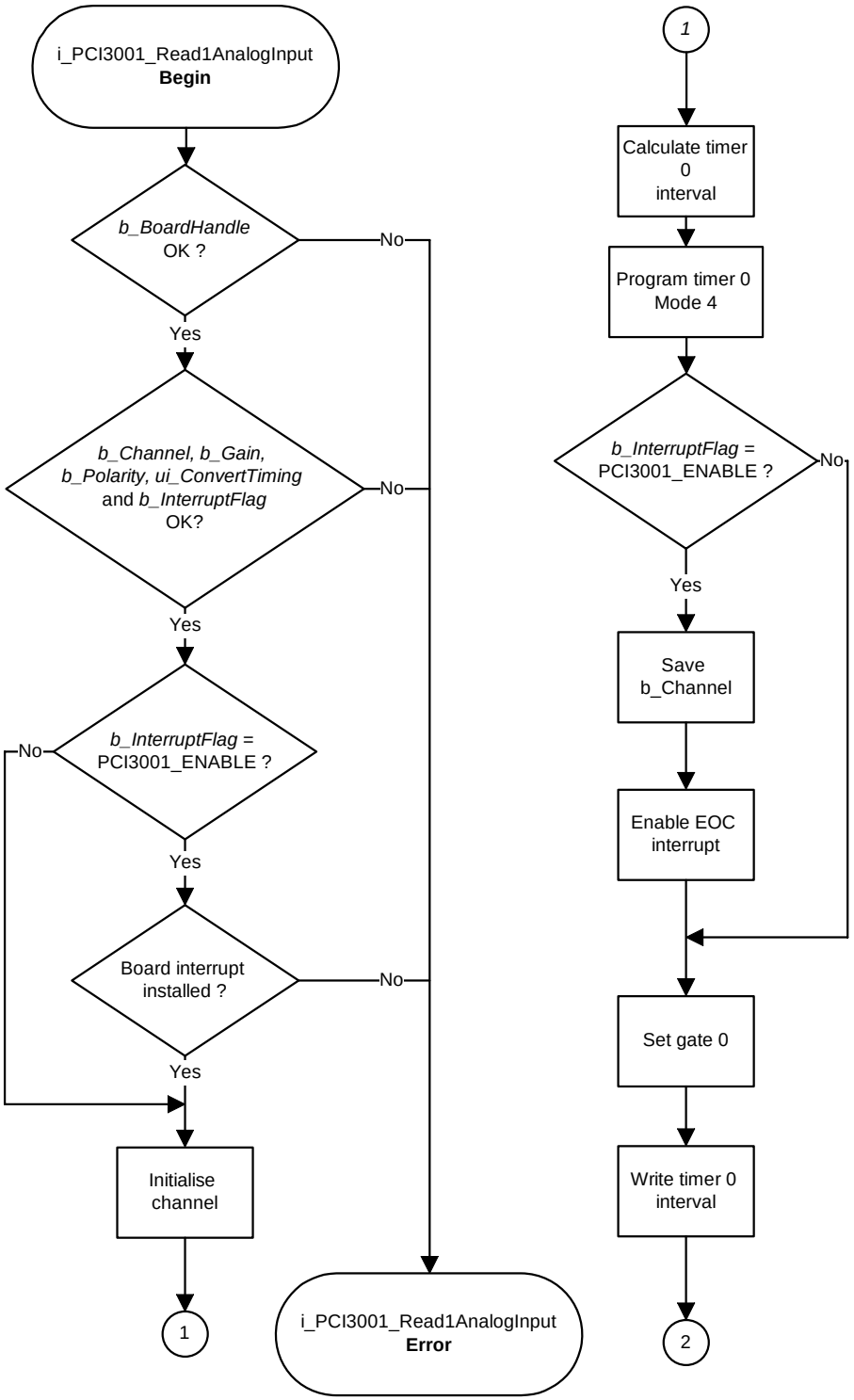
Table 3-5: Selection of the input voltage range

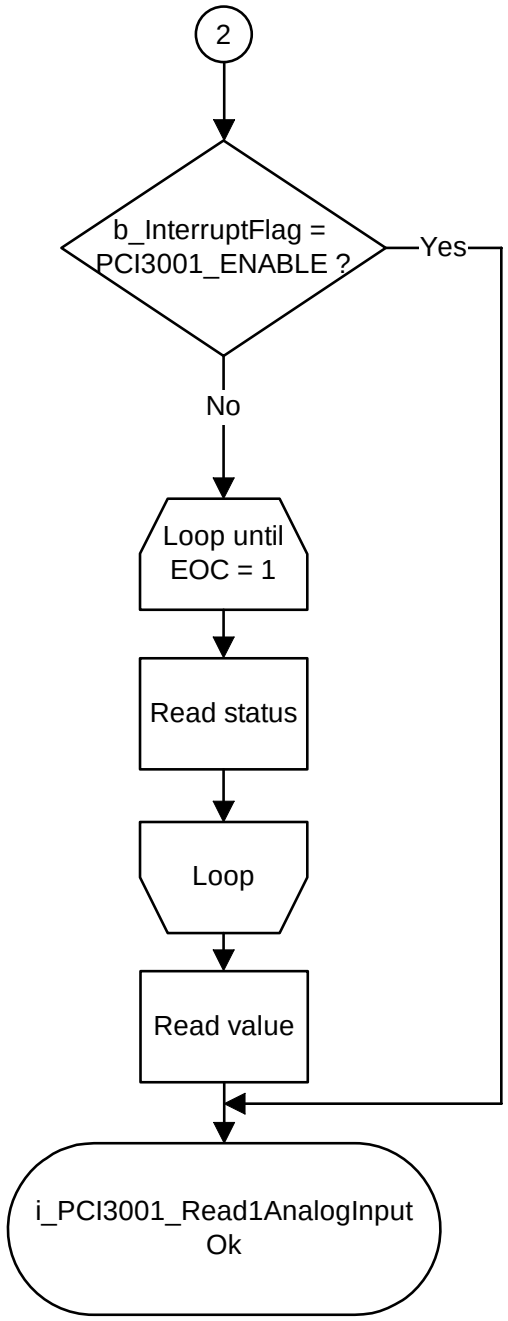
pb_PolarityArray Parameter	Spannungsbereich
PCI3001_UNIPOLAR	0-10V bei Verstärkung 1
PCI3001_BIPOLAR	±10V bei Verstärkung 1

Input

b_BoardHandle	b_Channel
b_Gain	b_Polarity
ui_ConvertTiming	b_InterruptFlag

Function diagram





Output
pui_AnalogInputValue
<Return Value>

2) i_PCI3001_ReadMoreAnalogInput (...)

Syntax:

```
<Return value> =  i_PCI3001_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)
```

Parameter:

- 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 inputs. 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	PCI3001_ENABLE: An interrupt is generated when the last conversion of the channel group is completed (EOS). See function "i_PCI3001_SetBoardIntRoutine". PCI3001_DISABLE: No interrupt is generated at the end of conversion. The analog values are located in parameter *pui_AnalogInputValueArray*.

- Output:

PUINT	pui_AnalogInputValueArray	Input values are returned

Task:

Reads several analog inputs.
The priority of the analog inputs is set with the scan list.
The scan list allows to determine the input voltage range and the gain for each analog input. The gain is defined with parameter *pb_Gain* for each analog input.
The input voltage range is defined with parameter *pb_PolarityArray* for each analog input.

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_PCI3001_ReadMoreAnalogInput  (b_BoardHandle,
                                                16,
                                                b_ChannelArray,
                                                b_GainArray,
                                                b_PolarityArray,
                                                PCI3001_DISABLE,
                                                ui_AnalogInputValue);
```

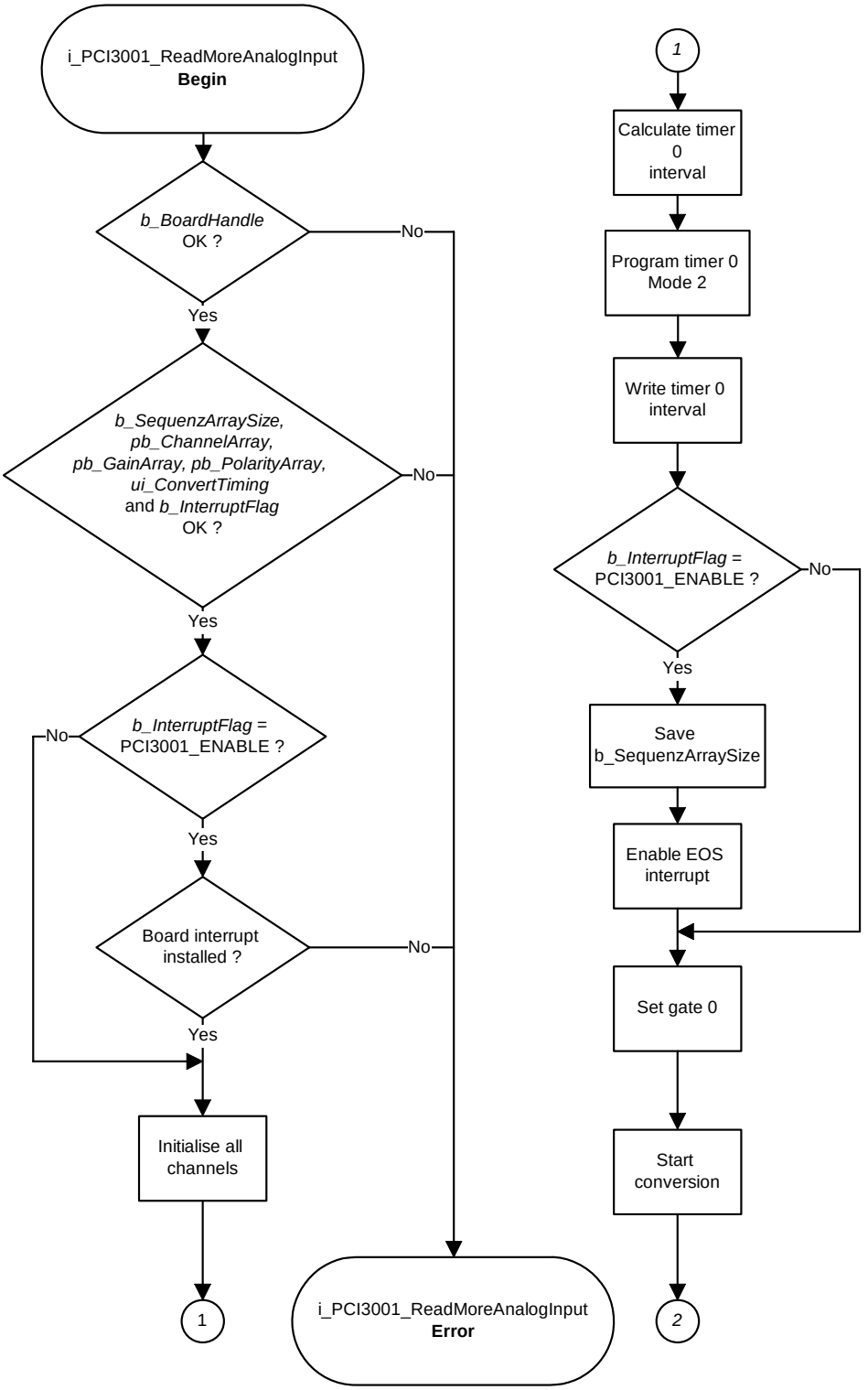
Return value:

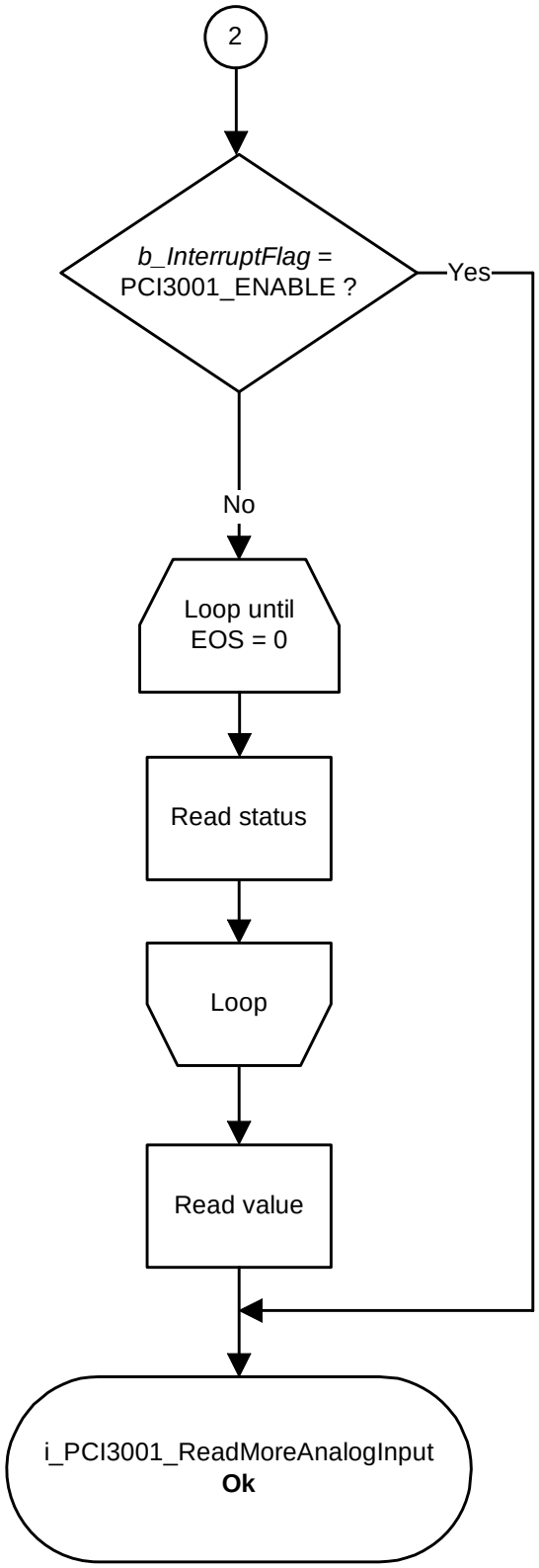
0: No error
-1: The handle parameter of the board is wrong
-2: The size of the scan list is wrong
-3: Wrong parameter detected in table "pb_ChannelArray"
-4: Wrong parameter detected in table "pb_GainArray"
-5: Wrong parameter detected in table "pb_PolarityArray"
-6: Selected conversion time is wrong
-7: Wrong parameter entered for *b_InterruptFlag*, or the user interrupt routine has not been installed.
See function "i_PCI3001_SetBoardIntRoutineXX".
-8: Time has run down.

Input

b_BoardHandle	b_SequenzArraySize
pb_ChannelArray	pb_GainArray
pb_PolarityArray	ui_ConvertTiming
b_InterruptFlag	

Function diagram





Output
pui_AnalogInputValueArray
<Return Value>

3.4 Cyclic conversion of analog input channels

IMPORTANT!
If you use Visual Basic 5 and want to make an acquisition with DMA, make sure that the function i_PCI3001_SetBoardIntRoutineWin32 has been correctly initialised.

1) i_PCI3001_InitAnalogInputAcquisition (...)

Syntax:

```
<Return value> =  i_PCI3001_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)
```

Parameter:

- 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 inputs. 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: - PCI3001_SIMPLE_MODUS: A conversion occurs every *ui_AcquisitionTiming* (time interval) See example 2. - PCI3001_DELAY_MODUS: Both times are used in this mode *ui_AcquisitionTiming* and *l_DelayTiming*. Conversions occur every *ui_AcquisitionTiming* (time interval) until all analog inputs have been acquired (determined by *b_SequenzArraySize).* Step 1 of example 3.

		Afterwards there is a waiting time of *l_DelayTiming*. Step 2 of example 3 The two steps are repeated. See example 3. - PCI3001_DELAY_1_MODUS: See example 4.
BYTE	b_ExternTrigger	Two modes are possible : - PCI3001_DISABLE: the cyclic conversion is initialised after calling the function "i_PCI3001_StartAnalogInputAcquisition". - PCI3001_ENABLE: the cyclic conversion is initialised after the calling the function "i_PCI3001_StartAnalogInputAcquisition". A logic "1" on the digital input 1 starts the conversion. The conversion is stopped by calling "i_PCI3001_StopAnalogInputAcquisition".
UINT	ui_AcquisitionTiming	Time interval in µs between 2 conversions of successive inputs. (10 µs to 32767 µs) See example 1 and 2.
LONG	l_DelayTiming	Waiting time in µs between two conversion cycles (from 100 µs to 3276750 µs) This parameter is relevant only if you use PCI3001_DELAY_MODUS or PCI3001_DELAY_1_MODUS.
ULONG	ul_NumberOfAcquisition	If you use DMA, this parameter determines how many conversions have to be performed. DOS: 1 to 32767 Win32: 1 to 2^{32} If you do not use DMA, this parameter determines how many acquisition cycles must be performed. **Warning:** if you work in DMA_CONTINUOUS with an acquisition time of 10 µs and under VB 5.0 the minimum value is 16.
BYTE	b_DMAUsed	Determines if DMA should be used or not. - PCI3001_DMA_USED: All conversion values are saved in the DMA buffer. An interrupt is generated, when *ui_NumberOfAcquisition* conversions have been completed. See function "i_PCI3001_SetBoardIntRoutineXX".

		- PCI3001_DMA_NOT_USED: An interrupt is generated when the conversion of the last channel group has been completed. The analog value is returned through the user interrupt routine. See function "i_PCI3001_SetBoardIntRoutineXX".
BYTE	b_AcquisitionCycle	Determines the type of DMA conversion. - PCI3001_CONTINUOUS: An interrupt is generated each time a DMA conversion cycle is completed. A new DMA conversion cycle is then started. - PCI3001_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 inputs is set through the scan list.
The scan list allows to set the input voltage range and the gain for each analog input. See example 1.
The DMA option (PCI3001_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_PCI3001_SetBoardIntRoutineXX".

You have to:
- set the priority of the analog inputs through the scan list.
- enter the mode through the parameter *b_AcquisitionMode*.
- enter the time between two conversions through the parameter *ui_AcquisitionTiming*.
- enter the waiting time between two conversion cycles through the parameter *l_DelayTiming*, if you work in mode PCI3001_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 PCI3001_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 inputs, gain and input voltage range

```
b_RamArraySize          = 16
pb_ChannelArray [0]     = PCI3001_CHANNEL_0
pb_ChannelArray [1]     = PCI3001_CHANNEL_1
pb_ChannelArray [2]     = PCI3001_CHANNEL_2
pb_ChannelArray [3]     = PCI3001_CHANNEL_3
pb_ChannelArray [4]     = PCI3001_CHANNEL_2
pb_ChannelArray [5]     = PCI3001_CHANNEL_1
pb_ChannelArray [6]     = PCI3001_CHANNEL_0
pb_ChannelArray [7]     = PCI3001_CHANNEL_4
pb_ChannelArray [8]     = PCI3001_CHANNEL_5
pb_ChannelArray [9]     = PCI3001_CHANNEL_6
pb_ChannelArray [10]    = PCI3001_CHANNEL_7
pb_ChannelArray [11]    = PCI3001_CHANNEL_8
pb_ChannelArray [12]    = PCI3001_CHANNEL_9
pb_ChannelArray [13]    = PCI3001_CHANNEL_10
pb_ChannelArray [14]    = PCI3001_CHANNEL_11
pb_ChannelArray [15]    = PCI3001_CHANNEL_12

pb_GainArray     [0]    = PCI3001_1_GAIN
pb_PolarityArray[0]     = PCI3001_UNIPOLAR
pb_GainArray     [1]    = PCI3001_1_GAIN
pb_PolarityArray[1]     = PCI3001_UNIPOLAR
pb_GainArray     [2]    = PCI3001_1_GAIN
pb_PolarityArray[2]     = PCI3001_UNIPOLAR
pb_GainArray     [3]    = PCI3001_1_GAIN
pb_PolarityArray[3]     = PCI3001_UNIPOLAR
pb_GainArray     [4]    = PCI3001_5_GAIN
pb_PolarityArray[4]     = PCI3001_BIPOLAR
pb_GainArray     [5]    = PCI3001_5_GAIN
pb_PolarityArray[5]     = PCI3001_BIPOLAR
pb_GainArray     [6]    = PCI3001_5_GAIN
pb_PolarityArray[6]     = PCI3001_BIPOLAR
pb_GainArray     [7]    = PCI3001_10_GAIN
pb_PolarityArray[7]     = PCI3001_UNIPOLAR
pb_GainArray     [8]    = PCI3001_10_GAIN
pb_PolarityArray[8]     = PCI3001_UNIPOLAR
pb_GainArray     [9]    = PCI3001_10_GAIN
pb_PolarityArray[9]     = PCI3001_BIPOLAR
pb_GainArray     [10]   = PCI3001_1_GAIN
pb_PolarityArray[10]    = PCI3001_UNIPOLAR
pb_GainArray     [11]   = PCI3001_1_GAIN
pb_PolarityArray[11]    = PCI3001_UNIPOLAR
pb_GainArray     [12]   = PCI3001_2_GAIN
pb_PolarityArray[12]    = PCI3001_UNIPOLAR
pb_GainArray     [13]   = PCI3001_2_GAIN
pb_PolarityArray[13]    = PCI3001_UNIPOLAR
pb_GainArray     [14]   = PCI3001_2_GAIN
pb_PolarityArray[14]    = PCI3001_UNIPOLAR
```

pb_GainArray [15] = PCI3001_1_GAIN
pb_PolarityArray[15] = PCI3001_UNIPOLAR

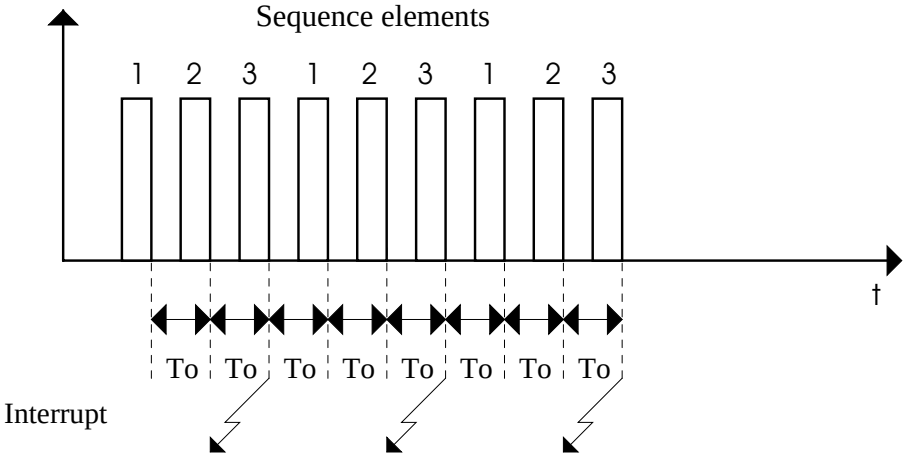
In this example the priority is set as follows:

Element	Analog input	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 and external trigger

```
i_PCI3001_InitAnalogInputAcquisition
                        (b_BoardHandle,
                         3,
                         pb_ChannelArray,
                         pb_GainArray,
                         pb_PolarityArray,
                         PCI3001_SIMPLE_MODUS,
                         PCI3001_DISABLE,
                         T0,
                         0,
                         3,
                         PCI3001_DMA_NOT_USED,
                         PCI3001_SINGLE)
```

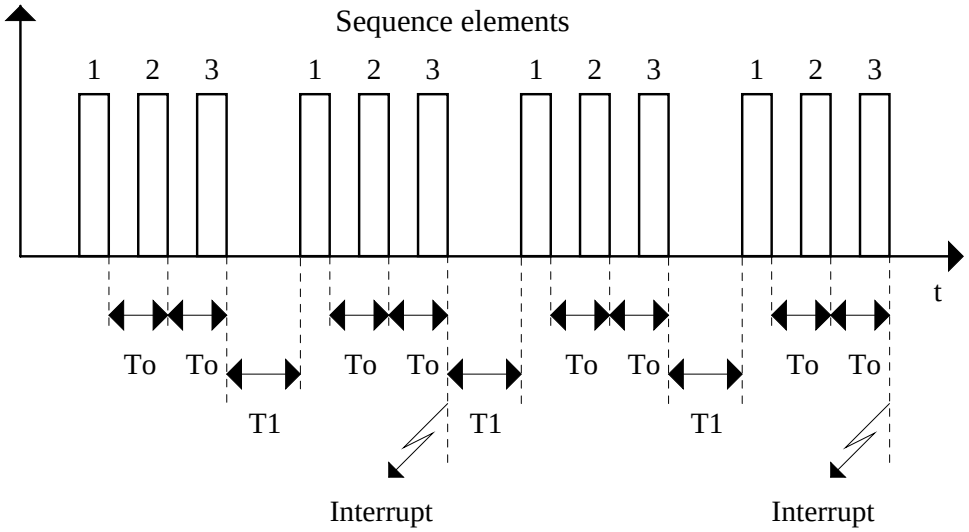
- b_AcquisitionMode = PCI3001_SIMPLE_MODUS
- b_ExternTrigger = PCI3001_DISABLE
- ui_AcquisitionTiming = To
- b_DMAUsed = PCI3001_DMA_NOT_USED
- ui_NumberOfAquisisition = 3



Example 3: Cyclic conversion with DMA without external trigger

```
i_PCI3001_InitAnalogInputAcquisition
                    (b_BoardHandle,
                     3,
                     pb_ChannelArray,
                     pb_GainArray,
                     pb_PolarityArray,
                     PCI3001_DELAY_MODUS,
                     PCI3001_DISABLE,
                     T0,
                     T1,
                     6,
                     PCI3001_DMA_USED,
                     PCI3001_CONTINUOUS)
```

- b_AcquisitionMode = PCI3001_DELAY_MODUS
- b_ExternTrigger = PCI3001_DISABLE
- ui_NumberOfAcquisition = 6
- ui_AcquisitionTiming = To
- l_DelayTiming = T1
- b_DMAUsed = PCI3001_DMA_USED
- b_AcquisitionCycle = PCI3001_CONTINUOUS
- b_ExternTrigger = PCI3001_DISABLE

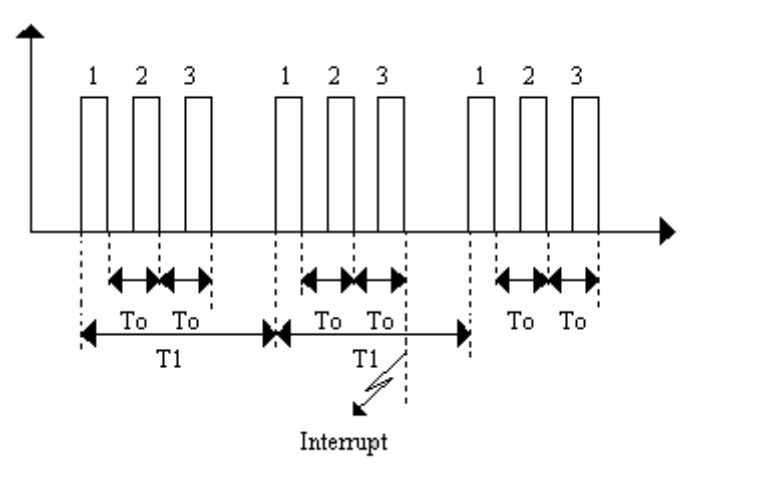


Example 4: Cyclic conversion with DMA without external trigger

```
i_PCI3001_InitAnalogInputAcquisition
                        (b_BoardHandle,
                         3,
                         pb_ChannelArray,
                         pb_GainArray,
                         pb_PolarityArray,
                         PCI3001_DELAY_1_MODUS,
                         PCI3001_DISABLE,
                         T0,
                         T1,
                         6,
                         PCI3001_DMA_USED,
                         PCI3001_CONTINUOUS)
```

- b_AcquisitionMode = PCI3001_DELAY_1_MODUS
- b_ExternTrigger = PCI3001_DISABLE
- ui_NumberOfAcquisition = 6
- ui_AcquisitionTiming = To
- l_DelayTiming = T1
- b_DMAUsed = PCI3001_DMA_USED
- b_AcquisitionCycle = PCI3001_CONTINUOUS
- b_ExternTrigger = PCI3001_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_PCI3001_InitAnalogInputAcquisition
                                                (b_BoardHandle,16,
                                                 b_ChannelArray,
                                                 b_GainArray,
                                                 b_PolarityArray,
                                                 PCI3001_DELAY_MODUS,
                                                 PCI3001_DISABLE,
                                                 100, 3000, 1000,
                                                 DMA_NOT_USED,
                                                 PCI3001_SINGLE);
```

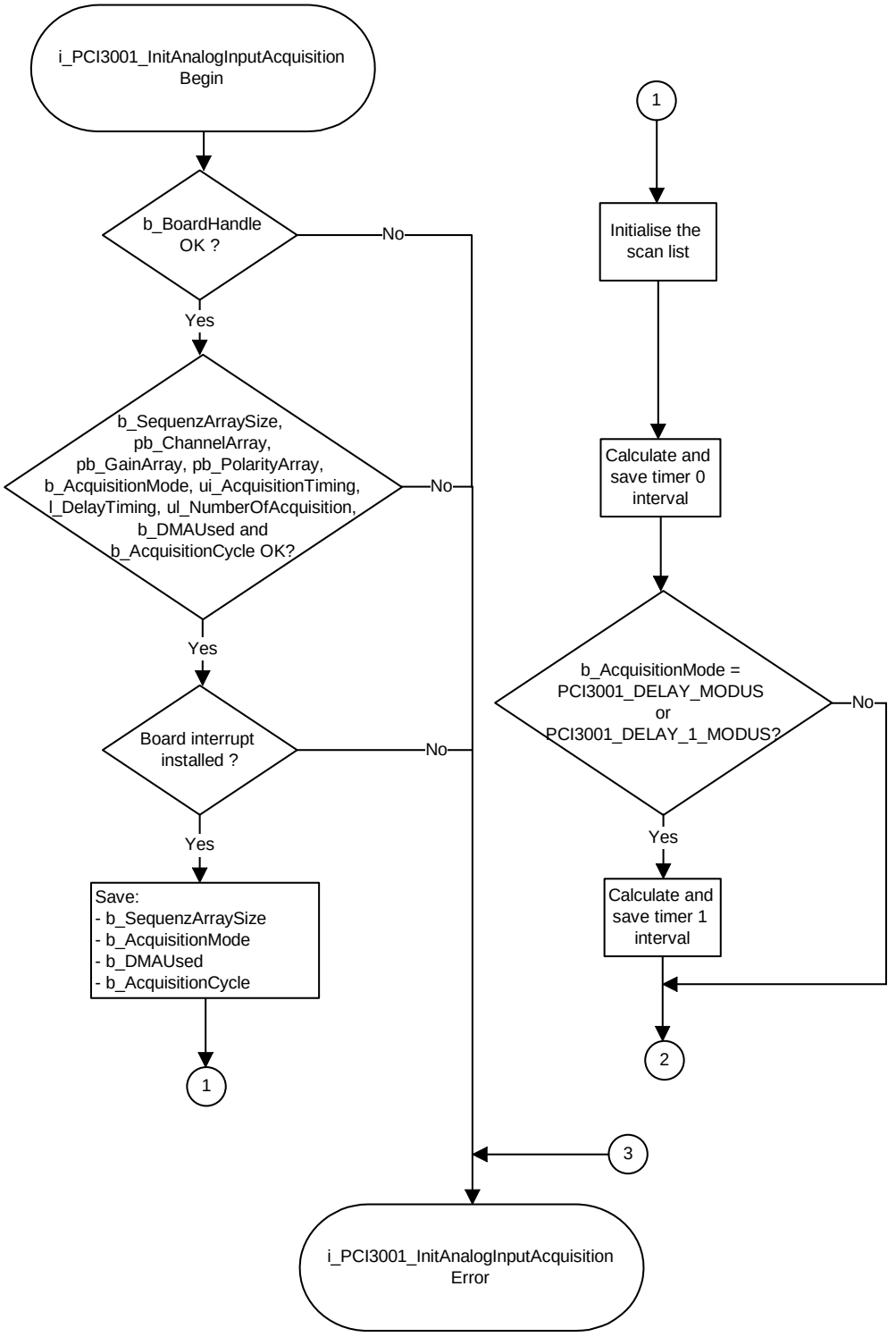
Return value:

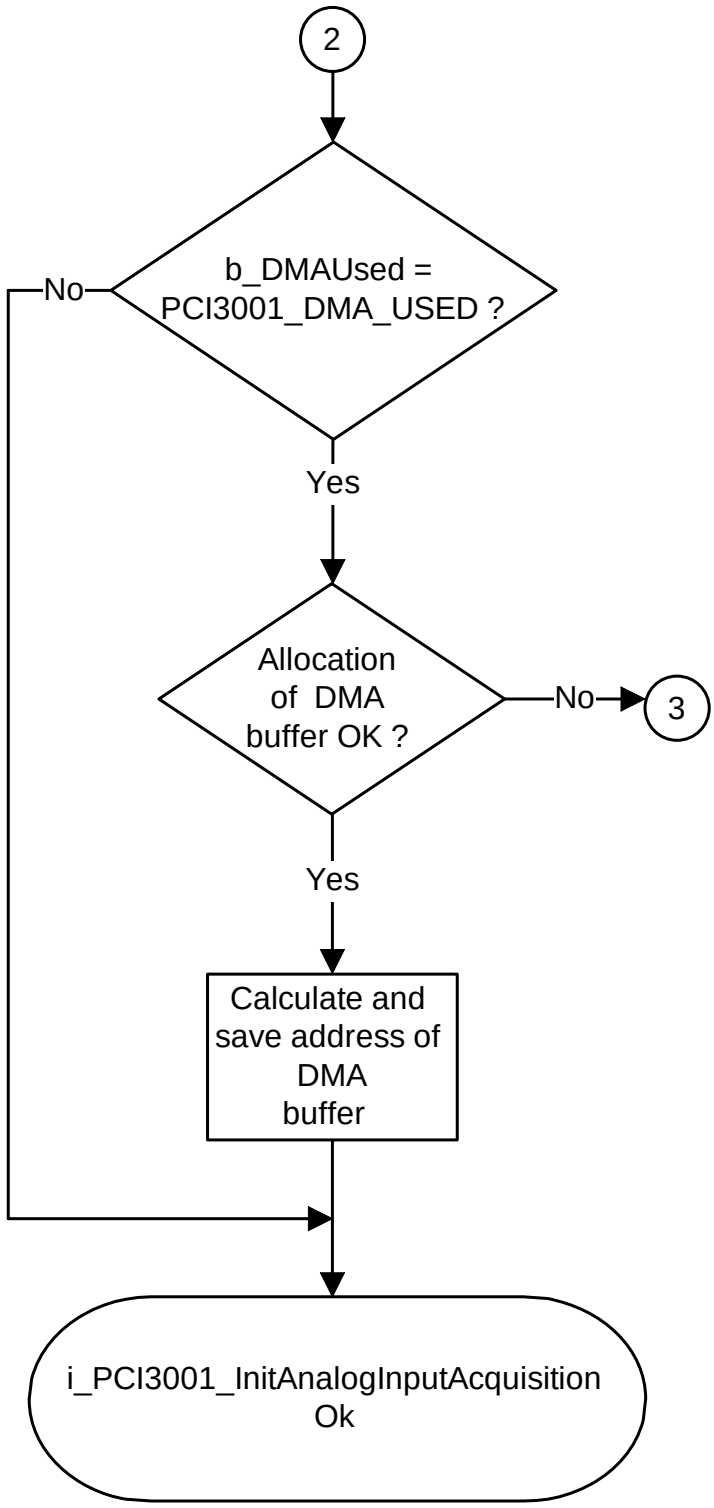
 0: No error
-1: The handle parameter of the board is wrong
-2: User interrupt routine has not been installed.
    See function "i_PCI3001_SetBoardIntRoutineXX"
-3: Size of the scan list is wrong
-4: Wrong parameter detected in table "pb_ChannelArray"
-5: Wrong parameter detected in table "pb_GainArray"
-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
     (PCI3001_CONTINUOUS or PCI3001_SINGLE)
-11: Parametered conversion cycle is wrong
     (PCI3001_SIMPLE_MODUS, PCI3001_DELAY_MODUS
      or PCI3001_DELAY_1_MODUS)
-12: Not enough memory available.
-13: External trigger is wrong.
-14: ul_NumberofAcquisition is wrong.
-15: DMA cannot be used under Windows (16-bit)

Input

b_BoardHandle	b_SequenzArraySize
pb_ChannelArray	pb_GainArray
pb_PolarityArray	b_AcquisitionMode
b_ExternTrigger	ui_AcquisitionTiming
l_DelayTiming	ul_NumberOfAcquisition
b_DMAUsed	b_AcquisitionCycle

Function diagram





Output

<Return Value>

2) i_PCI3001_StartAnalogInputAcquisition (...)

Syntax:
<Return value> = i_PCI3001_StartAnalogInputAcquisition
(BYTE b_BoardHandle)

Parameter:
- 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_PCI3001_InitAnalogInputAcquisition".

Calling convention:
ANSI C :

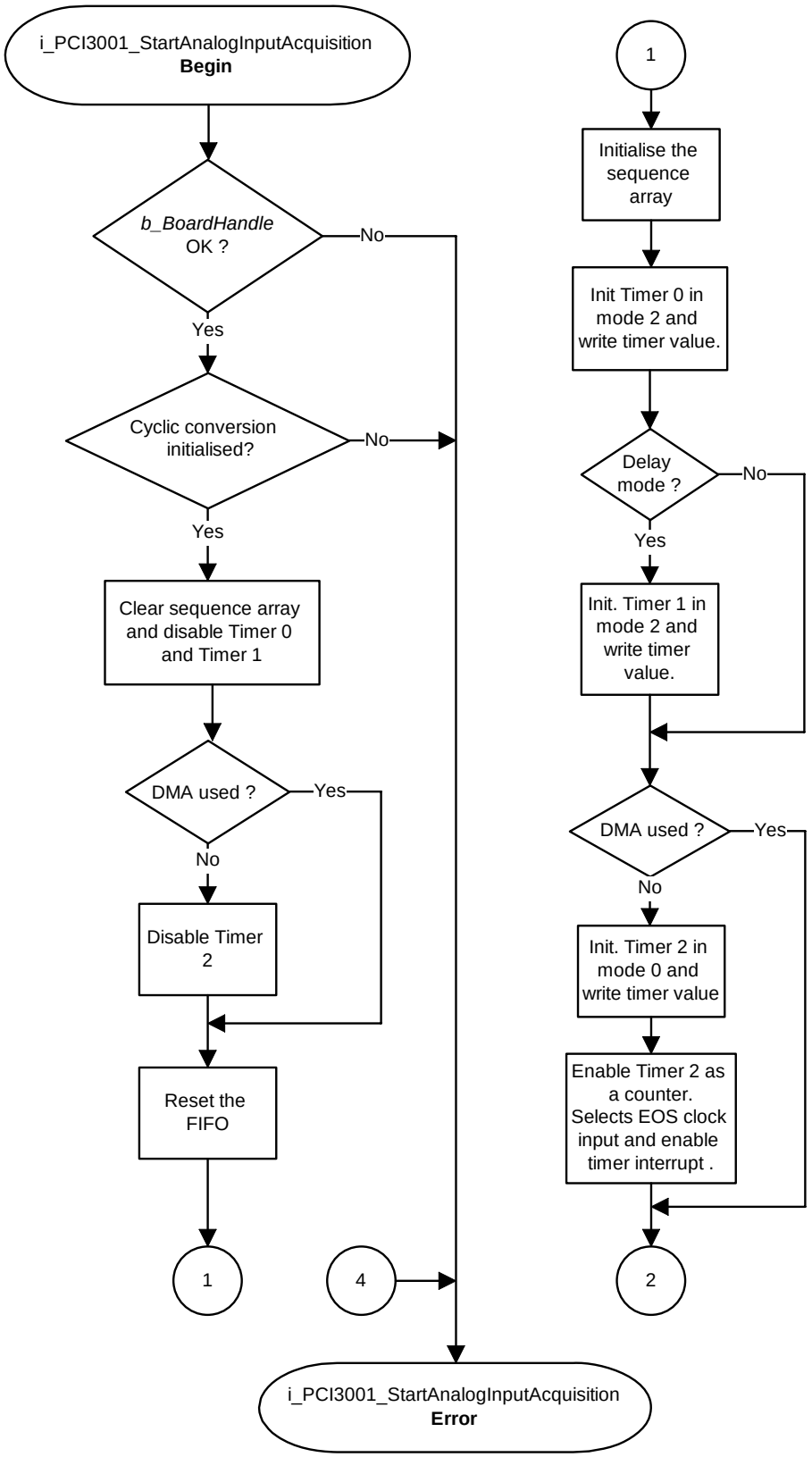
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PCI3001_StartAnalogInputAcquisition   (b_BoardHandle);
```

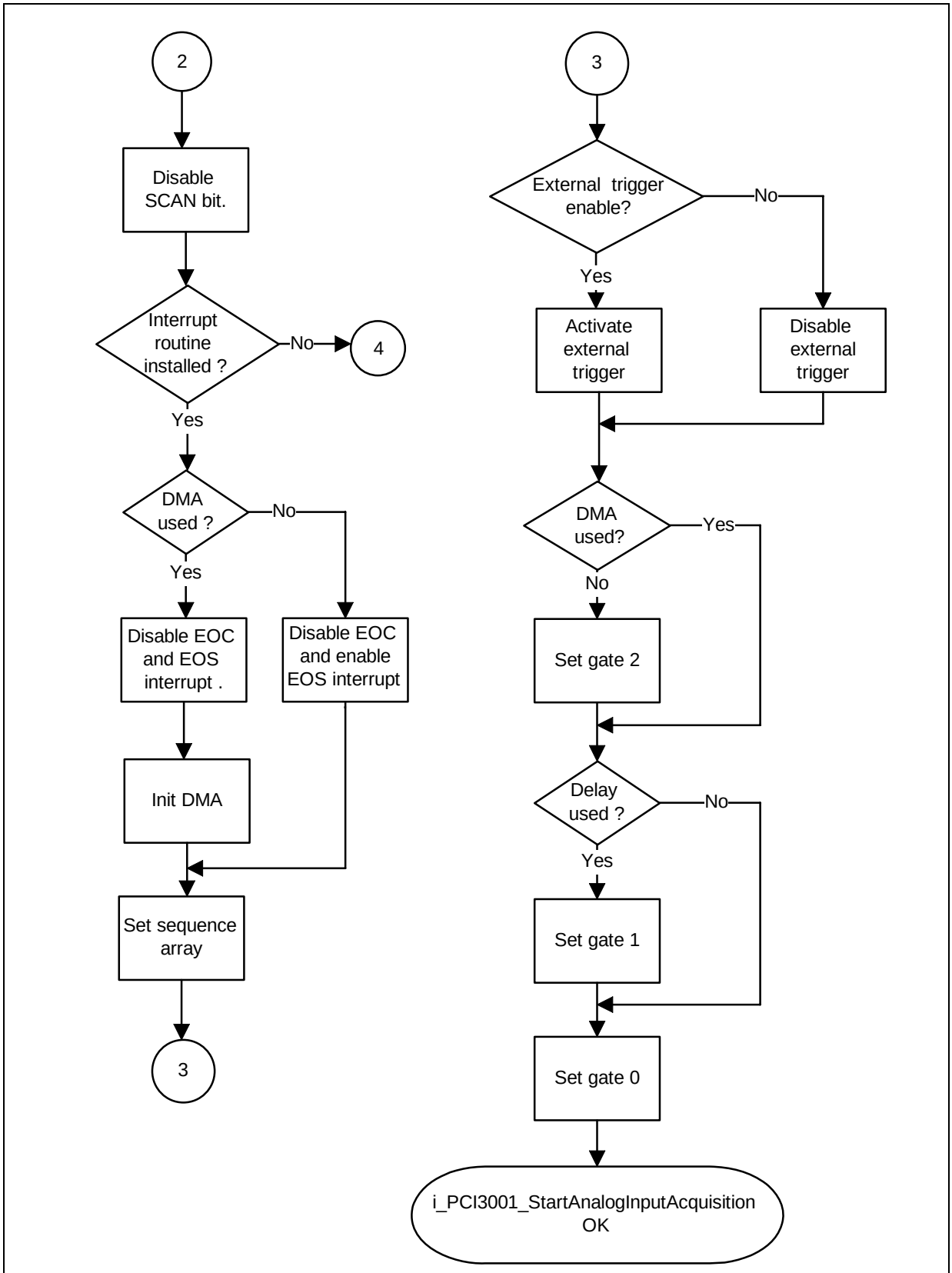
Return value:
0: No error
-1: The handle parameter of the board is wrong
-2: The cyclic conversion has not been initialised.
Please use function "i_PCI3001_InitAnalogInputAcquisition"
-3: User interrupt routine has not been installed.
See function "i_PCI3001_SetBoardIntRoutineXX"

Input

b_BoardHandle

Function diagram





Output

<Return Value>

3) i_PCI3001_StopAnalogInputAcquisition (...)

Syntax:
<Return value> = i_PCI3001_StopAnalogInputAcquisition
(BYTEb_BoardHandle)

Parameter:
- Input:
BYTE b_BoardHandle Handle of the board
- Output:
No output signal has occurred.

Task:
Stops the cyclic conversion. It has been started previously with function "i_PCI3001_StartAnalogInputAcquisition"

Calling convention:
ANSI C :

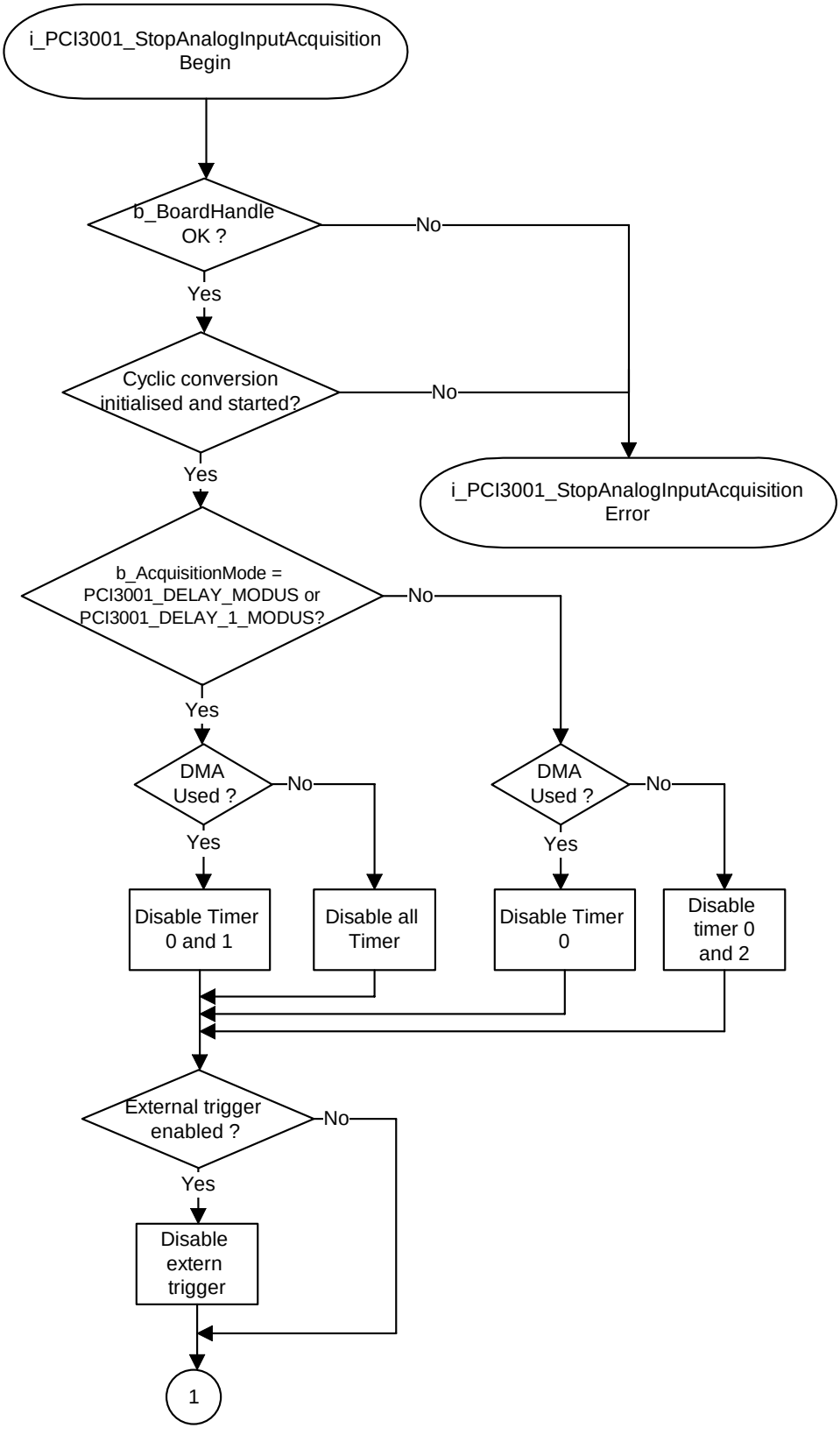
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PCI3001_StopAnalogInputAcquisition   (b_BoardHandle);
```

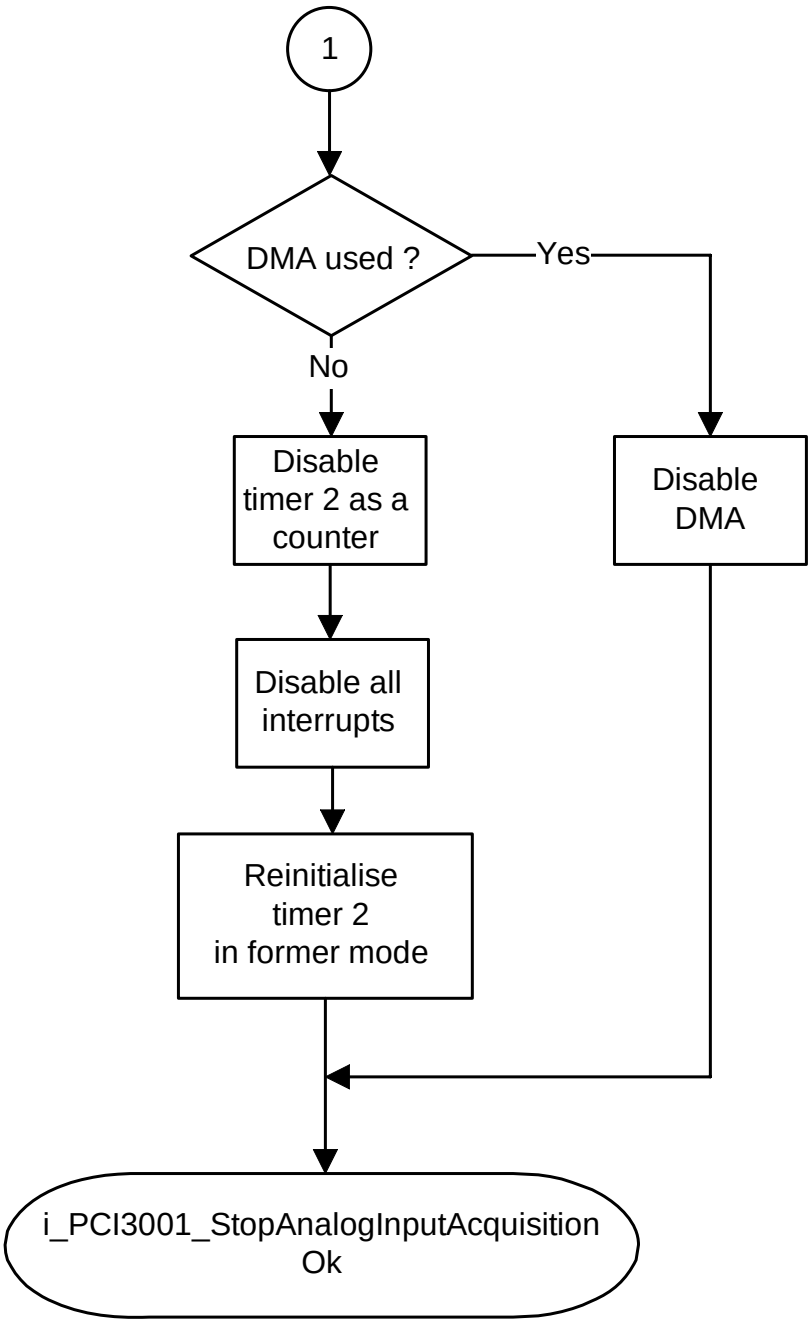
Return value:
0: No error
-1: The handle parameter of the board is wrong
-2: The cyclic conversion has not been started.
Please use function "i_PCI3001_StartAnalogInputAcquisition"

Input

b_BoardHandle

Function diagram





Output

<Return Value>

4) i_PCI3001_ClearAnalogInputAcquisition(...)

Syntax:
<Return value> = i_PCI3001_ClearAnalogInputAcquisition
(BYTE b_BoardHandle)

Parameter:
- Input:
BYTE b_BoardHandle Handle of the board
- Output:
No output signal has occurred.

Task:
Releases the DMA buffer.

Calling convention:
ANSI C :

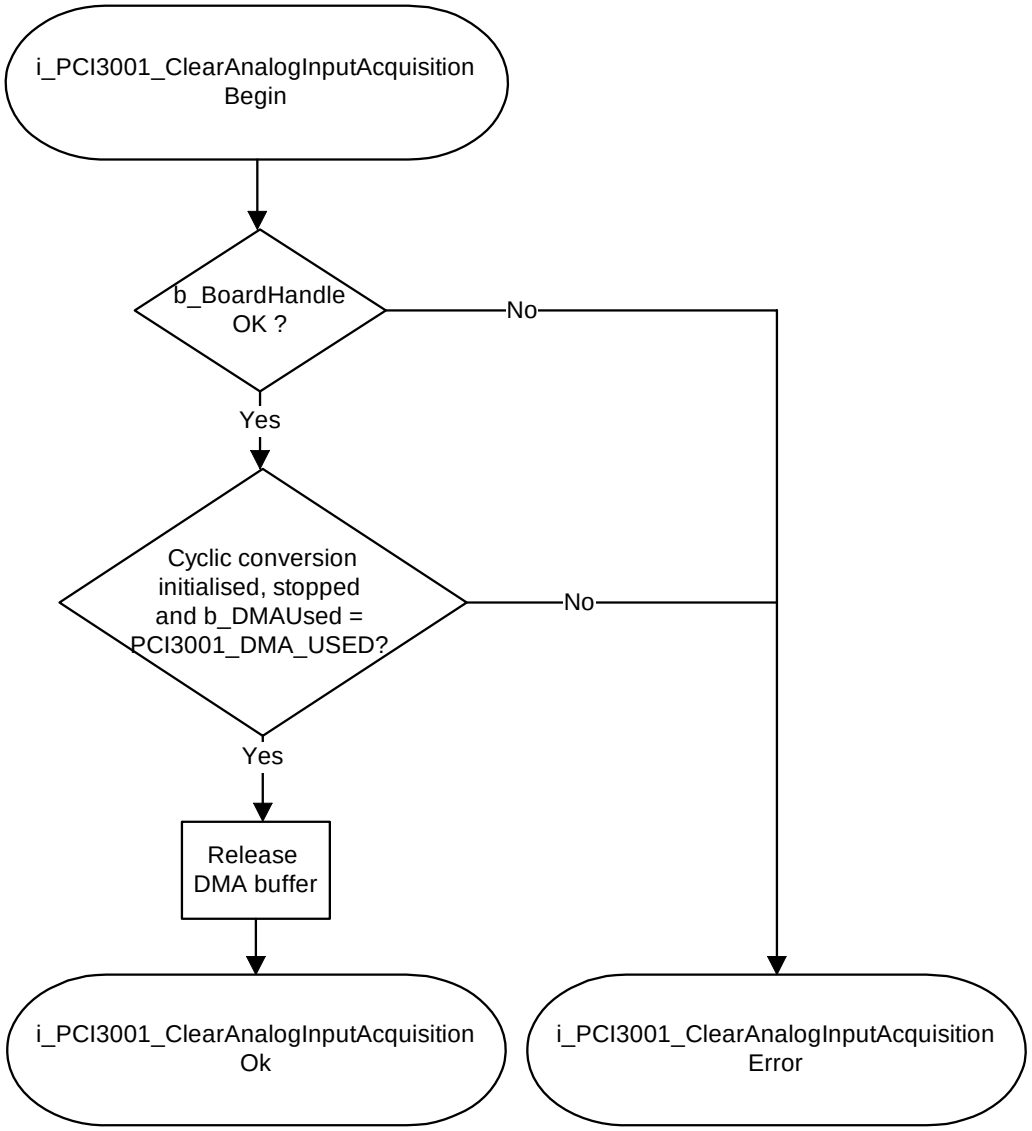
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PCI3001_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_PCI3001_InitAnalogInputAcquisition"

Input

b_BoardHandle

Function diagram



Output

<Return Value>

3.5 Timer

1) i_PCI3001_InitTimer (...)

Syntax:
<Return value> = i_PCI3001_InitTimer (BYTE b_BoardHandle,
LONG l_DelayValue,
BYTE b_InterruptFlag)

Parameter:
- Input:

BYTE	b_BoardHandle	Handle of the board
LONG	l_DelayValue	Time interval of the timer from 100 µs up to 838.5 s
BYTE	b_InterruptFlag	PCI3001_ENABLE: Timer: an interrupt is generated at the end of each time interval PCI3001_DISABLE: No interrupt is generated.

- Output:
No output signal has occurred.

Task:
Initialises the timer as an edge generator.

Calling convention:
ANSI C :

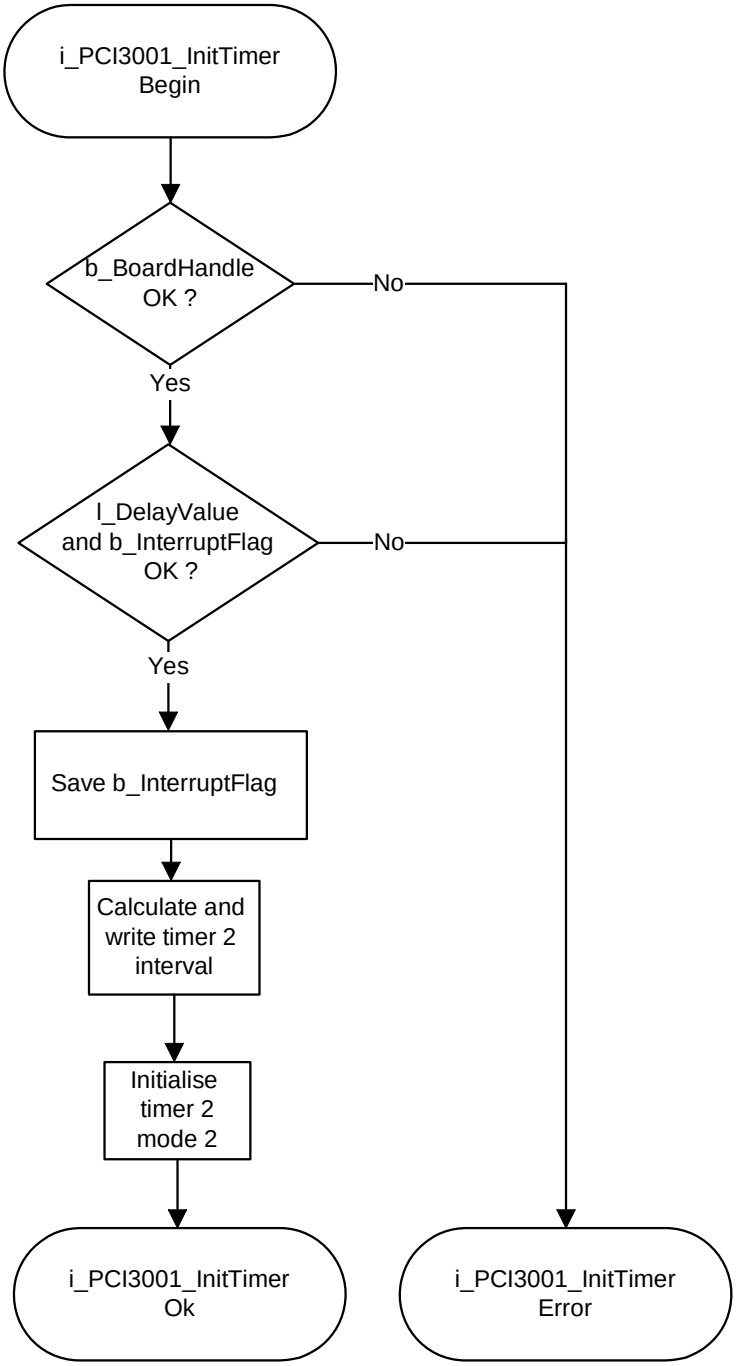
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PCI3001_InitTimer     (b_BoardHandle,
                                         1000,
                                         PCI3001_DISABLE);
```

Return value:
0: No error.
-1: The handle parameter of the board is wrong.
-2: The user interrupt routine has not been installed
See function "i_PCI3001_SetBoardIntRoutineXX"
-3: The interrupt parameter is wrong
-4: Time selection is wrong

Input

b_BoardHandle l_DelayValue
b_InterruptFlag

Function diagram



Output

<Return Value>

2) i_PCI3001_StartTimer (...)

Syntax:
<Return value> = i_PCI3001_StartTimer (BYTE b_BoardHandle)

Parameter:
- Input:
BYTE b_BoardHandle Handle of the board
- Output:
No output signal has occurred.

Task:
Starts the timer.

Calling convention:
ANSI C :

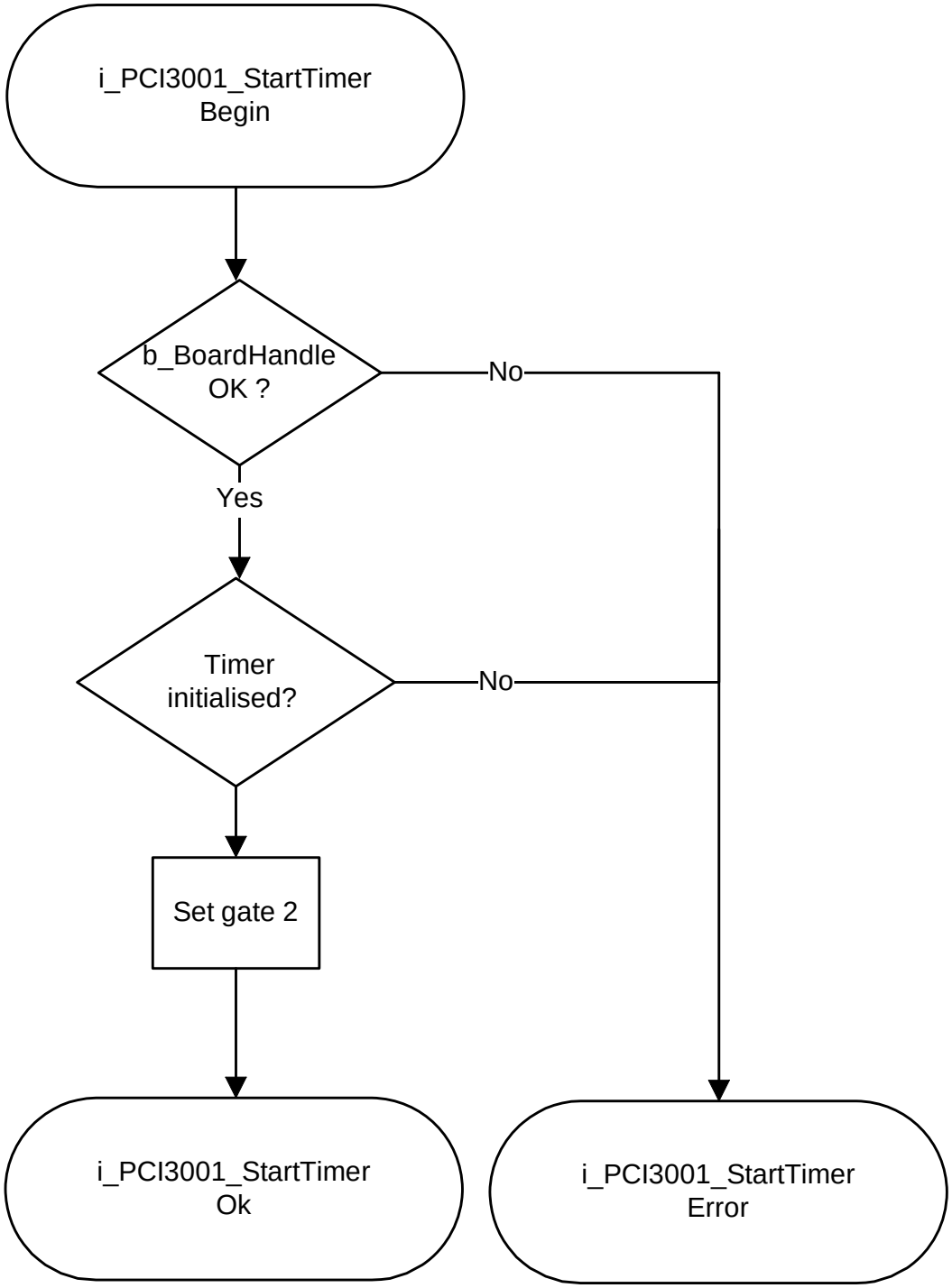
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PCI3001_StartTimer    (b_BoardHandle);
```

Return Value:
0: No error.
-1: The handle parameter of the board is wrong.
-2: Timer has not been initialised.

Input

b_BoardHandle

Function diagram



Output

<Return Value>

3) i_PCI3001_StopTimer (...)

Syntax:
<Return value> = i_PCI3001_StopTimer (BYTE b_BoardHandle)

Parameter:
- Input:
BYTE b_BoardHandle Handle of the board **xPCI-3001**
- Output:
No output signal has occurred.

Task:
Stops the timer.

Calling convention:
ANSI C :

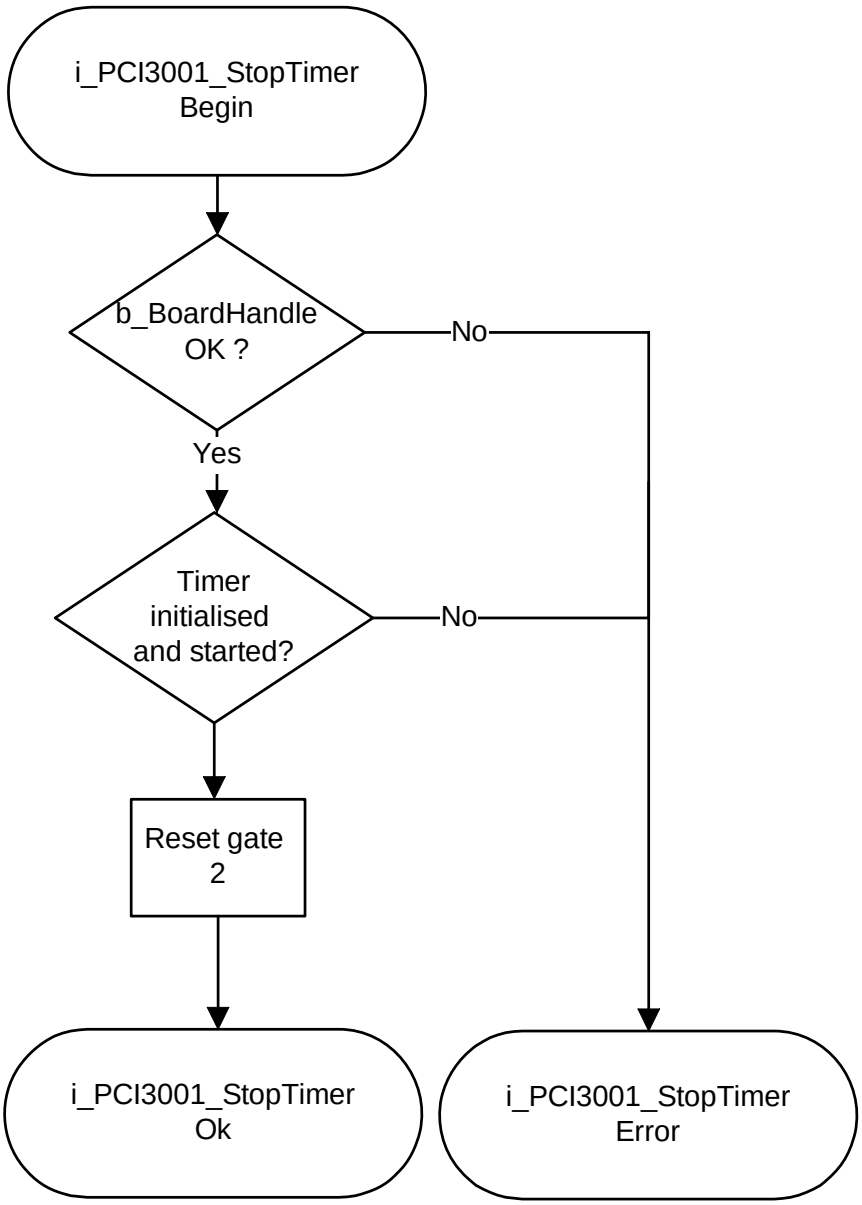
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PCI3001_StopTimer     (b_BoardHandle);
```

Return Value:
0: No error.
-1: The handle parameter of the board is wrong.
-2: Timer has not been initialised.
-3: Timer has not been started.

Input

b_BoardHandle

Function diagram



Output

<Return Value>

4) i_PCI3001_ReadTimer (...)

Syntax:
<Return value> = i_PCI3001_ReadTimer (BYTE b_BoardHandle
LONG pl_ReadValue)

Parameter:
- Input:

BYTE	b_BoardHandle	Handle of the board **xPCI-3001**

- Output:

PLONG	pl_ReadValue	Current timer value (from 0 to FFFFFF Hex)

Task:
Reads the current value of the timer.

Calling convention:
ANSI C :

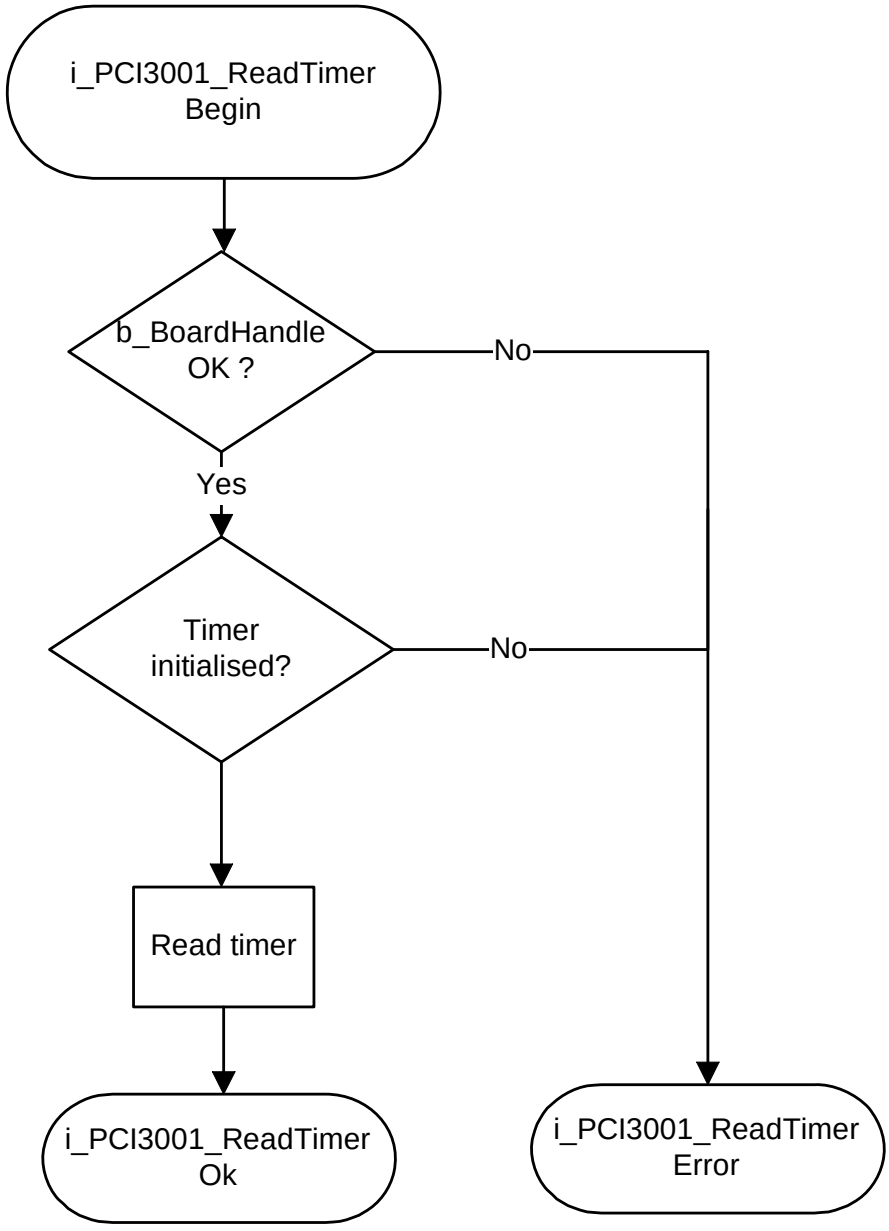
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
long            l_ReadValue;

i_ReturnValue = i_PCI3001_ReadTimer (b_BoardHandle,
                                     &l_ReadValue);
```

Return value:
0: No error.
-1: The handle parameter of the board is wrong.
-2: Timer has not been initialised.

Input

b_BoardHandle

Function diagram



Output

pl_ReadValue

<Return Value>

5) i_PCI3001_WriteTimer (...)

Syntax:
<Return value> = i_PCI3001_WriteTimer (BYTE b_BoardHandle
LONG l_WriteValue)

Parameter:
- Input:

BYTE	b_BoardHandle	Handle of the board **xPCI-3001**
LONG	l_WriteValue	New timer value (from 0 to FFFFFF Hex)

- Output:
No output signal has occurred.

Task:
Writes a new value in the timer.

Calling convention:
ANSI C :

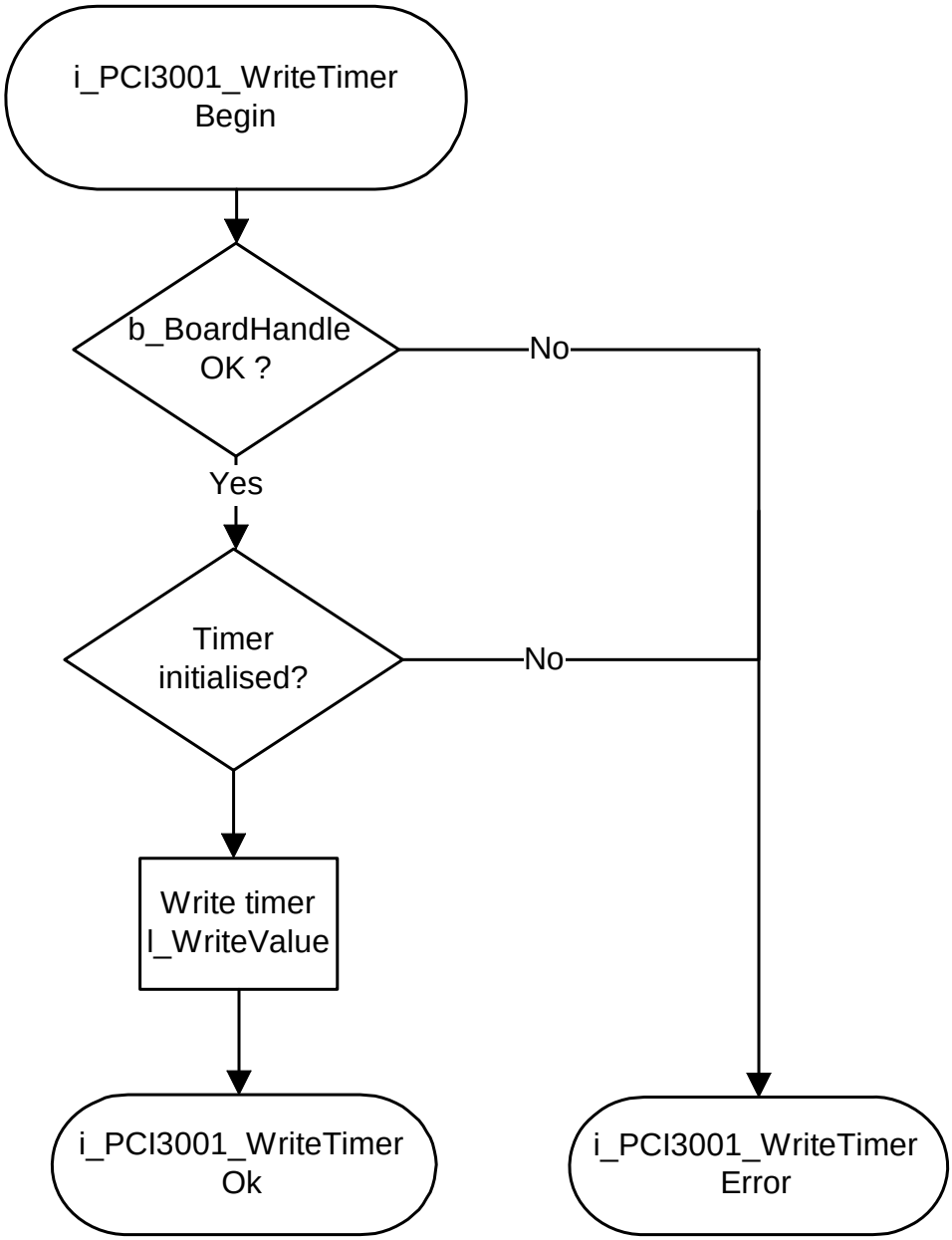
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PCI3001_WriteTimer (b_BoardHandle,
                                      1000);
```

Return value:
0: No error.
-1: The handle parameter of the board is wrong.
-2: Timer has not been initialised.

Input

b_BoardHandle l_WriteValue

Function diagram



Output

<Return Value>

3.6 Digital input

1) i_PCI3001_Read1DigitalInput (...)

Syntax:
<Return value> = i_PCI3001_Read1DigitalInput
(BYTE b_BoardHandle,
BYTE b_Channel,
PBYTE pb_ChannelValue)

Parameter:
- Input:

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

- Output:

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

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

Calling convention:
ANSI C :

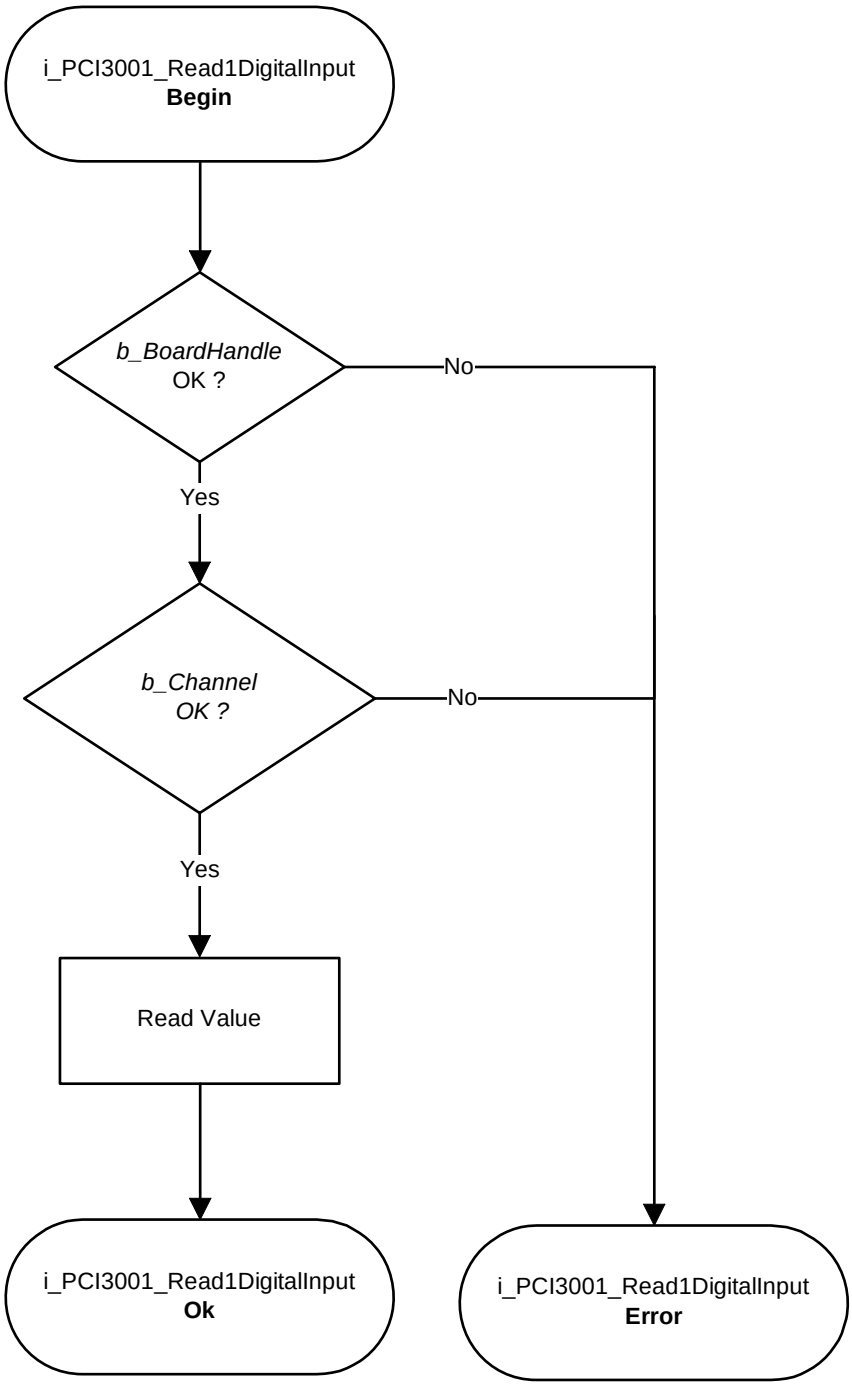
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   pb_ChannelValue;

i_ReturnValue = i_PCI3001_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.

Input

b_BoardHandle b_Channel

Function diagram



Output

pb_ChannelValue

<Return Value>

2) i_PCI3001_Read4DigitalInput (...)

Syntax:
<Return value> = i_PCI3001_Read4DigitalInput
(BYTE b_BoardHandle,
PBYTE pb_PortValue)

Parameter:
- 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 :

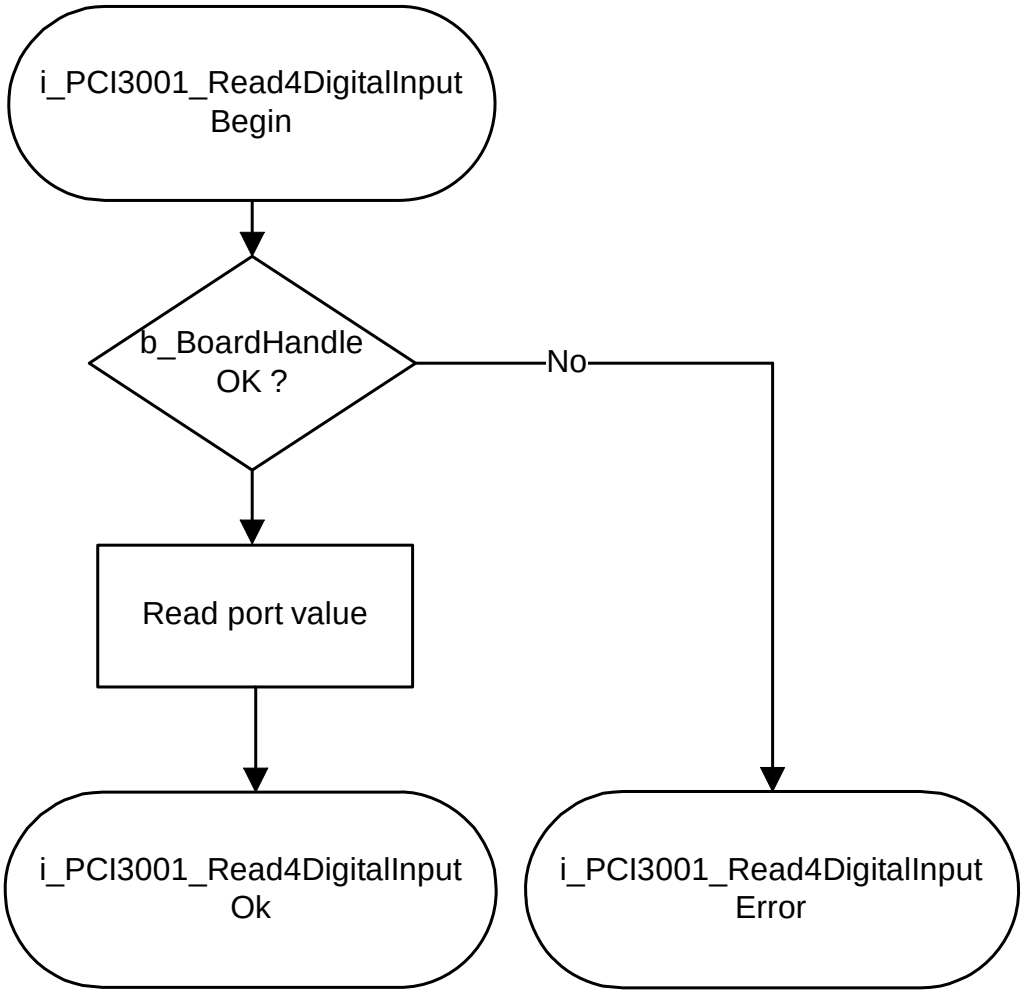
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   pb_PortValue;

i_ReturnValue = i_PCI3001_Read4DigitalInput     (b_BoardHandle,
                                                &pb_PortValue);
```

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

Input

b_BoardHandle

Function diagram



Output

pb_PortValue

<Return Value>

3.7 Digital output channels

1) i_PCI3001_Set1DigitalOutputOn (...)

Syntax:
<Return value> = i_PCI3001_Set1DigitalOutputOn
(BYTE b_BoardHandle,
BYTE b_Channel)

Parameter:
- Input:

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

- Output:
No output signal has occurred.

Task:
Sets the output which has been passed with the parameter b_Channel. Setting an output means setting an output high.

Calling convention:
ANSI C :

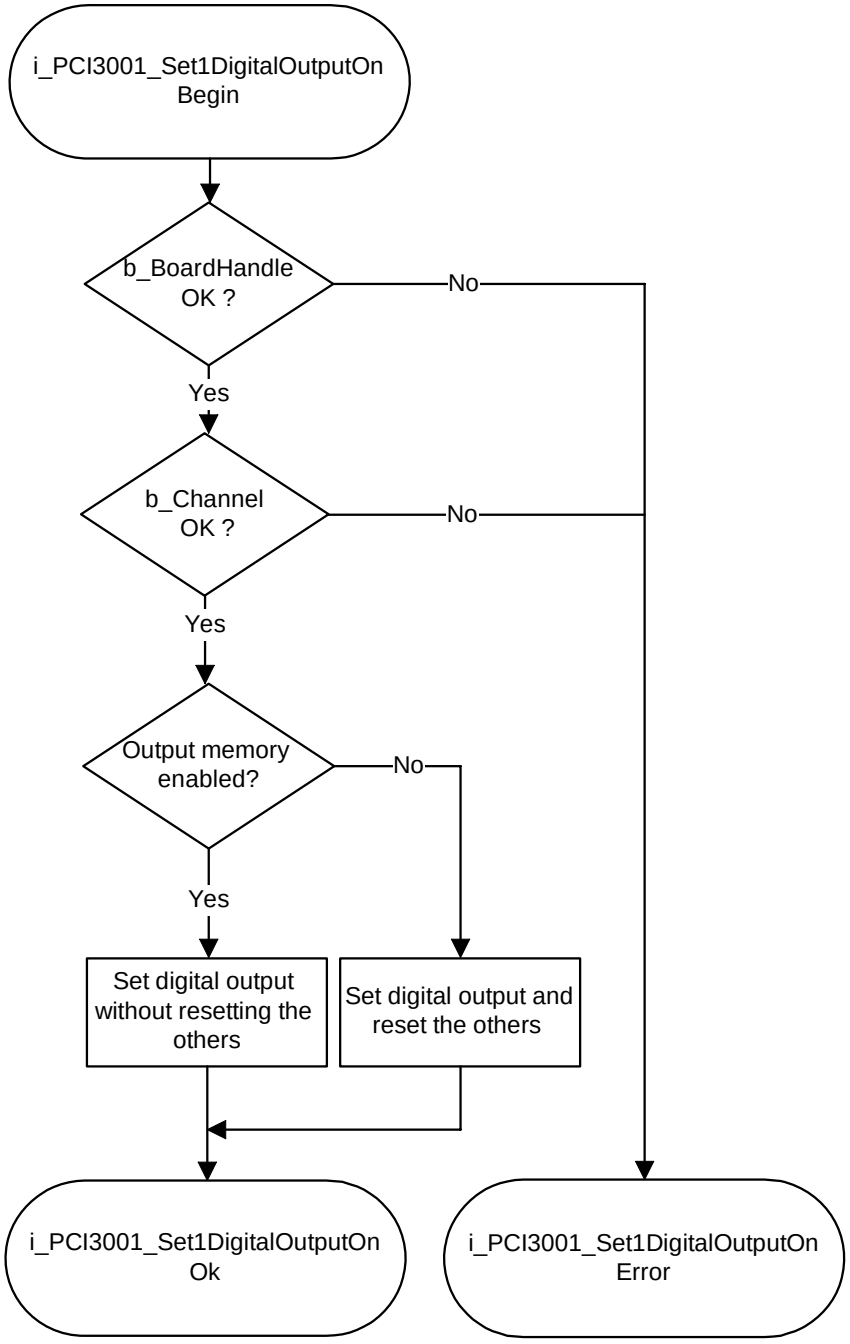
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;


i_ReturnValue = i_PCI3001_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.

Input

b_BoardHandle b_Channel

Function diagram



Output

<Return Value>

2) i_PCI3001_Set1DigitalOutputOff (...)

Syntax:
<Return value> = i_PCI3001_Set1DigitalOutputOff
(BYTE b_BoardHandle,
BYTE b_Channel)

Parameter:
- Input:

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

- Output:
No output signal has occurred.

Task:
Resets the output which has been passed with the parameter b_Channel. Resetting an output means setting an output low.

Calling convention:
ANSI C :

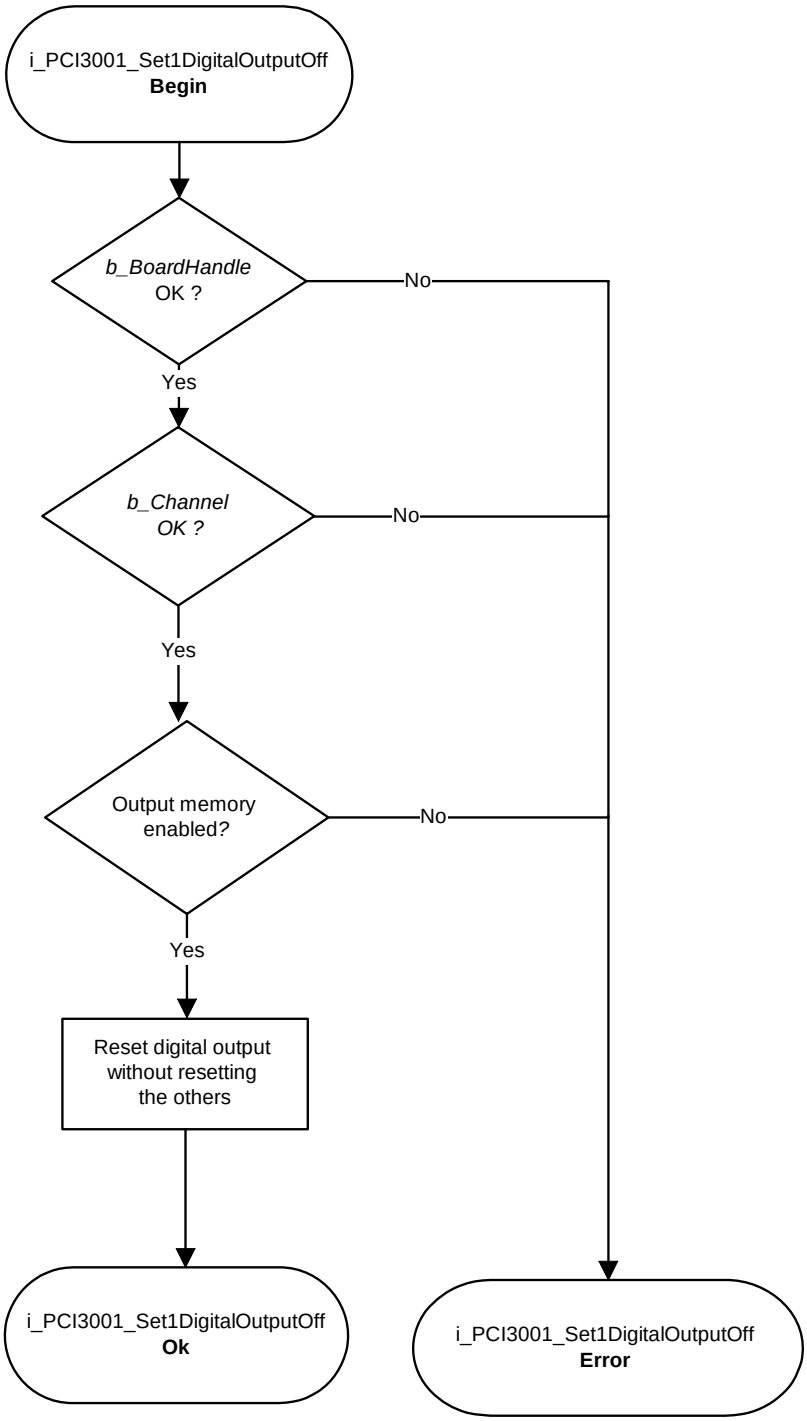
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;


i_ReturnValue = i_PCI3001_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 is OFF.
Use previously the function "i_PCI3001_SetOutputMemoryOn".

Input

b_BoardHandle b_Channel

Function diagram



Output

<Return Value>

3) i_PCI3001_Set4DigitalOutputOn (...)

Syntax:
<Return value> = i_PCI3001_Set4DigitalOutputOn
(BYTE b_BoardHandle,
BYTE b_Value)

Parameter:
- 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 outputs of a port. Setting an output means setting an output high. If you have switched OFF the digital output memory, all other outputs are set to "0".

Calling convention:
ANSI C :

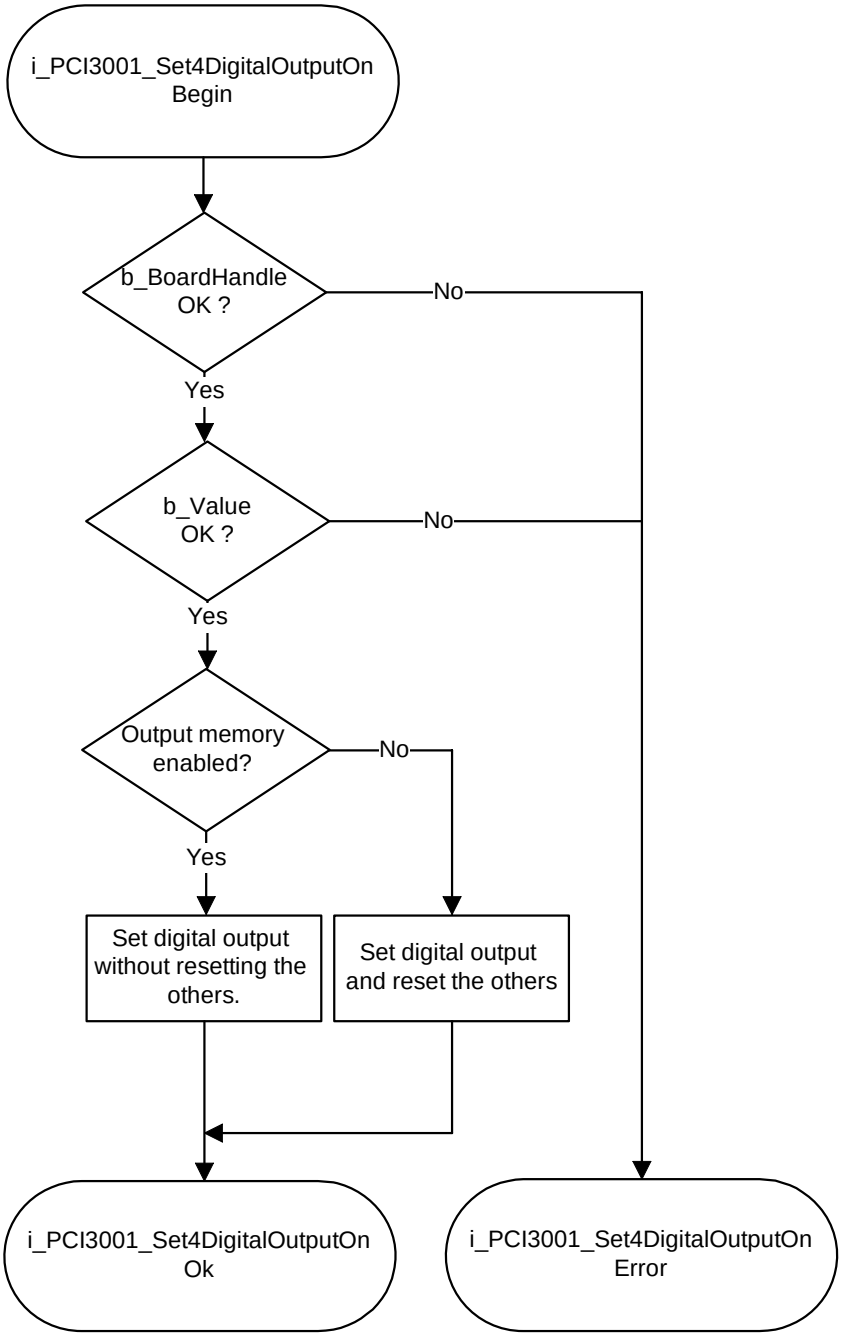
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;


i_ReturnValue = i_PCI3001_Set4DigitalOutputOn   (b_BoardHandle,
                                                 1);
```

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

Input

b_boardHandle b_Value

Function diagram



Output

<Return Value>

4) i_PCI3001_Set4DigitalOutputOff (...)

Syntax:
<Return value> = i_PCI3001_Set4DigitalOutputOff
(BYTE b_BoardHandle,
BYTE b_Value)

Parameter:
- 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 outputs of one port. Resetting means setting low.

Calling convention:
ANSI C :

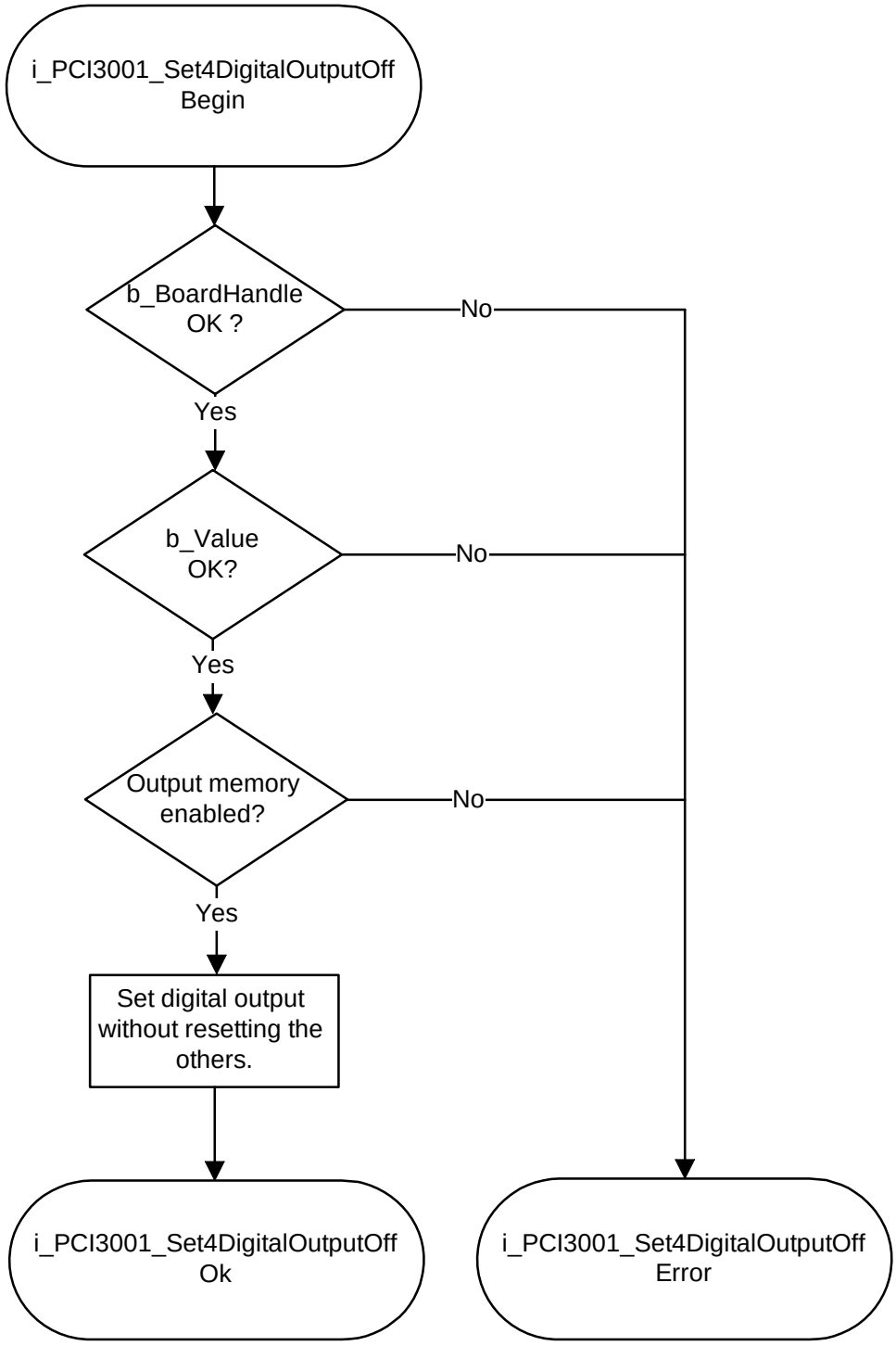
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;


i_ReturnValue = i_PCI3001_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_PCI3001_SetOutputMemoryOn".

Input

b_BoardHandle b_Value

Function diagram



Output

<Return Value>

5) i_PCI3001_SetOutputMemoryOn (...)

Syntax:
<Return value> = i_PCI3001_SetOutputMemoryOn
(BYTE b_BoardHandle)

Parameter:
- 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 outputs you have previously activated with the function "i_PCI3001_SetXDigitalOutputOn" are not reset. You can reset them with the function "i_PCI3001_SetXDigitalOutputOff".

Calling convention:
ANSI C :

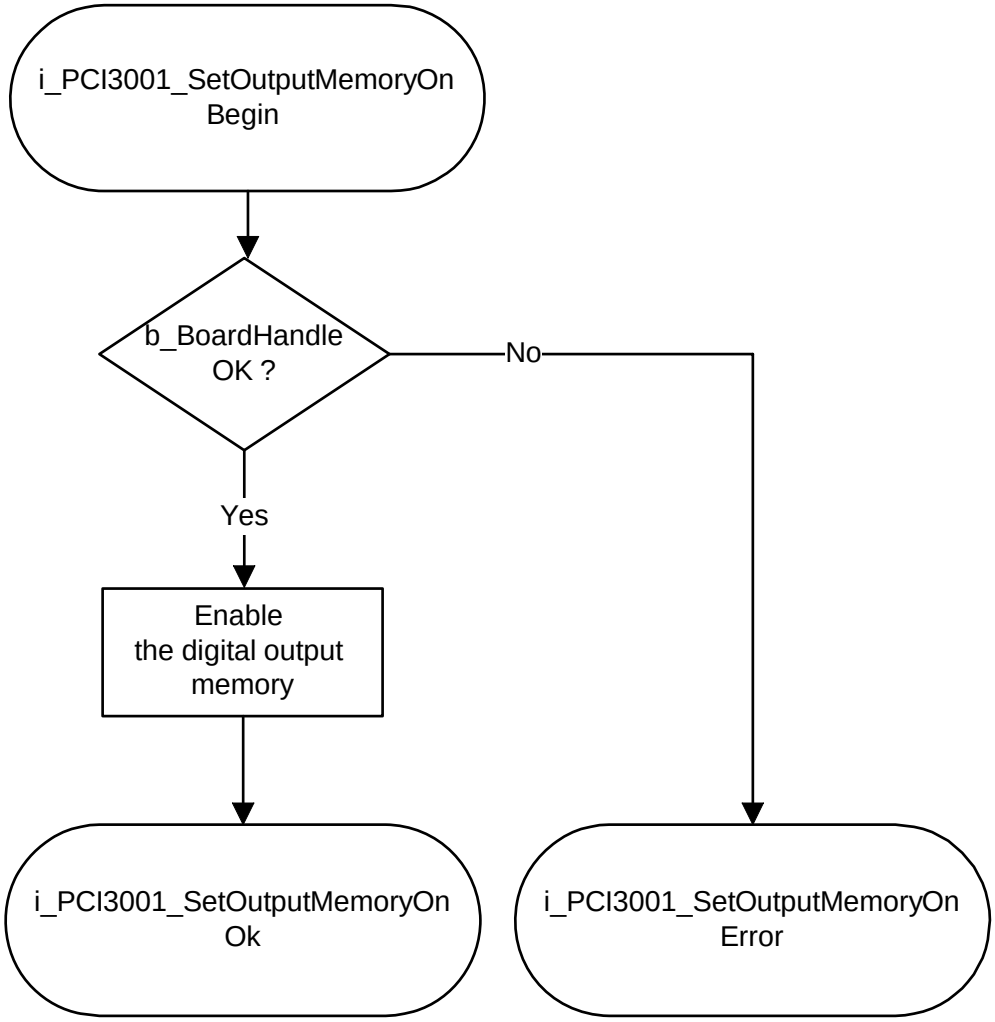
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;


i_ReturnValue = i_PCI3001_SetOutputMemoryOn   (b_BoardHandle);
```

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

Input

b_BoardHandle

Function diagram



Output

<Return Value>

6) i_PCI3001_SetOutputMemoryOff (...)

Syntax:
<Return value> = i_PCI3001_SetOutputMemoryOff
(BYTE b_BoardHandle)

Parameter:
- Input:
BYTE b_BoardHandle Handle of the board
- Output:
No output signal has occurred.

Task:
Deactivates the digital output memory.

Calling convention:
ANSI C :

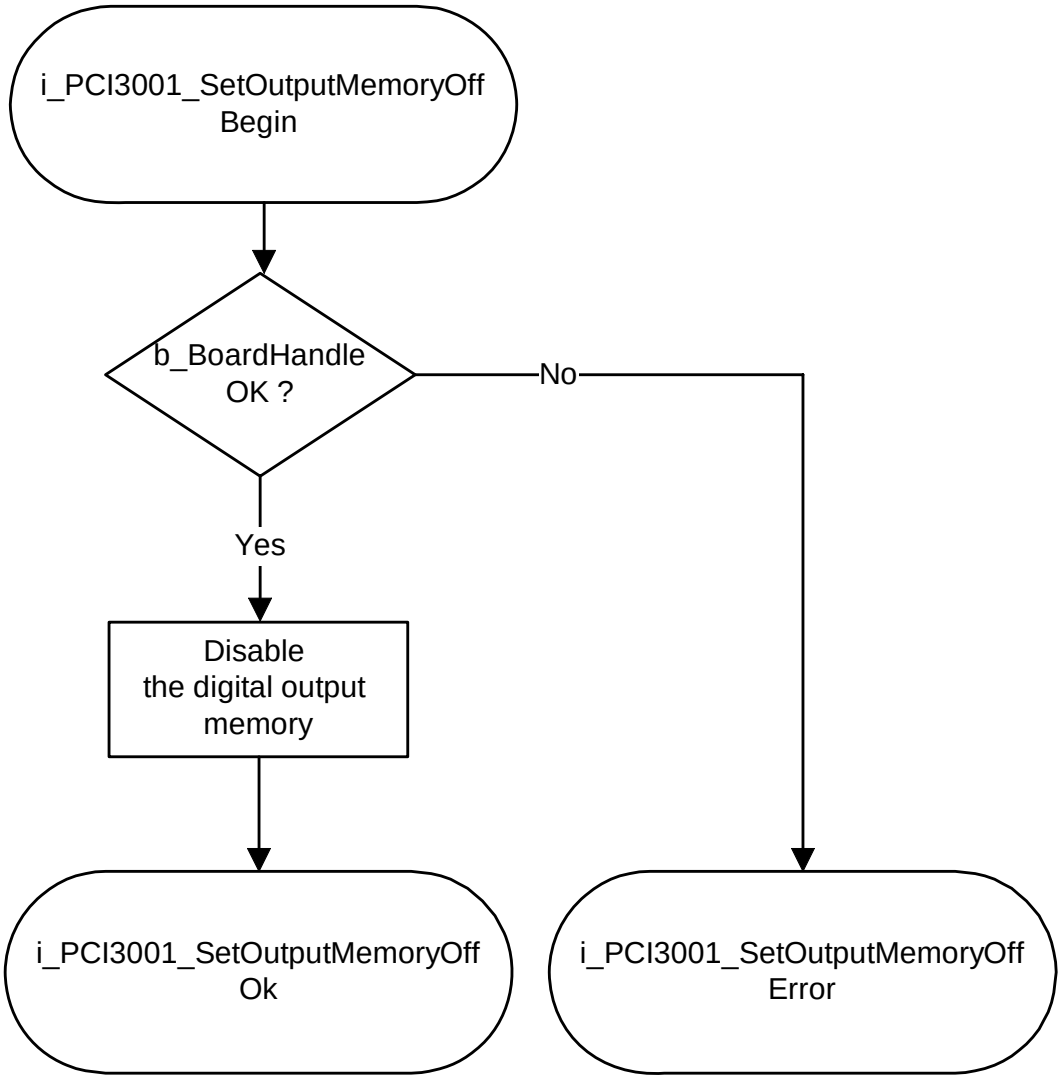
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;


i_ReturnValue = i_PCI3001_SetOutputMemoryOff  (b_BoardHandle);
```

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

Input

b_BoardHandle

Function diagram



Output

<Return Value>

3.8 Function to use in KERNEL Mode

1) i_PCI3001_KRNL_Read1DigitalInput (...)

Syntax:
<Return value> = i_PCI3001_KRNL_Read1DigitalInput
(UINT ui_Address,
BYTE b_Channel,
PBYTE pb_ChannelValue)

Parameter:
- Input:

UINT	ui_Address	Address of the board **xPCI-3001**
BYTE	b_Channel	The number of the input to be read (1 to 4).

- Output:

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

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

Calling convention:
ANSI C :

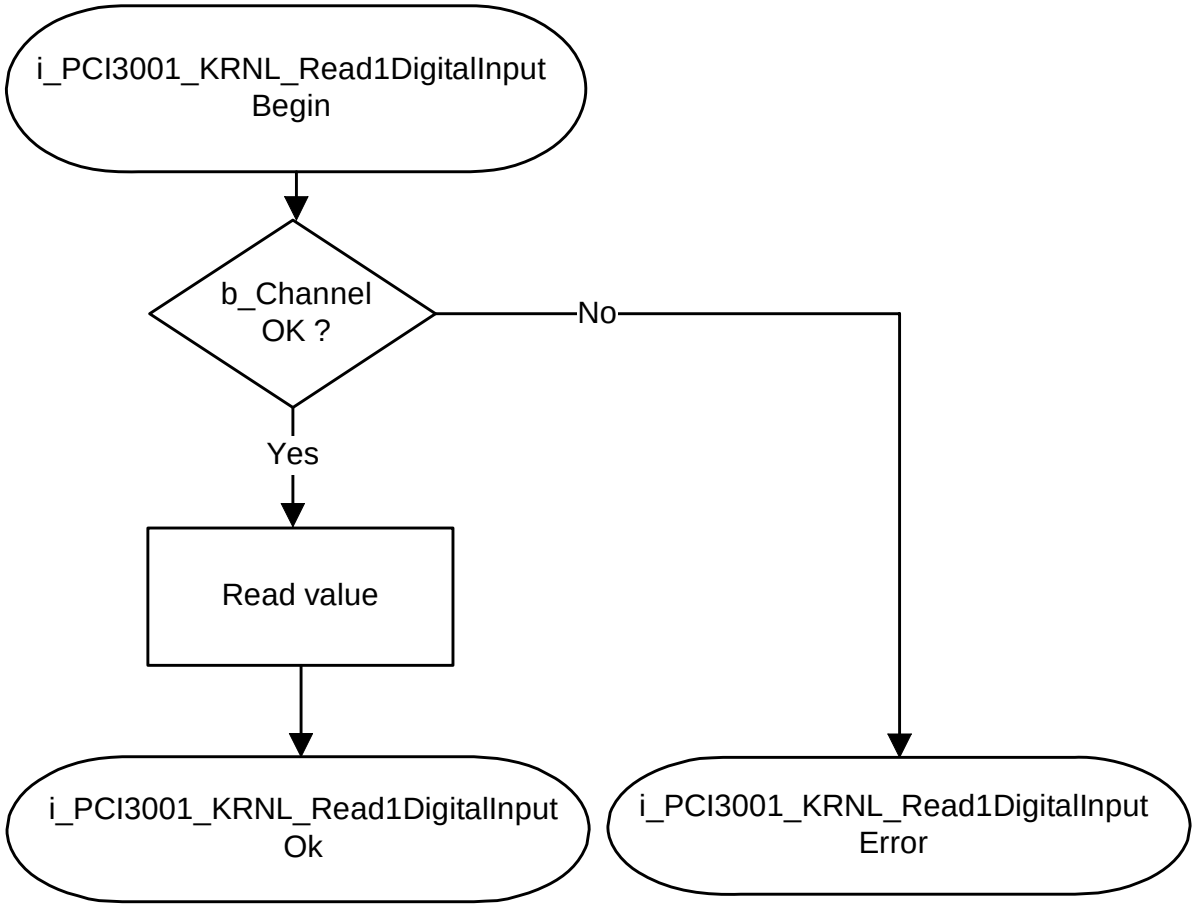
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   pb_ChannelValue;

i_ReturnValue = i_PCI3001_KRNL_Read1DigitalInput (0x390,
                                                  1,
                                                  &pb_ChannelValue);
```

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

Input

ui_Address b_Channel

Function diagram



Output

pb_ChannelValue

<Return Value>

2) i_PCI3001_KRNL_Read4DigitalInput (...)

Syntax:
<Return value> = i_PCI3001_KRNL_Read4DigitalInput
(UINT ui_Address,
PBYTE pb_PortValue)

Parameter:
- Input:

UINT	ui_Address	Address of the board **xPCI-3001**

- Output:

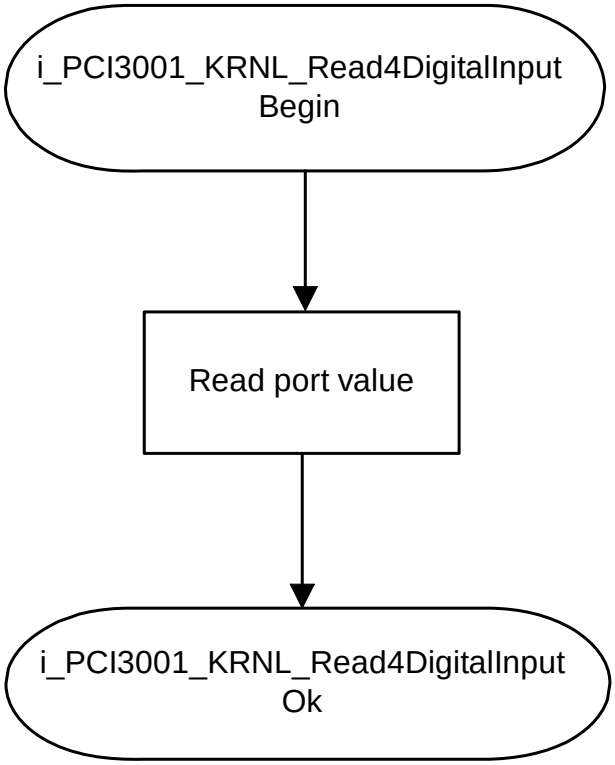
PBYTE	pb_PortValue	State of the digital input port (0 to 15)

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_PCI3001_Read4DigitalInput   (0x390,
                                               &pb_PortValue);
```

Return value:
0: No error.

Input
ui_Address
Function diagram

Output
pb_PortValue
<Return Value>

3) i_PCI3001_KRNL_Set1DigitalOutputOn (...)

Syntax:
<Return value> = i_PCI3001_KRNL_Set1DigitalOutputOn
(UINT ui_Address,
BYTE b_Channel)

Parameter:
- Input:

UINT	ui_Address	Address of the board **xPCI-3001**
BYTE	b_Channel	The number of the output to set (1 to 4).

- Output:
No output signal has occurred.

Task:
Sets the output which has been passed with the parameter b_Channel. Setting an output means setting an output high.

Calling convention:
ANSI C :

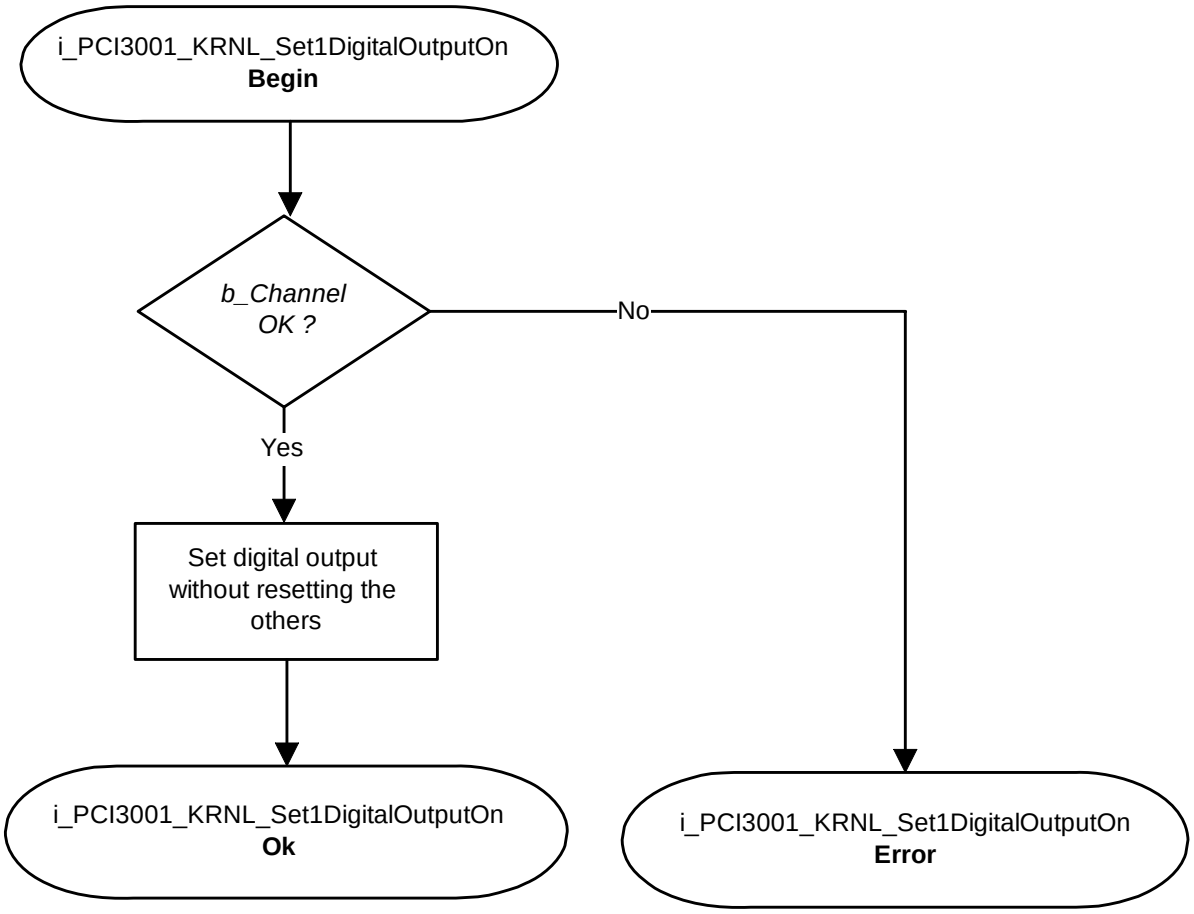
```
int             i_ReturnValue;

i_ReturnValue = i_PCI3001_KRNL_Set1DigitalOutputOn  (0x390,
                                                     1);
```

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

Input

ui_Address b_Channel

Function diagram



Output

<Return Value>

4) i_PCI3001_KRNL_Set4DigitalOutputOn (...)

Syntax:
<Return value> = i_PCI3001_KRNL_Set4DigitalOutputOn
(UINT ui_Address,
BYTE b_Value)

Parameter:
- Input:

UINT	ui_Address	Address of the board **xPCI-3001**
BYTE	b_Value	Output value (0 to 15)

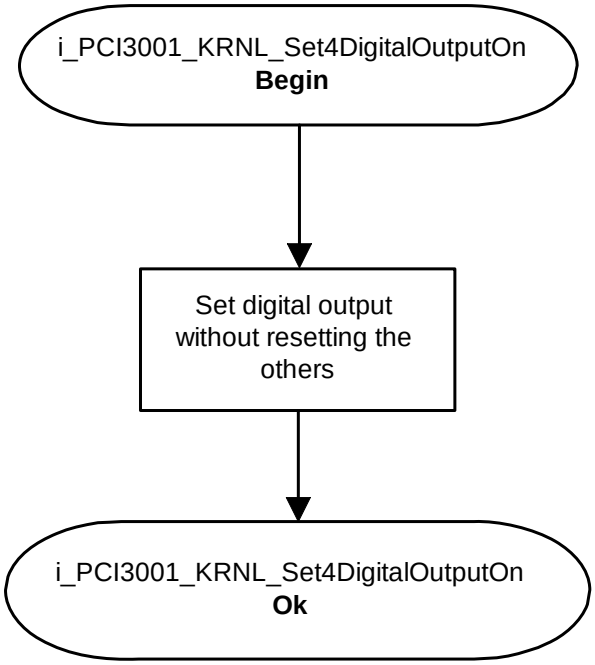
- Output:
No output signal has occurred.

Task:
Sets one or several outputs of a port. Setting an output means setting an output high. All other outputs are set to "0".

Calling convention:
ANSI C :

```
int             i_ReturnValue;


i_ReturnValue = i_PCI3001_KRNL_Set4DigitalOutputOn  (0x390,
                                                     1);
```

Return value:
0: No error.

Input
ui_Address b_Value
Function diagram

Output
<Return Value>