ADDI-DATA®

Technichal support:
+49 7229 1847-0

CE

Technical description

ADDIALOG PA 311

Standard software

Edition: 11.01-07/2005

Tables

1 INTRODUCTION

IMPORTANT!
Note the following conventions in the text:

Function: "i_PA 311_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

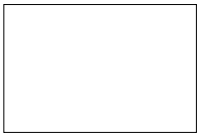
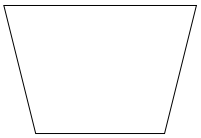
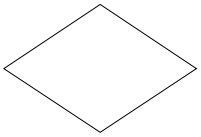
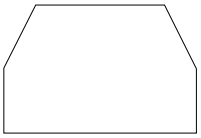
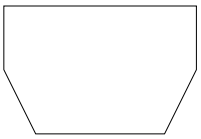
Define name	Decimal value	Hexadecimal value
DLL_COMPILER_C	0	0
DLL_COMPILER_VB	1	1
DLL_COMPILER_PASCAL	2	2
DLL_LABVIEW	3	3
DLL_COMPILER_VB_5	4	4
PA311_ENABLE	1	1
PA311_DISABLE	0	0
PA311_1_GAIN	1	1
PA311_2_GAIN	2	2
PA311_10_GAIN	4	4
PA311_USER_GAIN	8	8
PA311_UNIPOLAR	1	1
PA311_BIPOLAR	0	0
PA311_SIMPLE_MODUS	0	0
PA311_DELAY_MODUS	1	1
PA311_DMA_NOT_USED	0	0
PA311_DMA_USED	1	1
PA311_SINGLE	0	0
PA311_CONTINUOUS	1	1

2 DIN 66001- GRAPHICAL SYMBOLS

This chapter describes all software functions (API) necessary for the operation of the **PA 311** board.

To illustrate these functions, we designed flow charts with the graphical symbols listed below.

It gives the user a quick overview of the hierarchy between the different functions.

	**Process,** general (including inputs and outputs)
	**Manual operation** (including inputs and outputs)
	**Decision** **Selection unit** (eg.: switch)
	**Loop limit** Beginning
	**Loop limit** End

3 SOFTWARE FUNCTIONS (API)

3.1 Initialisation

1) i_PA311_InitCompiler (..)

Syntax:
<Return value> = i_PA311_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. - DLL_COMPILER_PASCAL: The user programs in Pascal or Delphi. - DLL_LABVIEW : The user programs in Labview.

- Output:
No output signal has occurred

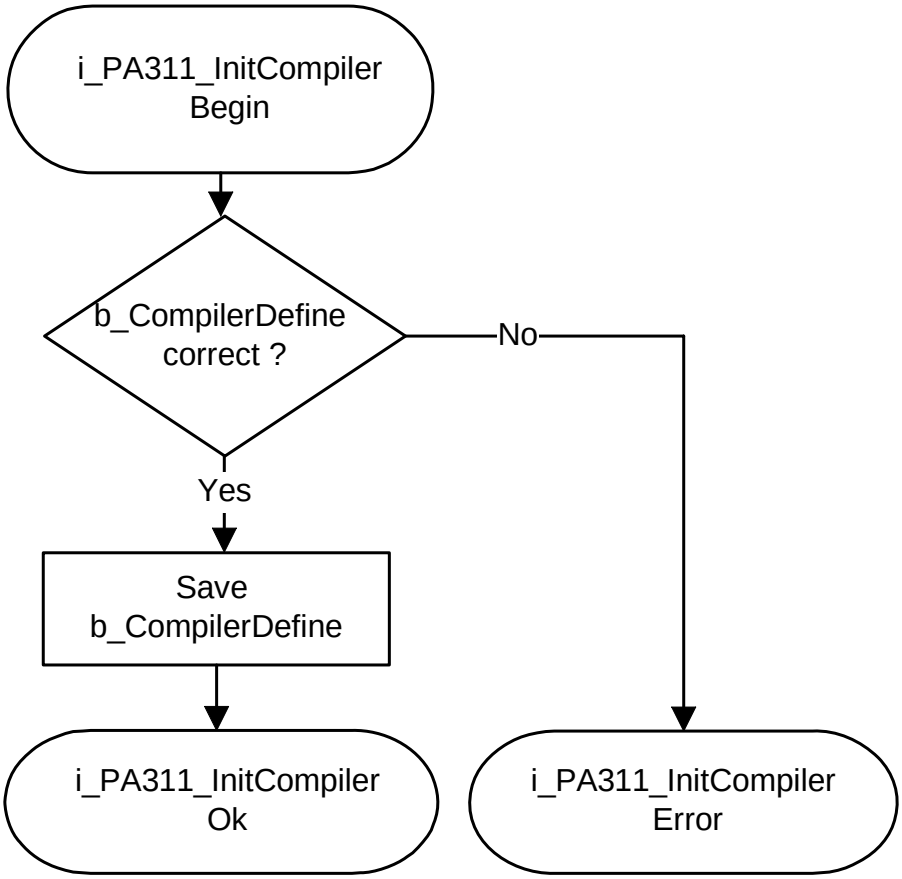
Task:
If you want to use the DLL functions, choose the language in which you want to program. This function must be the first to be called up.

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

Calling convention:
ANSI C :

```
int i_ReturnValue;

i_ReturnValue = i_PA311_InitCompiler (DLL_COMPILER_C);
```

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

Input
b_Compilerdefine
Function diagram

Output
<Return Value>

2) i_PA311_SetBoardInformation (...)

Syntax:

```
<Return value> =  i_PA311_SetBoardInformation
                                (UINT    ui_BaseAddress,
                                 BYTE    b_EOCInterruptNbr,
                                 BYTE    b_Timer2InterruptNbr,
                                 BYTE    b_PIOInterruptNbr,
                                 BYTE    b_DMAInterruptNbr,
                                 BYTE    b_DMAChannel,
                                 BYTE    b_AnalogInputResolution,
                                 BYTE    b_AnalogInputChannelNbr,
                                 BYTE    b_AnalogOutputResolution,
                                 BYTE    b_AnalogOutputChannelNbr,
                                 PBYTE   pb_BoardHandle)
```

Parameter:

- Input:

UINT	ui_BaseAddress	Base address of **PA311** board
BYTE	b_EOCInterruptNbr	Interrupt for the end of conversion (3,5,9,10,11,12,14 or 15). If 0, no interrupt is used
BYTE	b_Timer2InterruptNbr	Timer 2 interrupt. (3,5,9,10,11,12,14 or 15). If 0, no interrupt is used
BYTE	b_PIOInterruptNbr	PC3 of 82C55A interrupt line (3,5,9,10,11,12,14 or 15). If 0, no interrupt is used
BYTE	b_DMAInterruptNbr	Terminal count interrupt of DMA (3,5,9,10,11,12,14 or 15). If 0, no interrupt is used
BYTE	b_DMAChannel	Number of the DMA channel (5,6 or 7). If 0, no DMA is used
BYTE	b_AnalogInputResolution	Analog input resolution 14 : 14-bit resolution 16 : 16-bit resolution
BYTE	b_AnalogInputChannelNbr	Number of analog inputs (8 or 16)
BYTE	b_AnalogOutputResolution	Analog output resolution 14 : 14-bit resolution 16 : 16-bit resolution
BYTE	b_AnalogOutputChannelNbr	Number of analog outputs (4 or 8)

- Output:

PBYTE	pb_BoardHandle	Handle[1] of board **PA311** to use the functions

[1] Identification number of the board

Task:
Verifies if board **PA311** is present. Stores the following information:
- the base address,
- the interrupt of EOC, Timer2, PIO and terminal count,
- the number of analog inputs
- and the number of analog outputs.

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

Calling convention:
ANSI C :

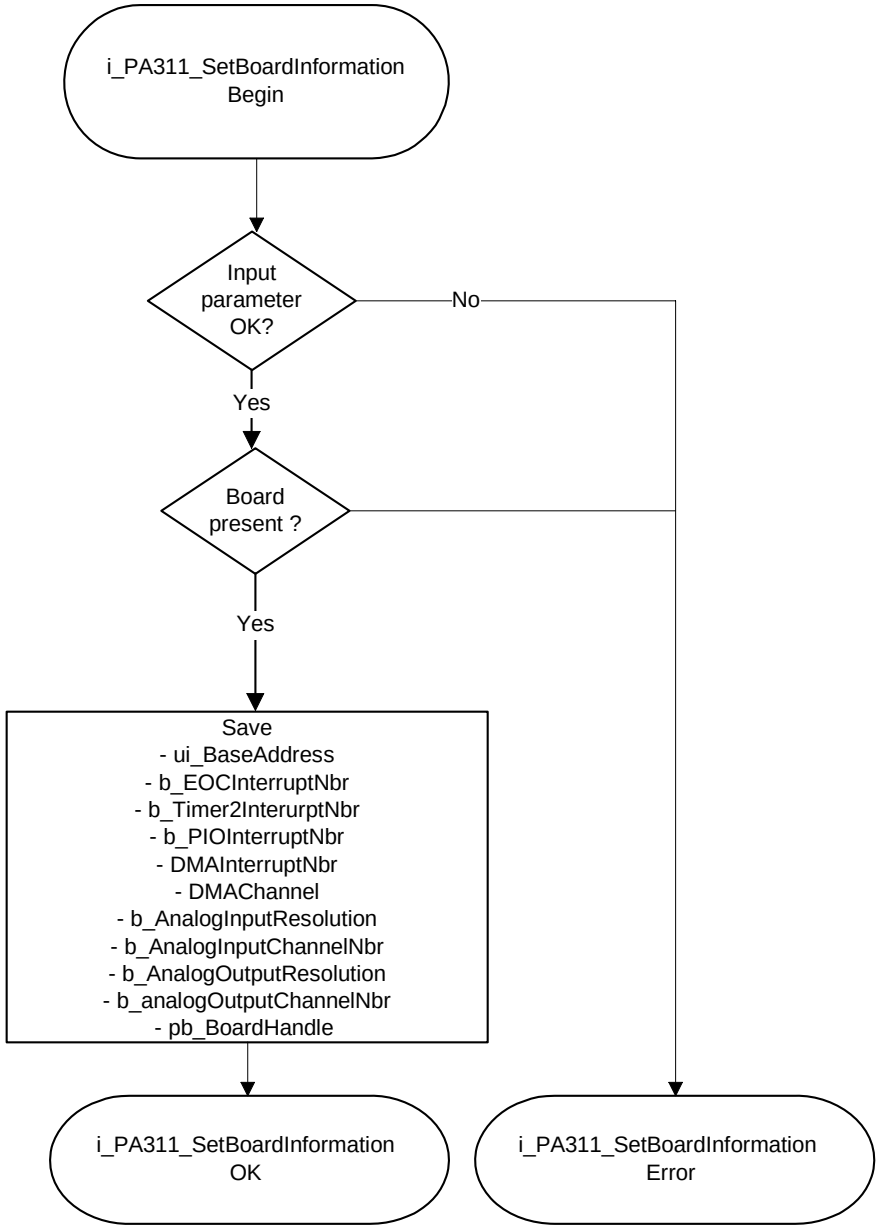
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PA311_SetBoardInformation (0x300, 0, 0, 0,
                                             0, 0, 14, 16, 14,
                                             8, &b_BoardHandle);
```

Return value:
 0 : No error
-1 : Board not present
-2 : EOC interrupt selection is wrong
-3 : Timer 2 interrupt selection is wrong
-4 : PIO interrupt selection is wrong
-5 : Terminal count (DMA) interrupt selection is wrong
-6 : Two interrupts cannot be equal
-7 : One or several interrupts are already used by another board
-8 : DMA channel selection is wrong
-9 : Analog input resolution is wrong
-10: Number of analog inputs is wrong.
-11: Analog output resolution is wrong
-12: Number of analog outputs is wrong.
-13: No handle is available for the board (up to 10 handles can be used)

Input

ui_BaseAddress	b_AnalogInputResolution
b_EOCInterruptNbr	b_AnalogInputchannelNbr
b_Timer2InterruptNbr	b_AnalogOutputResolution
b_PIOInterruptNbr	b_AnalogOutputChannelNbr
b_DMAInterruptNbr	pb_BoardHandle
b_DMAChannel	

Function diagram



Output

pb_BoardHandle

<Return Value>

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

3) i_PA311_SetBoardInformationWin32 (...)

Syntax:

```
<Return value> =  i_PA311_SetBoardInformationwin32
                                    (PCHAR    pc_Identifier,
                                     BYTE     b_AnalogInputResolution,
                                     BYTE     b_AnalogInputChannelNbr,
                                     BYTE     b_AnalogOutputResolution,
                                     BYTE     b_AnalogOutputChannelNbr,
                                     PBYTE    pb_BoardHandle)
```

Parameter:

- Input:

PCHAR	pc_Identifier	Identifier string for the selection of the PA 311 board. The identifier string is determined with the ADDIREG registration program
BYTE	b_AnalogInputResolution	Analog input resolution 14 : 14-bit resolution 16 : 16-bit resolution
BYTE	b_AnalogInputChannelNbr	Number of analog inputs (8 or 16)
BYTE	b_AnalogOutputResolution	Analog output resolution 14 : 14-bit resolution 16 : 16-bit resolution
BYTE	b_AnalogOutputChannelNbr	Number of analog outputs (4 or 8)

- Output:

PBYTE	pb_BoardHandle	Handle of board **PA311** to use the functions

Task:

Calls all hardware information about the **PA311** included in the ADDIREG registration program and stores the following information:
- the base address
- the interrupt of EOC, Timer2, PIO and terminal count,
- the number of analog inputs
- and the number of analog outputs.

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_PA311_SetBoardInformationWin32
                                        ("PA311-00" 14, 16,
                                         14, 8, &b_BoardHandle);
```

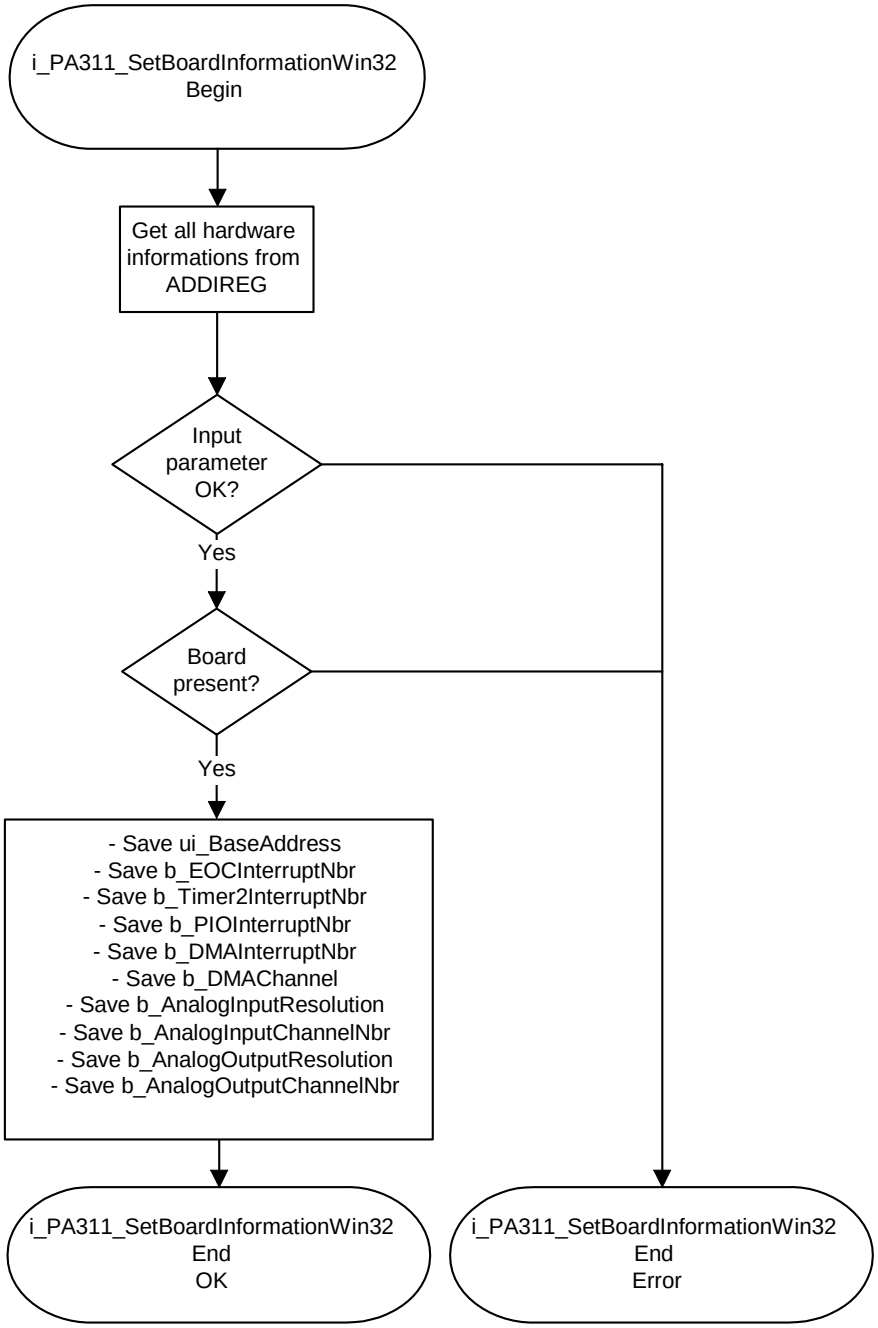
Return value:

0: No error
-1: Board not present
-2: One or several interrupts are already used by another board
-3: Analog input resolution is wrong
-4: Number of analog inputs is wrong.
-5: Analog output resolution is wrong
-6: Number of analog outputs is wrong.
-7: No handle is available for the board (up to 10 handles can be used)
-8: Error by opening the driver under Windows NT/95

Input

pc_Identifier	b_AnalogOutputResolution
b_AnalogInputResolution	b_AnalogOutputChannelNbr
b_AnalogInputChannelNbr	pb_BoardHandle

Function diagram



Output

pb_BoardHandle

<Return Value>

4) i_PA311_GetHardwareInformation (...)

Syntax:

```
<Return value> =  i_PA311_GetHardwareInformation
                                    (BYTE     b_BoardHandle,
                                     PUINT    pui_BaseAddress,
                                     PBYTE    pb_EOCInterruptNbr,
                                     PBYTE    pb_Timer2InterruptNbr,
                                     PBYTE    pb_PIOInterruptNbr,
                                     PBYTE    pb_DMAInterruptNbr,
                                     PBYTE    pb_DMAChannel)
```

Parameter:

- Input:

BYTE	b_BoardHandle	Handle of board **PA311**

- Output:

PUINT	pui_BaseAddress	**PA311** base address
PBYTE	pb_EOCInterruptNbr	Interrupt for End of conversion (3, 5, 9, 10, 11, 12, 14 or 15). If 0, no interrupt is used
PBYTE	pb_Timer2InterruptNbr	Timer 2 interrupt. (3, 5, 9, 10, 11, 12, 14 or 15). If 0, no interrupt is used
PBYTE	pb_PIOInterruptNbr	PC3 of 82C55A interrupt line (3, 5, 9, 10, 11, 12, 14 or 15). If 0, no interrupt is used
PBYTE	pb_DMAInterruptNbr	Terminal count interrupt of DMA (3, 5, 9, 10, 11, 12, 14 or 15). If 0, no interrupt is used
PBYTE	pb_DMAChannel	Number of the DMA channel (5,6 or 7). If 0, no DMA is used

Task:

Returns the base address, the interrupts and the DMA channel from the **PA311**.

Calling convention:

ANSI C :

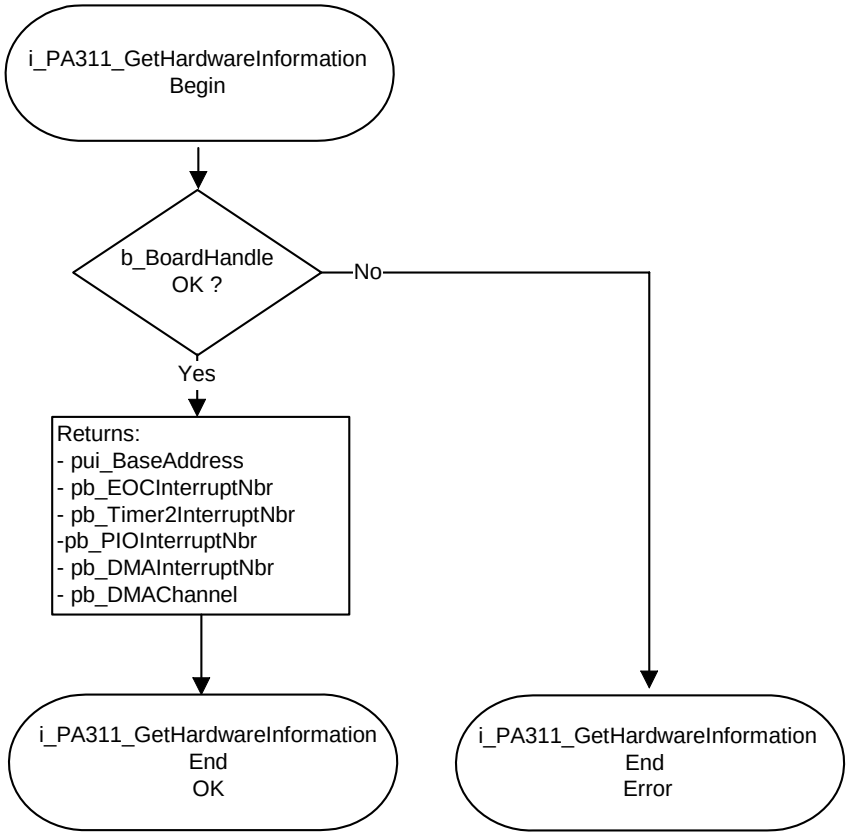
```
int             i_ReturnValue;
unsigned int    ui_BaseAddress;
unsigned char   b_InterruptNbr;
unsigned char   b_SlotNumber;
unsigned char   b_BoardHandle;
unsigned char   b_EOCInterruptNbr;
unsigned char   b_Timer2InterruptNbr;
unsigned char   b_PIOInterruptNbr;
unsigned char   b_DMAInterruptNbr;
unsigned char   b_DMAChannel;
```

```
i_ReturnValue = i_PA311_GetHardwareInformation (b_BoardHandle,
                                                &ui_BaseAddress,
                                                &b_EOCInterruptNbr,
                                                &b_Timer2InterruptNbr,
                                                &b_PIOInterruptNbr,
                                                &b_DMAInterruptNbr,
                                                &b_DMAChannel);
```

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

Input	
b_BoardHandle	pb_PIOInterruptNbr
pui_BaseAddress	pb_DMAInterruptNumber
pb_EOCInterruptNbr	pb_DMAChannel
pb_Timer2InterruptNbr	

Function diagram



Output	
pui_BaseAddress	pb_PIOInterruptNumber
pb_EOCInterruptNumber	pb_DMAInterruptNumber
pb_Timer2InterruptNumber	pb_DMAChannel

<Return Value>

5) i_PA311_CloseboardHandle (..)

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

Syntax:
<Return value> = i_PA311_CloseBoardHandle (BYTE b_BoardHandle)

Parameter:
- Input:
BYTE b_BoardHandle Handle of board **PA311**
- Output:
No output signal has occurred.

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

Calling convention:
ANSI C :

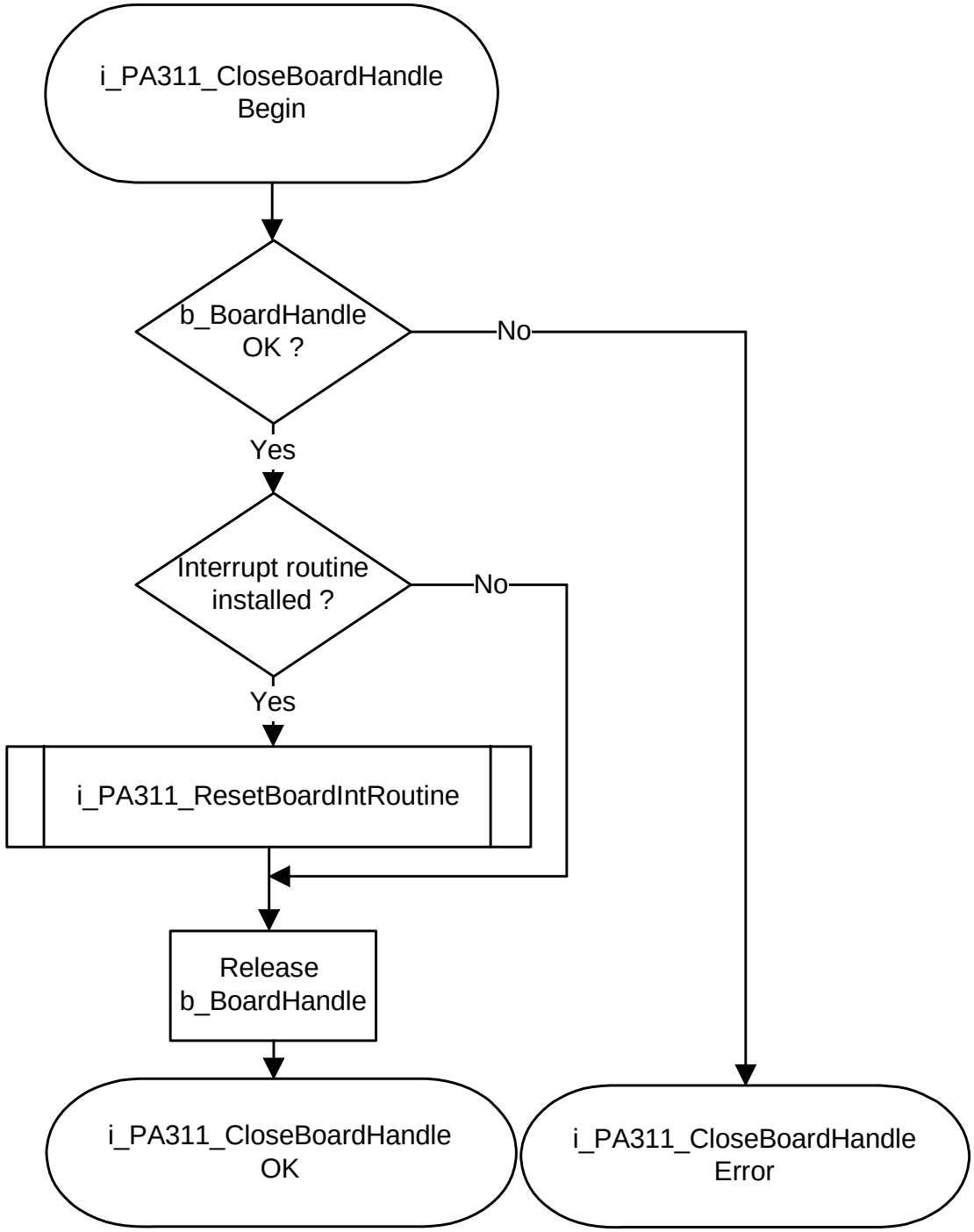
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PA311_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_PA311_SetBoardIntRoutineDos (..)

Syntax:

```
<Return value> = i_PA311_SetBoardIntRoutineDos
                                    (BYTE     b_BoardHandle,
                                     VOID     v_FunctionName
                                              (BYTE   b_BoardHandle,
                                               BYTE   b_InterruptMask,
                                               PUINT  pui_AnalogInputValue))
```

Parameter:

- Input

BYTE	b_BoardHandle	Handle of board **PA 311**
VOID	v_FunctionName	Name of the user interrupt routine

- Output:

No output signal has occurred.

Task:

This function must be called up each time you want to enable an interrupt action on a **PA 311**. It installs one user interrupt function on all boards on which you have enabled an interrupt.

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

If you operate several boards **PA311** which have to react to interrupts, call up the function as often as you operate boards **PA311**. The variable *v_FunctionName* is only relevant **for the first callup.**

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 **PA311** 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
00**1**0 0000	PIO interrupt

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_PA311_SetBoardIntRoutineDos    (b_BoardHandle,
                                                  v_FunctionName );
```

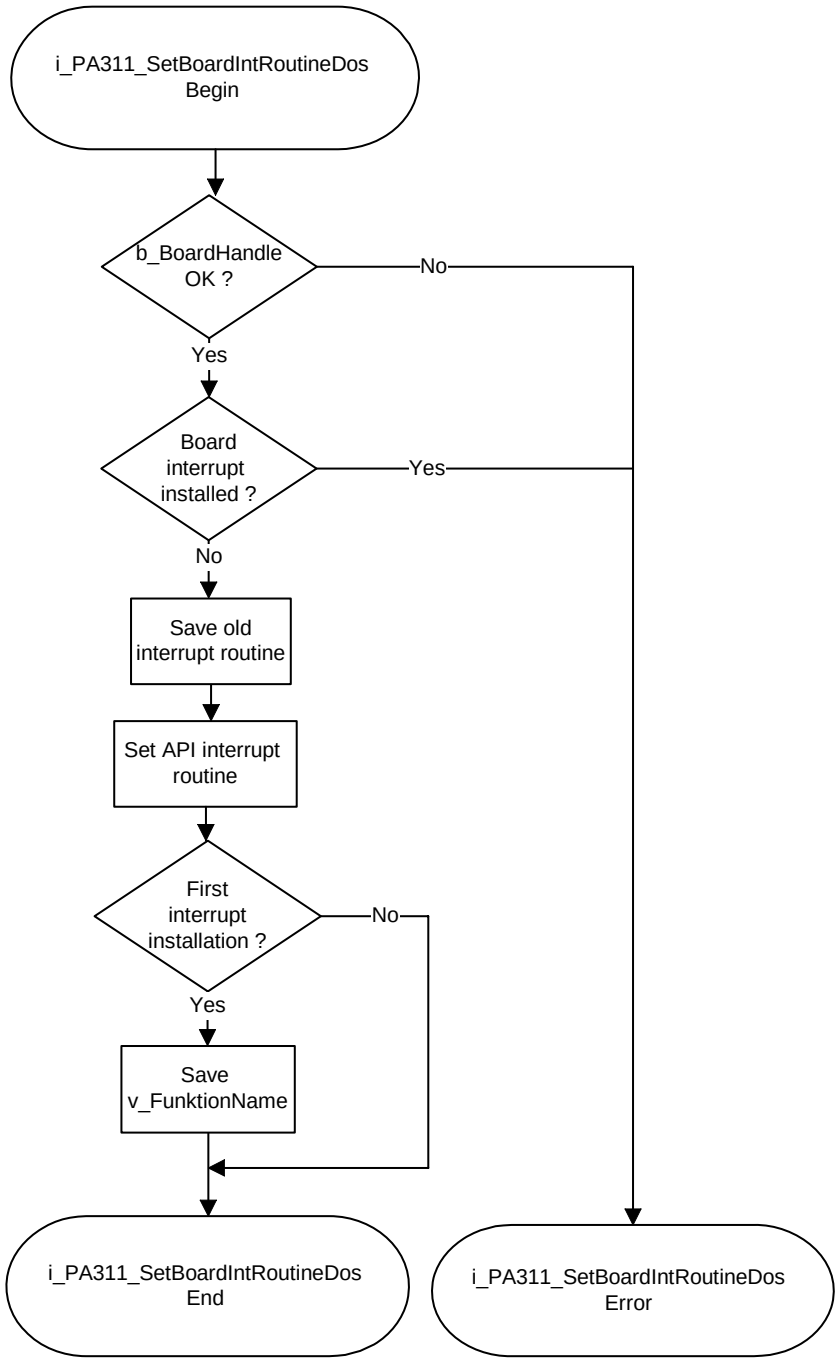
Return value:

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_PA311_SetBoardIntRoutineVBDos (..)

Syntax:
<Return value> = i_PA311_SetBoardIntRoutineVBDos
(BYTE b_BoardHandle)

Parameter:
- Input:
BYTE b_BoardHandle Handle of board **PA311**
- Output:
No output signal has occured.

Task:
This function must be called up each time you want to enable an interrupt action on a **PA 311**. It installs one user interrupt function on all boards on which you have enabled an interrupt.

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

If you operate several boards **PA311** which have to react to interrupts, call up the function as often as you operate boards **PA311**.

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

Controlling the interrupt management
Please use instead the following functions

"ON UEVENT GOSUB xxxxxxxxx" of Visual Basic for DOS

and

"i_PA311_TestInterrupt"

This function tests the interrupt of board **PA311**. It is used for obtaining 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_PA311_TestInterrupt       (i_BoardHandle, _
                                             i_InterruptMask,  _
                                             l_AnalogInputValue (0) )
.
.
.

Return


ON UEVENT GOSUB IntLabel
UEVENT ON
i_ReturnValue = i_PA311_SetBoardIntRoutineVBDos       (b_BoardHandle)
```

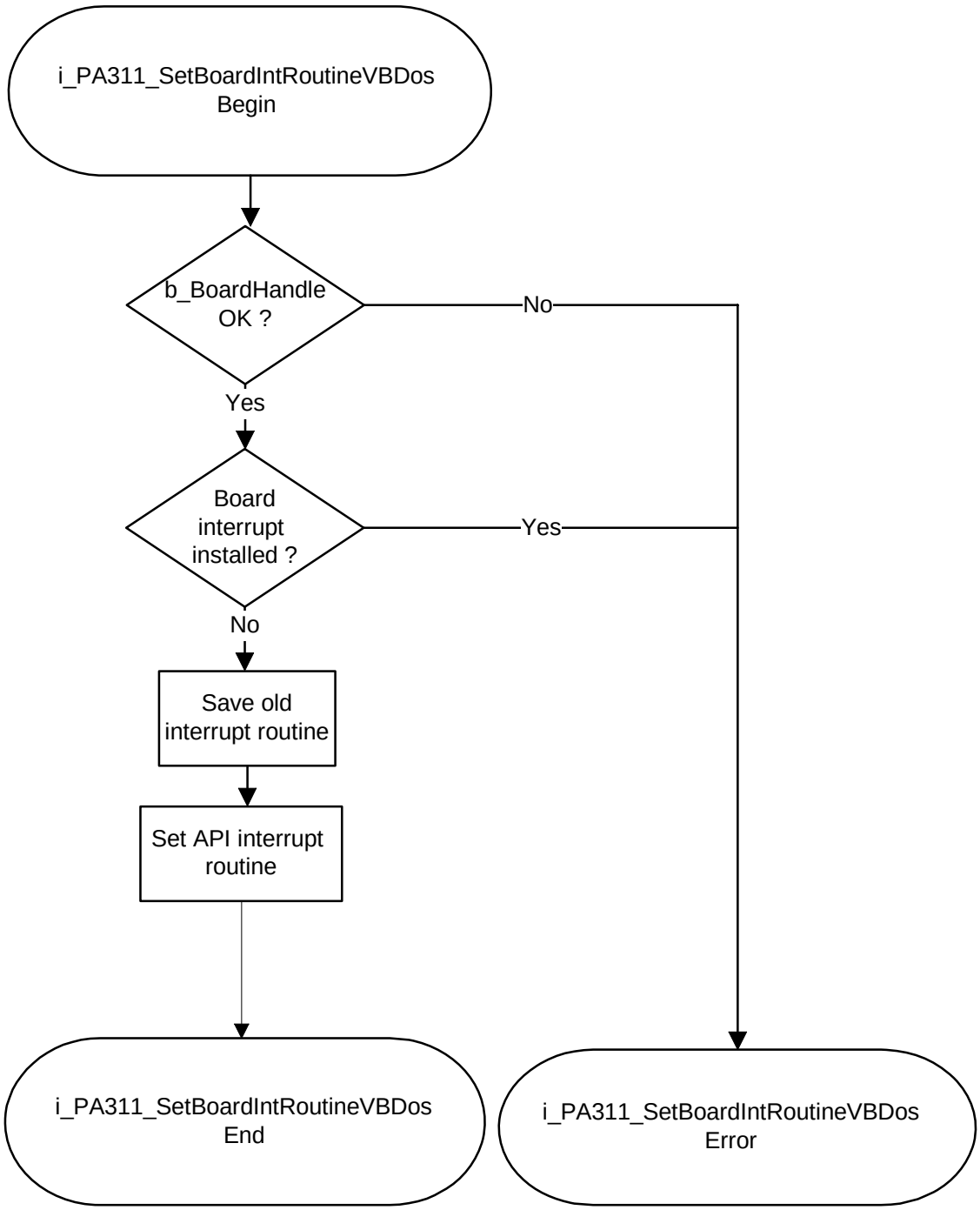
Return value:

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_PA311_SetBoardIntRoutineWin16 (..)

Syntax:
<Return value> = i_PA311_SetBoardIntRoutineWin16
(BYTE b_BoardHandle,
VOID v_FunctionName (BYTE b_BoardHandle,
BYTE b_InterruptMask,
PUINT pui_AnalogInputValue))

Parameter:
- Input:

BYTE	b_BoardHandle	Handle of board **PA311**
VOID	v_FunctionName	Name of the user interrupt routine

- Output:
No output signal has occurred.

Task:
This function must be called up each time you want to enable an interrupt action on a **PA 311**. It installs one user interrupt function on all boards on which you have enabled an interrupt.

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

If you operate several boards **PA311** which have to react to interrupts, call up the function as often as you operate boards **PA311**. The variable *v_FunctionName* is only relevant **for the first callup**.

From the second call of the function (next board):
- interrupts are allowed.

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 (BYTEb_BoardHandle,
                        BYTE        b_InterruptMask,
                        PUINT       pui_AnalogInputValue)
```

v_FunctionName	Name of the user interrupt routine
b_BoardHandle	Handle of the **PA311** 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 have no signification. Please use the „i_PA311_TestInterrupt“ function.

```
VOID    v_FunctionName (BYTEb_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_PA311_SetBoardIntRoutineWin16 (b_BoardHandle,
                                                 v_FunctionName );
```

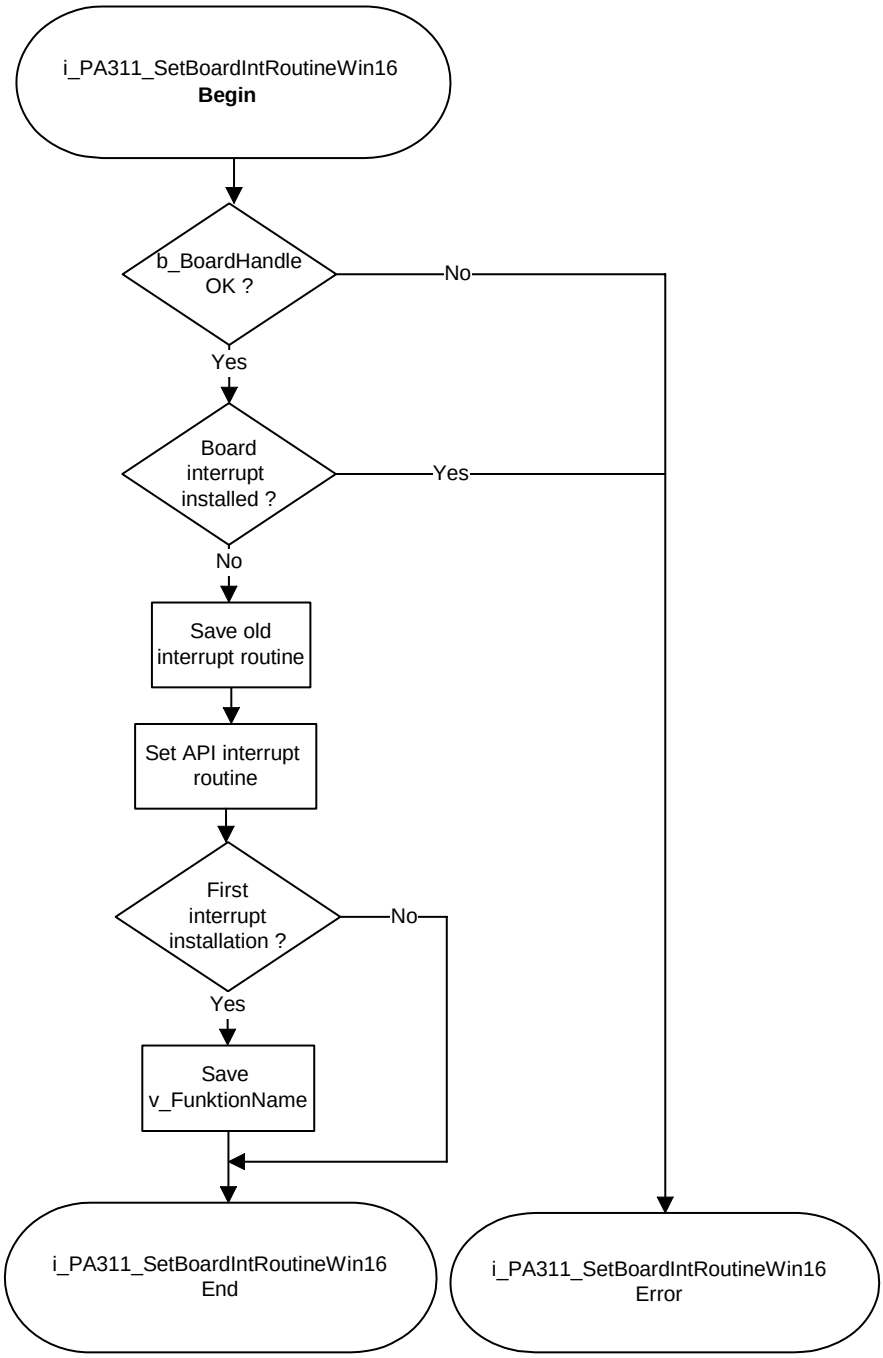
Return value:

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

Input

b_BoardHandle
v_FunctionName

Function diagram



Output

<Return Value>

4) i_PA311_SetBoardIntRoutineWin32 (..)

Syntax:

```
<Return value> = i_PA311_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))
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of board **PA 311**
BYTE	b_UserCallingMode	PA311_SYNCHRONOUS_MODE : User routine is directly called by driver interrupt routine. PA311_ASYNCHRONOUS_MODE : User routine is called by driver interrupt thread.
VOID	v_FunctionName	Name of the user interrupt routine
ULONG	ul_UserSharedMemorySize	Determines in byte the size of the user shared memory. Only used if you have selected PA311_SYNCHRONOUS_MODE

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

- Output:

VOID **	ppv_UserSharedMemory	User shared memory address Only used if you have selected PA311_SYNCHRONOUS_MODE

Task:

If you use Visual Basic 5.0 :
- only the asynchronous mode is available.

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 **PA 311** 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 PA311_SYNCHROUNOUS_MODE has been selected.

If you operate several boards **PA 311** which have to react to interrupts, call up the function as often as you operate boards **PA 311**. The variable *v_FunctionName* is only relevant **for the first calling**.

From the second call of the function (next board):
- interrupts are enabled.

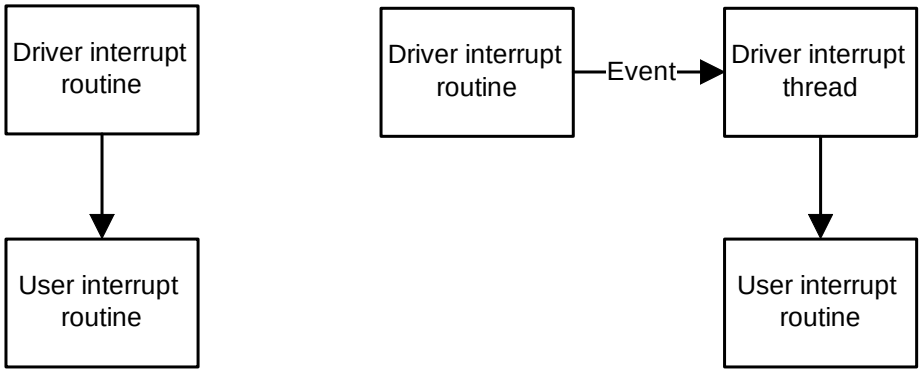
Interrupt
The user interrupt routine is called up by the system when an interrupt is generated.

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.

User interrupt routine can be called :

- 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 **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.
	The user routine can only call PA 311 driver functions with the following extension “i_PA311_KRNL_XXXX“
	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 PA 311 driver functions with the following extension: “i_PA311_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 PA311_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 byte 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 (BYTEb_BoardHandle,
                        BYTE        b_InterruptMask,
                        PUINT       b_AnalogInputValue,
                        BYTE        b_UserCallingMode,
                        VOID *      pv_UserSharedMemory)
```

v_FunctionName	Name of the user interrupt routine
b_BoardHandle	Handle of the **PA 311** which has generated the interrupt
b_InterruptMask	Mask of the events which have generated the interrupt.
pui_AnalogInputValue	The latched values of the counter are returned.
b_UserCallingMode	PA311_SYNCHRONOUS_MODE : The user routine is directly called by driver interrupt routine. PA311_ASYNCHRONOUS_MODE : The user routine is called by driver interrupt thread
pv_UserSharedMemory	Pointer of the user shared memory.

IMPORTANT!
If you use Visual Basic 4 the following parameters have no meaning. You must used the „i_PA311_TestInterrupt“ function.

```
BYTE        b_UserCallingMode,
ULONG       ul_UserSharedMemorySize,
VOID **     ppv_UserSharedMemory,
VOID        v_FunctionName      (BYTEb_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_PA311_SetBoardIntRoutineWin32
                                        (b_BoardHandle,
                                        PA311_SYNCHRONOUS_MODE,
                                         sizeof (str_UserStruct),
                                         (void **) &ps_UserSharedMemory,
                                         v_FunctionName);
```

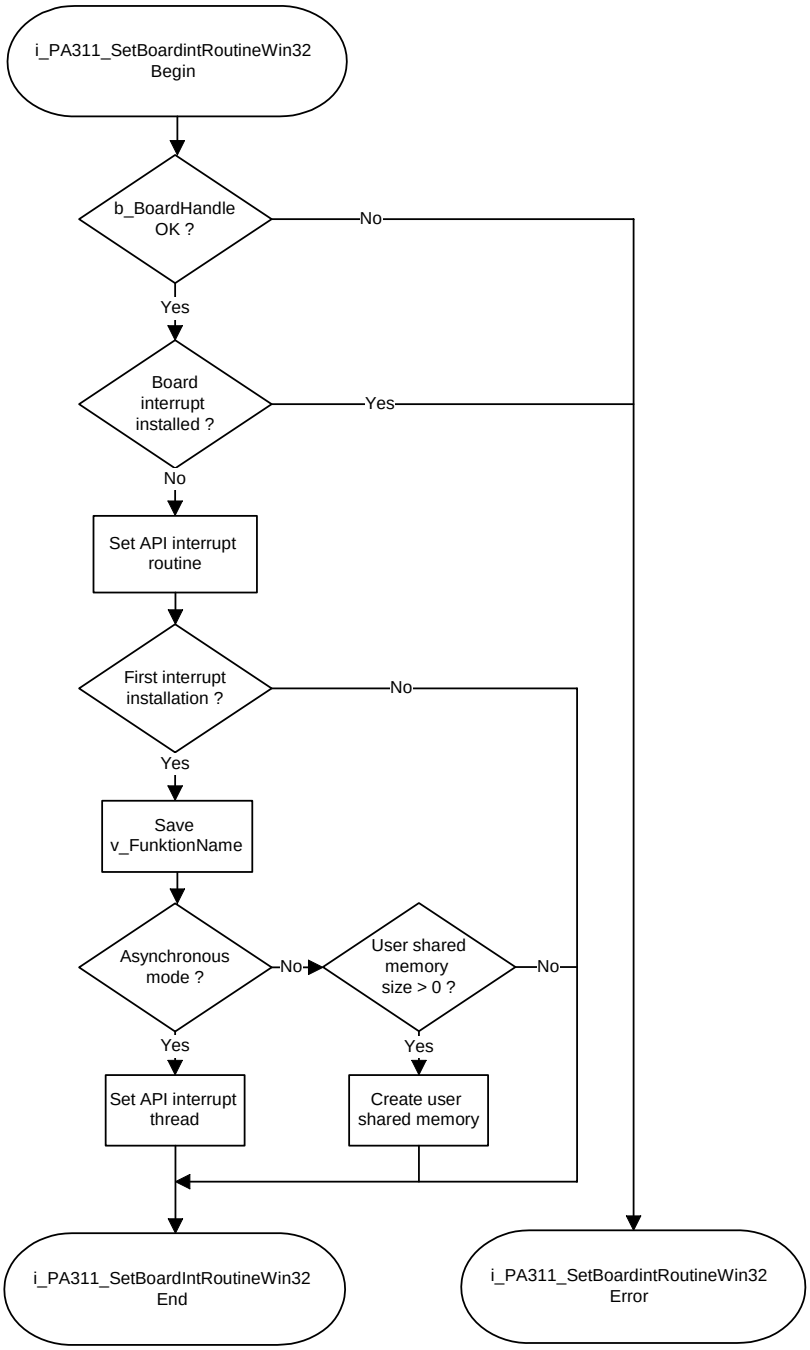
Return value:

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

Input

b_BoardHandle	ul_UserSharedMemorySize
b_UserCallingMode	ppv_UserSaredMemory
	v_FunctionName

Function diagram



Output

ppv_UserSaredMemory

<Return Value>

5) i_PA311_TestInterrupt (..)

Syntax:
<Return value> = i_PA311_TestInterrupt
(PBYTE pb_BoardHandle,
PBYTE pb_InterruptMask,
PUINT pui_AnalogInputValue)

Parameter:
- Input:
No input signal has to occur
- Output:

PBYTE	pb_BoardHandle	Handle of the board **PA 311** which has generated the interrupt
PBYTE	pb_InterruptMask	Mask of the event(s) which has(have) generated the interrupt. Several events can occur simultaneously
PUINT	pui_AnalogInputValue	The values of the analog inputs and of the DMA buffer are returned.
y	(See tables 3-1 and 3-2)	

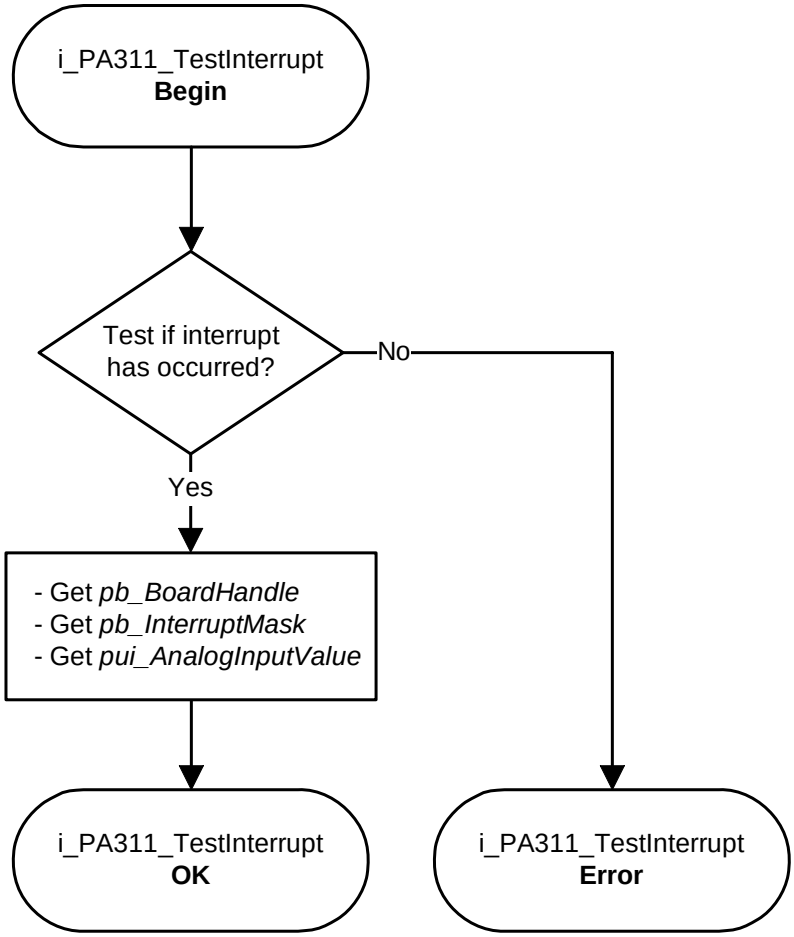
Task:
Verifies if a board **PA 311** has generated an interrupt. If yes, the function returns the board handle and the interrupt source.

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

Calling convention:
ANSI C :

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

i_ReturnValue = i_PA311_TestInterrupt   (&b_BoardHandle,
                                         &b_InterruptMask,
                                         ui_AnalogInputValue);
```

Return value:
-1 : No interrupt
> 0 : IRQ number

Input
b_BoardHandle pui_AnalogInputValue pb_InterruptMask
Function diagram

Output
b_BoardHandle pui_AnalogInputValue pb_InterruptMask
<Return Value>

6) i_PA311_ResetBoardIntRoutine (..)

Syntax:
<Return value> = i_PA311_ResetBoardIntRoutine (BYTE b_BoardHandle)

Parameter:
- Input:
BYTE b_BoardHandle Handle of board **PA 311**
- Output:
No output signal has occurred.

Task:
Stops the interrupt management of board **PA 311.**
Deinstalls the user interrupt routine if the management of interrupts of all boards **PA 311** is stopped.

Calling convention:
ANSI C :

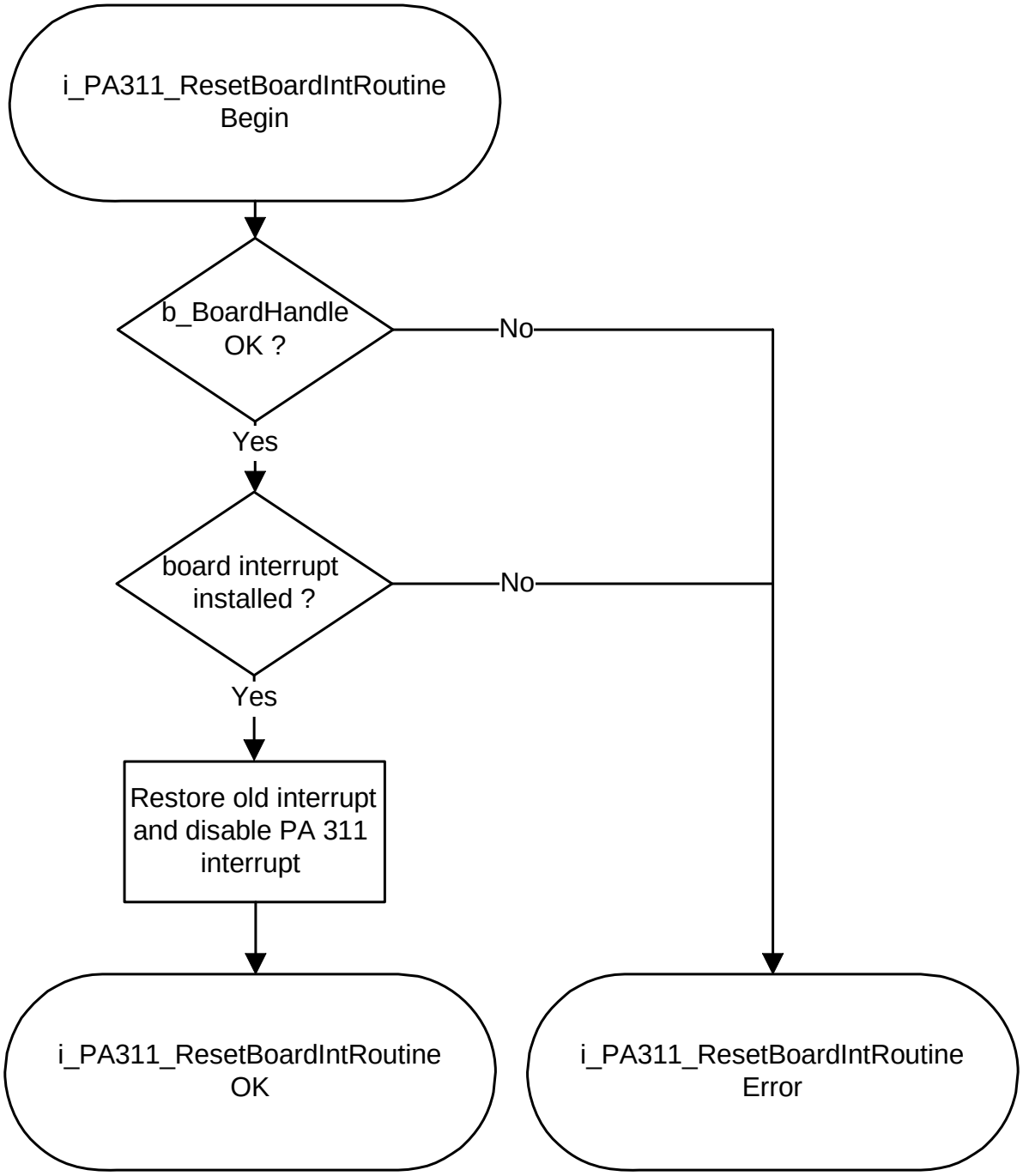
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PA311_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_PA311_SetBoardIntRoutineXXX"

Input

b_BoardHandle

Function diagram



Output

<Return Value>

3.3 Direct conversion of anlog input channels

1) i_PA311_Read1AnalogInput (...)

Syntax:

```
<Return value> =  i_PA311_Read1AnalogInput
                                (BYTEb_BoardHandle,
                                 BYTE       b_Channel,
                                 BYTE       b_Gain,
                                 BYTE       b_Polarity,
                                 BYTE       b_InterruptFlag,
                                 PUINT      pui_AnalogInputValue)
```

Parameter:

- Input:

BYTE	b_BoardHandle	Handle of board **PA311**
BYTE	b_Channel	Number of the analog input to be read (see table 3-3).
BYTE	b_Gain	Gain selection. See table 3-4 PA311_1_GAIN: Gain 1 (1 Hex) PA311_2_GAIN: Gain 2 (2 Hex) PA311_10_GAIN: Gain 10 (4 Hex) PA311_USER_GAIN: User gain (8 Hex)
BYTE	b_Polarity	Selection of the input voltage range of the analog input to be converted. See table 3-5. PA311_UNIPOLAR : Unipolar (1 hex) PA311_BIPOLAR : Bipolar (0 Hex)
BYTE	b_InterruptFlag	PA311_ENABLE : An interrupt is generated at EOC. See function "i_PA311_SetBoardIntRoutine". PA311_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 65535 for 16-Bit - 1 to 16383 for 14-Bit

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*.

Calling convention:

ANSI C :

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

i_ReturnValue = i_PA311_Read1AnalogInput        (b_BoardHandle,
                                                 PA311_CHANNEL_1,
                                                 PA311_1_GAIN,
                                                 PA311_UNIPOLAR,
                                                 PA311_DISABLE,
                                                 & ui_AnalogInputValue);
```

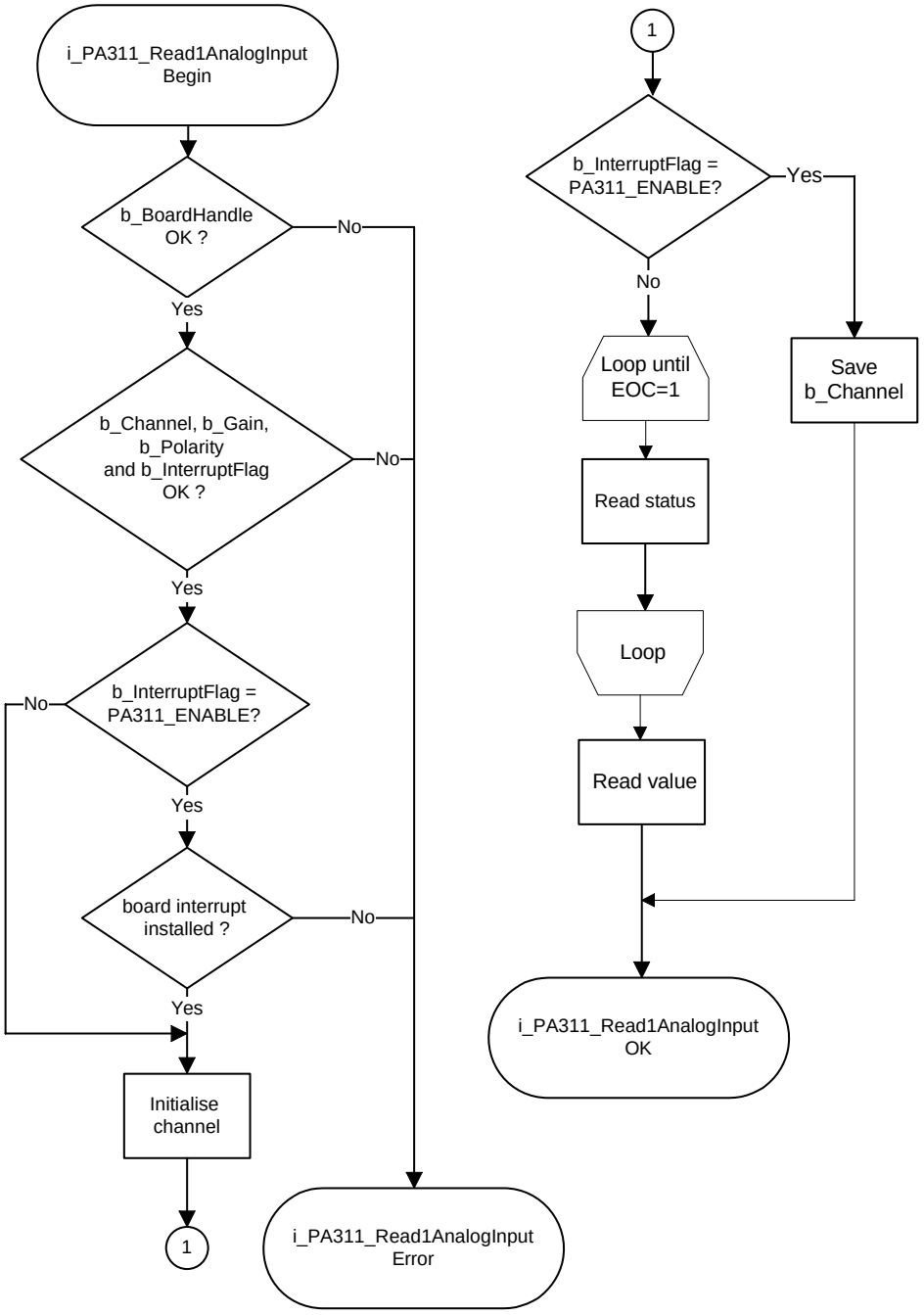
Return value:

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: Wrong parameter entered for *b_InterruptFlag* or the user interrupt routine has not been installed. See function "i_PA311_SetBoardIntRoutine".

Input

b_BoardHandle	b_Polarity
p_Channel	b_InterruptFlag
b_Gain	pui_AnalogInputValue

Function diagram



Output

pui_AnalogInputValue

<Return Value>

2) i_PA311_ReadMoreAnalogInput (...)

Syntax:

```
<Return value> =  i_PA311_ReadMoreAnalogInput
                                    (BYTE     b_BoardHandle,
                                     BYTE     b_SequenzArraySize,
                                     PBYTE    pb_ChannelArray,
                                     PBYTE    pb_GainArray,
                                     PBYTE    pb_PolarityArray,
                                     BYTE     b_InterruptFlag,
                                     PUINT    pui_AnalogInputValueArray)
```

Parameter:

- Input:

BYTE	b_BoardHandle	Handle of board **PA311**
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_InterruptFlag	PA311_ENABLE : An interrupt is generated when the last conversion of the channel group is completed (EOS). See function "i_PA311_SetBoardIntRoutine". PA311_DISABLE : No interrupt is generated at the end of conversion. The analog values are located in parameter *pui_AnalogInputValueArray*.

- Output:

PUINT	pui_AnalogInputValueArray	Output 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_PA311_ReadMoreAnalogInput     (b_BoardHandle,
                                                 16,
                                                 b_ChannelArray,
                                                 b_GainArray,
                                                 b_PolarityArray,
                                                 PA311_DISABLE,
                                                 ui_AnalogInputValue);
```

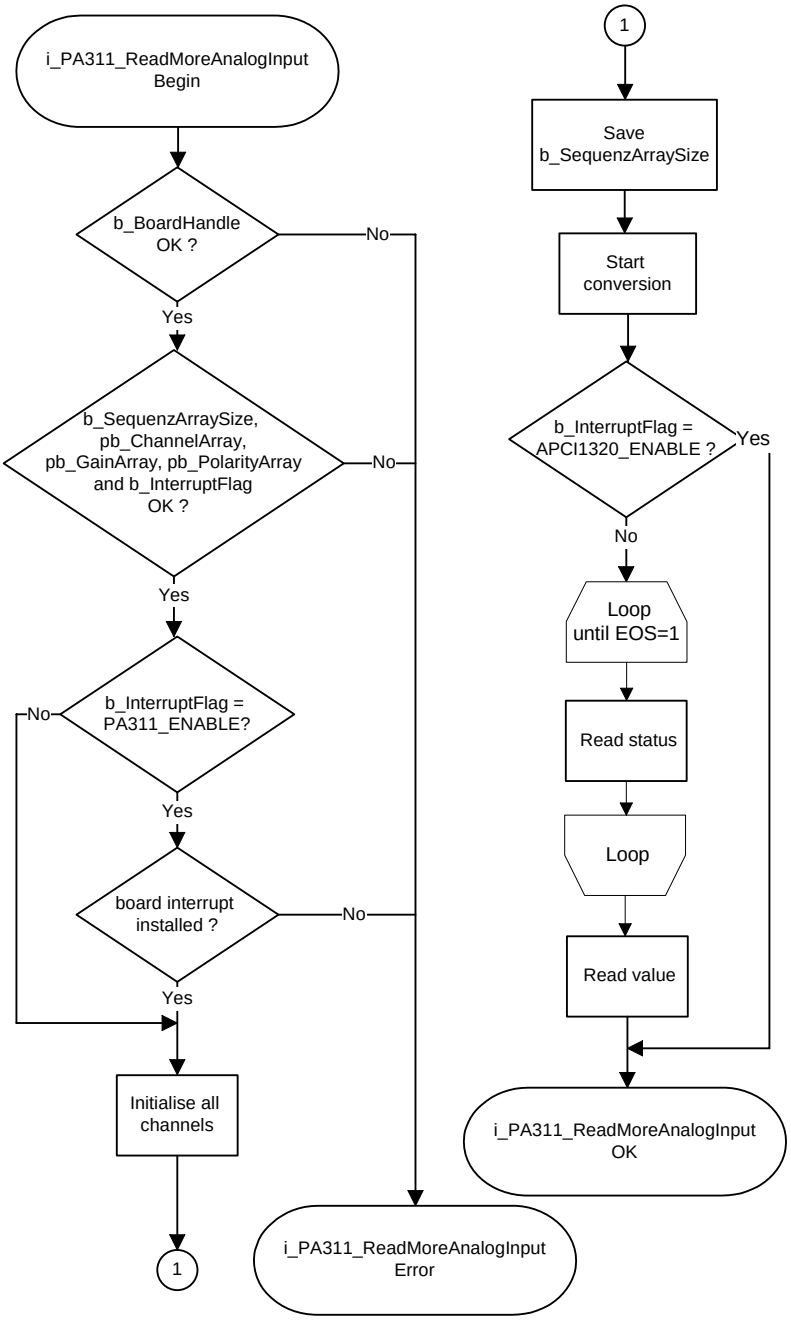
Return value:

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: Wrong parameter entered for *b_InterruptFlag*, or the user interrupt routine has not been installed. See function "i_PA311_SetBoardIntRoutine".

Input

b_BoardHandle	b_SequenzArraySize
p_ChannelArray	b_GainArray
b_PolarityArray	pui_AnalogInputValue
b_InterruptFlag	

Function diagram



Output

pui_AnalogInputValueArray

<Return Value>

3.4 Cyclic conversion of the analog input channels

1) i_PA311_InitAnalogInputAcquisition (...)

Syntax:

```
<Return value> =  i_PA311_InitAnalogInputAcquisition
                                    (BYTE    b_BoardHandle,
                                     BYTE    b_ChannelNbr,
                                     BYTE    b_Gain,
                                     BYTE    b_Polarity,
                                     UINT    ui_NumberOfAcquisition,
                                     BYTE    b_AcquisitionMode,
                                     BYTE    b_AcquisitionCycle,
                                     BYTE    b_Timer0InputFrequenzy,
                                     LONG    l_AcquisitionTiming,
                                     BYTE    b_Timer1InputFrequenzy,
                                     LONG    l_DelayTiming,
                                     BYTE    b_DMAUsed)
```

Parameter:

- Input:

BYTE	b_BoardHandle	Handle of board **PA311**
BYTE	b_ChannelNbr	Number of channles to be converted. (1 up to 16) Test if the first channel is channel 0 and if the last channel is *b_ChannelNbr* - 1
BYTE	b_Gain	Gain selection. See table 3-4 PA311_1_GAIN: Gain fron 1 (1 Hex) PA311_2_GAIN: Gain fron 2 (2 Hex) PA311_10_GAIN: Gain fron 10 (4 Hex) PA311_USER_GAIN: User gain (8 Hex)
BYTE	b_Polarity	Selection of the input voltage range of the analog input to convert. See table 3-5. PA311_UNIPOLAR: Unipolar (1 hex) PA311_BIPOLAR: Bipolar (0 Hex)
UINT	ui_NumberOfAcquisition	If you use DMA, this parameter determines how many conversions have to occur (1 to 2^{15}). If you do not use DMA, this parameter determines how many acquisition cycles must be performed.

BYTE	b_AcquisitionMode	Two conversion cycles are possible: - PA311_SIMPLE_MODUS: A conversion occurs every *ui_AcquisitionTiming* (time interval) See example 2. - PA311_DELAY_MODUS: Both times are used in this mode *ui_AcquisitionTiming* and *l_DelayTiming*. Conversions occur every *ui_AcquisitionTiming* (time interval) until all the analog inputs have been acquired (determined by *b_ChannelNbr).* 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.
BYTE	b_AcquisitionCycle	Defines the type of DMA conversion. - PA311_CONTINUOUS: An interrupt is generated each time, a DMA conversion cycle is completed. A new DMA conversion cycle is started. - PA311_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.
BYTE	b_Timer0InputFrequency	Input frequency of timer 0: PA311_LOW_FREQUENCY: 27.901 kHz PA311_HIGH_FREQUENCY: 892.857 kHz
LONG	l_AcquisitionTiming	Time interval in µs between 2 conversions of successive inputs: - from 10 µs to 73399 µs, if you have selected the low frequenzy for timer 0 - from 72 µs to 2348841 µs, if you have selected the low frequenzy for timer 0 See example 1 and 2.
BYTE	b_Timer1InputFrequency	Input frequency of timer 1: PA311_LOW_FREQUENCY: 27.901 kHz PA311_HIGH_FREQUENCY: 892.857 kHz

LONG	l_DelayTiming	Waiting time in µs between two conversion cycles - from 10 µs to 73399 µs, if you have selected the low frequenzy for timer 1 - from 72 µs to 2348841 µs, if you have selected the low frequenzy for timer 1 This parameter is relevant only if you use the mode PA311_DELAY_MODUS.
BYTE	b_DMAUsed	Determines if DMA should be used or not. - PA311_DMA_USED: All the conversion values are saved in the DMA buffer. An interrupt is generated, when *ui_NumberOfAcquisition* conversions have been completed. See function "i_PA311_SetBoardIntRoutine". - PA311_DMA_NOT_USED : An interrupt is generated when the conversion of the last channel group has been competed. You obtain the analog value through the user interrupt routine. See function "i_PA311_SetBoardIntRoutine".

- Output:
No output signal has occurred.

Task:
This function initialises a cyclic conversion.
The number of analog channels is determined via *b_ChannelNbr*.
The input voltage range and the gain are the same for each analog input.
The DMA option (PA311_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 if you have not used the DMA.
A "8" is passed if you have used the DMA.
The DMA buffer is returned through the parameter *pui_AnalogInputValue*.
See function "i_PA311_SetBoardIntRoutine".

Do the following:
- enter the mode through the parameter *b_AcquisitionMode*.
- enter the time between 2 conversions through the parameter *l_AcquisitionTiming*.
- enter the waiting time between two conversion cycles through the parameter *l_DelayTiming*, if you work in mode PA311_DELAY_MODUS.
- determine if you use DMA (parameter *b_DMAUsed*)
- enter the number of acquisitions through parameter *ui_NumberOfAcquisition*
- enter the DMA conversion cycle through parameter *b_AcquisitionCycle*, if DMA is used.

Table 3-3: Selection of the analog inputs

pb_ChannelArray Parameter	Analog input	Define decimal value
PA311_CHANNEL_0	0	0
PA311_CHANNEL_1	1	1
PA311_CHANNEL_2	2	2
PA311_CHANNEL_3	3	3
PA311_CHANNEL_4	4	4
PA311_CHANNEL_5	5	5
PA311_CHANNEL_6	6	6
PA311_CHANNEL_7	7	7
PA311_CHANNEL_8	8	8
PA311_CHANNEL_9	9	9
PA311_CHANNEL_10	10	10
PA311_CHANNEL_11	11	11
PA311_CHANNEL_12	12	12
PA311_CHANNEL_13	13	13
PA311_CHANNEL_14	14	14
PA311_CHANNEL_15	15	15

Table 3-4: Gain selection

pb_GainArray Parameter	Gain	Define decimal value
PA311_1_GAIN	1	1
PA311_2_GAIN	2	2
PA311_10_GAIN	10	4
PA311_USER_GAIN	User selection	8

Table 3-5 : Selection of the input voltage range

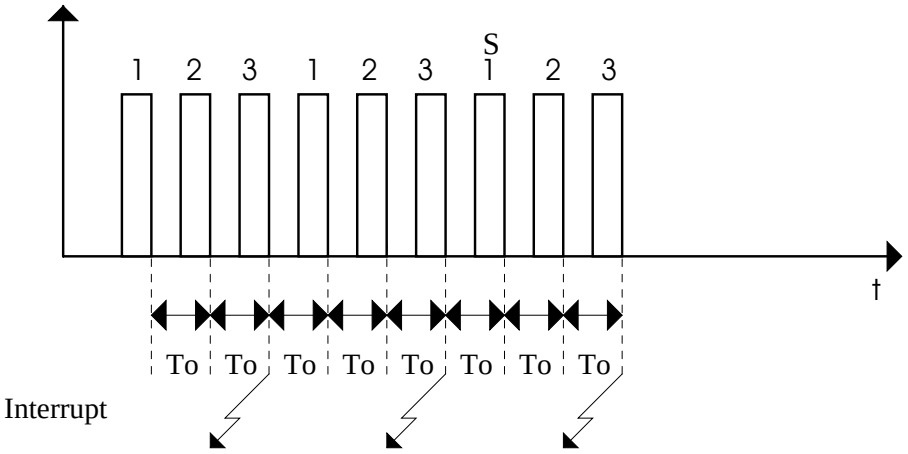
pb_PolarityArray Parameter	Voltage range	Define decimal value
PA311_UNIPOLAR	0-10V with gain 1	1
PA311_BIPOLAR	±10V with gain 1	0

Examples:

Example 1: Cyclic conversion without DMA

```
i_PA311_InitAnalogInputAcquisition
                        (b_BoardHandle,
                         3,
                         PA311_1_GAIN,
                         PA311_UNIPOLAR,
                         9,
                         PA311_SIMPLE_MODUS,
                         PA311_SINGLE,
                         PA311_LOW_FREQUENCY,
                         T0,
                         PA311_HIGH_FREQUENCY,
                         0,
                         PA311_DMA_NOT_USED)
```

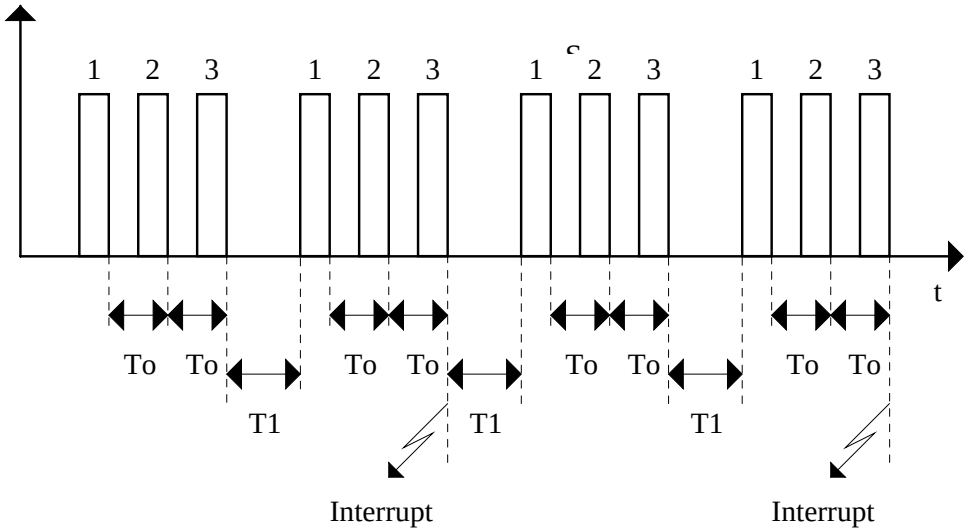
- b_AcquisitionMode = PA311_SIMPLE_MODUS
- b_ExternTrigger = PA311_DISABLE
- ui_AcquisitionTiming = To
- b_DMAUsed = PA311_DMA_NOT_USED
- ui_NumberOfAquisisition = 3



Example 2: Cyclic conversion with DMA

```
i_PA311_InitAnalogInputAcquisition
                    (b_BoardHandle,
                     3,
                     PA311_1_GAIN,
                     PA311_UNIPOLAR,
                     6,
                     PA311_DELAY_MODUS,
                     PA311_SINGLE,
                     PA311_LOW_FREQUENCY,
                     T0,
                     PA311_HIGH_FREQUENCY,
                     T1,
                     PA311_DMA_USED)
```

- b_AcquisitionMode = PA311_DELAY_MODUS
- b_ExternTrigger = PA311_DISABLE
- ui_NumberOfAcquisition = 6
- ui_AcquisitionTiming = To
- l_DelayTiming = T1
- b_DMAUsed = PA311_DMA_USED
- b_AcquisitionCycle = PA311_CONTINUOUS
- b_ExternTrigger = PA311_DISABLE



Calling convention:

ANSI C :

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

i_PA311_InitAnalogInputAcquisition
                                (b_BoardHandle,
                                 3,
                                 PA311_1_GAIN,
                                 PA311_UNIPOLAR,
                                 6,
                                 PA311_SIMPLE_MODUS,
                                 PA311_SINGLE,
                                 PA311_LOW_FREQUENCY,
                                 10,
                                 PA311_HIGH_FREQUENCY,
                                 0,
                                 PA311_DMA_USED)
```

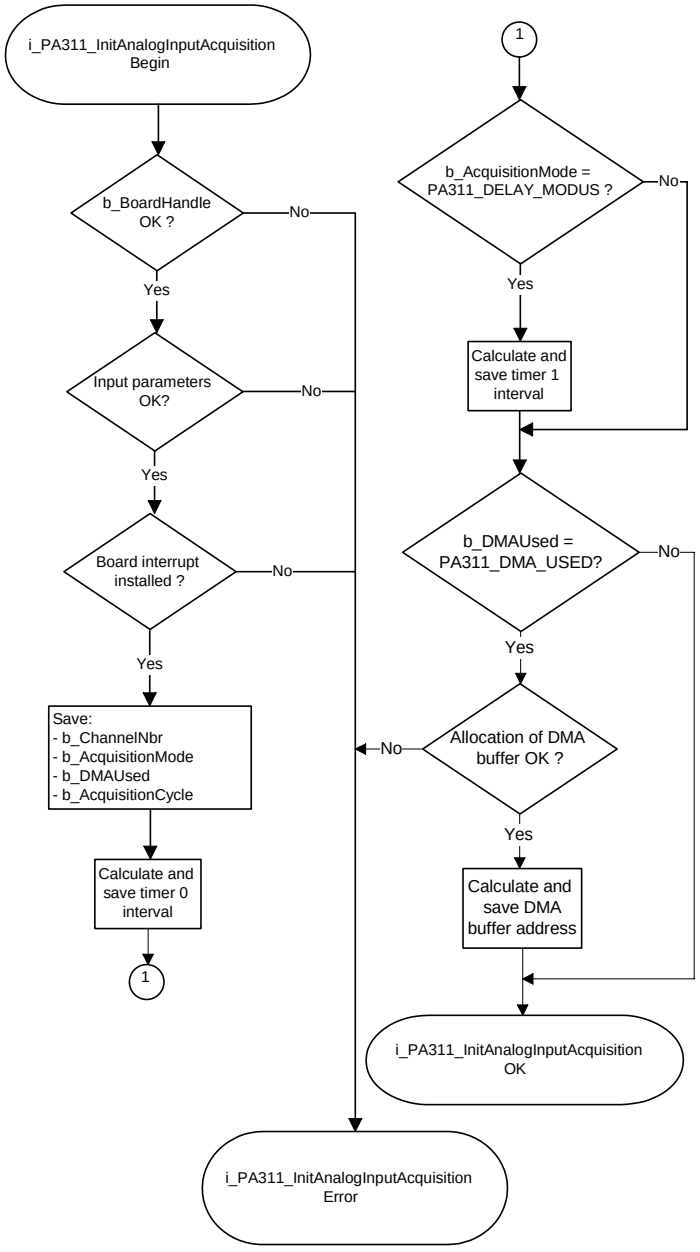
Return value:

```
 0  : No error
-1  : The handle parameter of the board is wrong
-2  : The user interrupt routine is not installed.
      See function "i_PA311_SetBoardIntRoutine"
-3  : The number of analog inputs is wrong
-4  : Gain selection is wrong
-5  : Polarity selection is wrong
-6  : The number of analog conversions is wrong (1 to 32767)
-7  : The input clock selection for timer 0 is wrong
-8  : The waiting time between two conversion cycles is too high
-9  : The input clock selection for timer 1 is wrong
-10: The selected time for l_DelayTiming is wrong
-11: The parameter b_DMAMode is wrong
-12: The parametered running time of the DMA conversion cycle is wrong
      (PA311_CONTINUOUS or PA311_SINGLE)
-13 : The parametered conversion cycle is wrong
      (PA311_SIMPLE_MODUS or PA311_DELAY_MODUS
-14 : DMA channel not installed
-15 : Not enough memory available
```

Input

b_BoardHandle	b_AcquisitionCycle
b_ChannelNbr	b_Timer0InputFrequency
b_Gain	l_AcquisitionTiming
b_Polarity	b_Timer1InputFrequency
ui_NumberOfAcquisition	l_DelayTiming
b_AcquisitonMode	b_DMAUsed

Function diagram



Output

<Return Value>

2) i_PA311_StartAnalogInputAcquisition (...)

Syntax:
<Return value> = i_PA311_StartAnalogInputAcquisition
(BYTEb_BoardHandle)

Parameter:
- Input:
BYTE b_BoardHandle Handle of board **PA311**
- Output:
No output signal has occurred.

Task:
Starts the cyclic conversion. It has been previously initialised with function „i_PA311_InitAnalogInputAcquisition“.

Calling convention:
ANSI C :

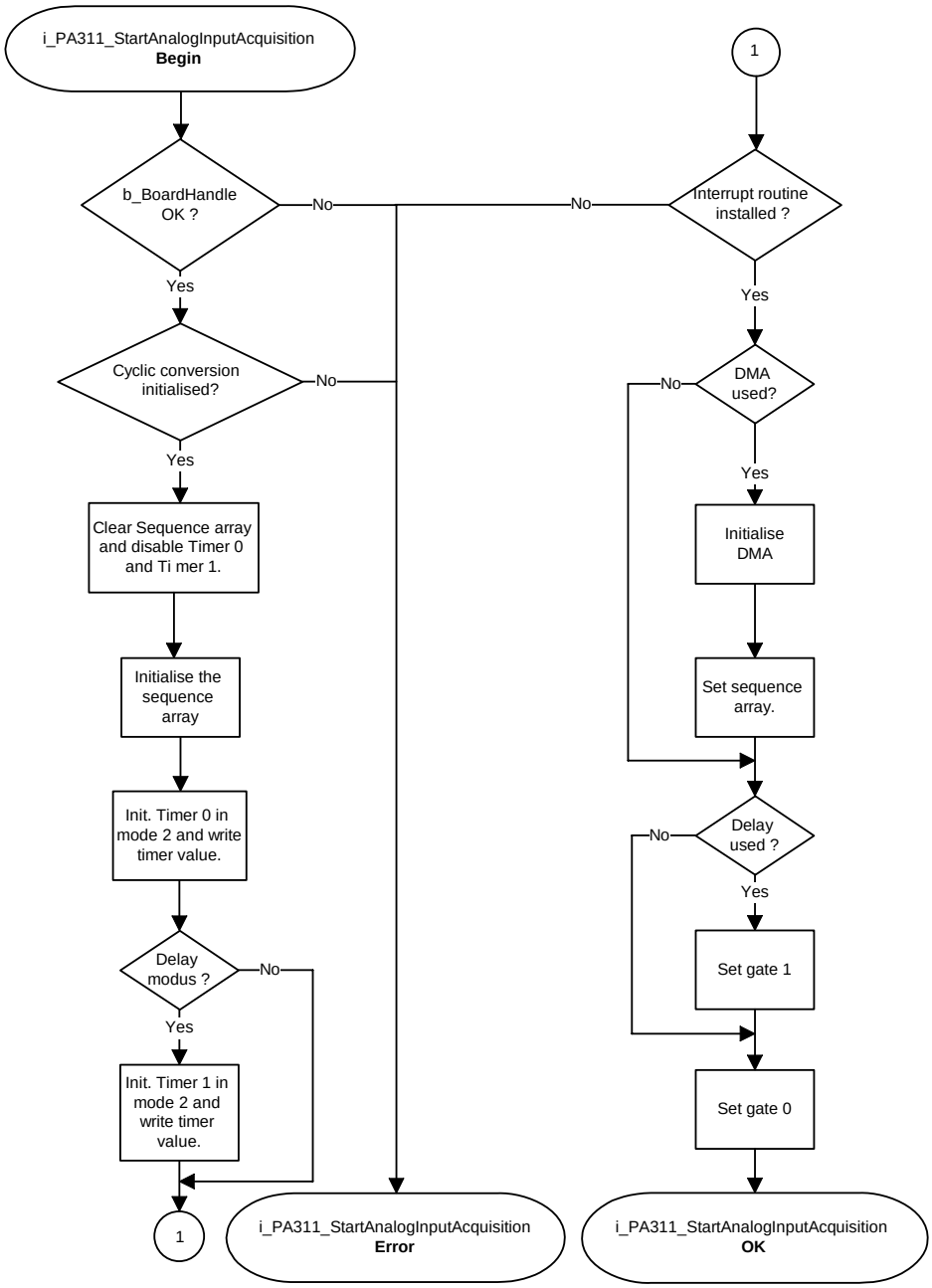
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PA311_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_PA311_InitAnalogInputAcquisition"
-3: User interrupt routine has not been installed.
See function "i_PA311_SetBoardIntRoutine"

Input

b_BoardHandle

Function diagram



Output

<Return Value>

3) i_PA311_StopAnalogInputAcquisition (...)

Syntax:
< Return value > = i_PA311_StopAnalogInputAcquisition
(BYTEb_BoardHandle)

Parameter:
- Input:
BYTE b_BoardHandle Handle of board **PA311**
- Output:
No output signal has occurred.

Task:
Stops the cyclic conversion. It has been previously started with function „i_PA311_StartAnalogInputAcquisition“

Calling convention:
ANSI C :

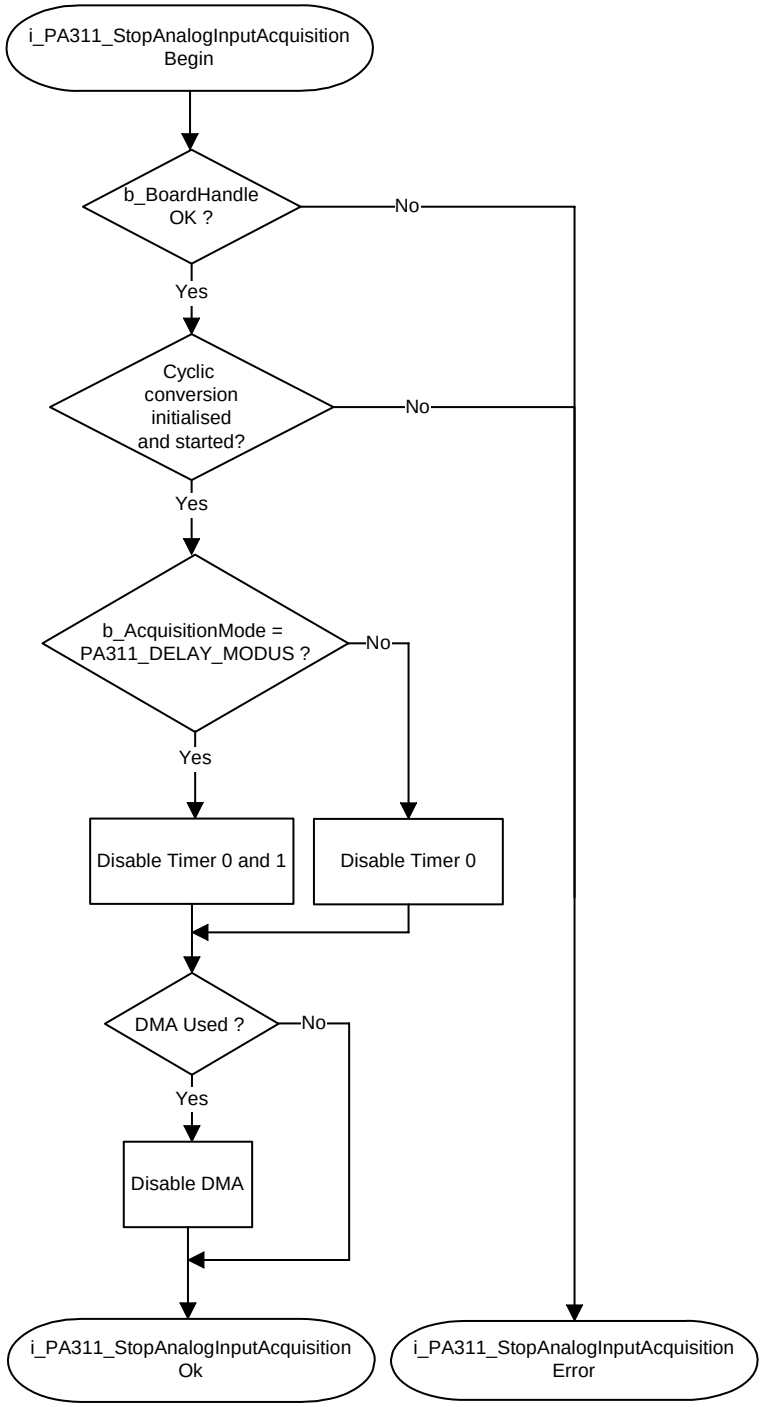
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PA311_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_PA311_StartAnalogInputAcquisition"

Input

b_BoardHandle

Function diagram



Output

<Return Value>

4) i_PA311_ClearAnalogInputAcquisition(...)

Syntax:
<Return value> = i_PA311_ClearAnalogInputAcquisition
(BYTE b_BoardHandle)

Parameter:
- Input:
BYTE b_BoardHandle Handle of board **PA311**
- Output:
No output signal has occurred.

Task:
Releases the DMA buffer.

Calling convention:
ANSI C :

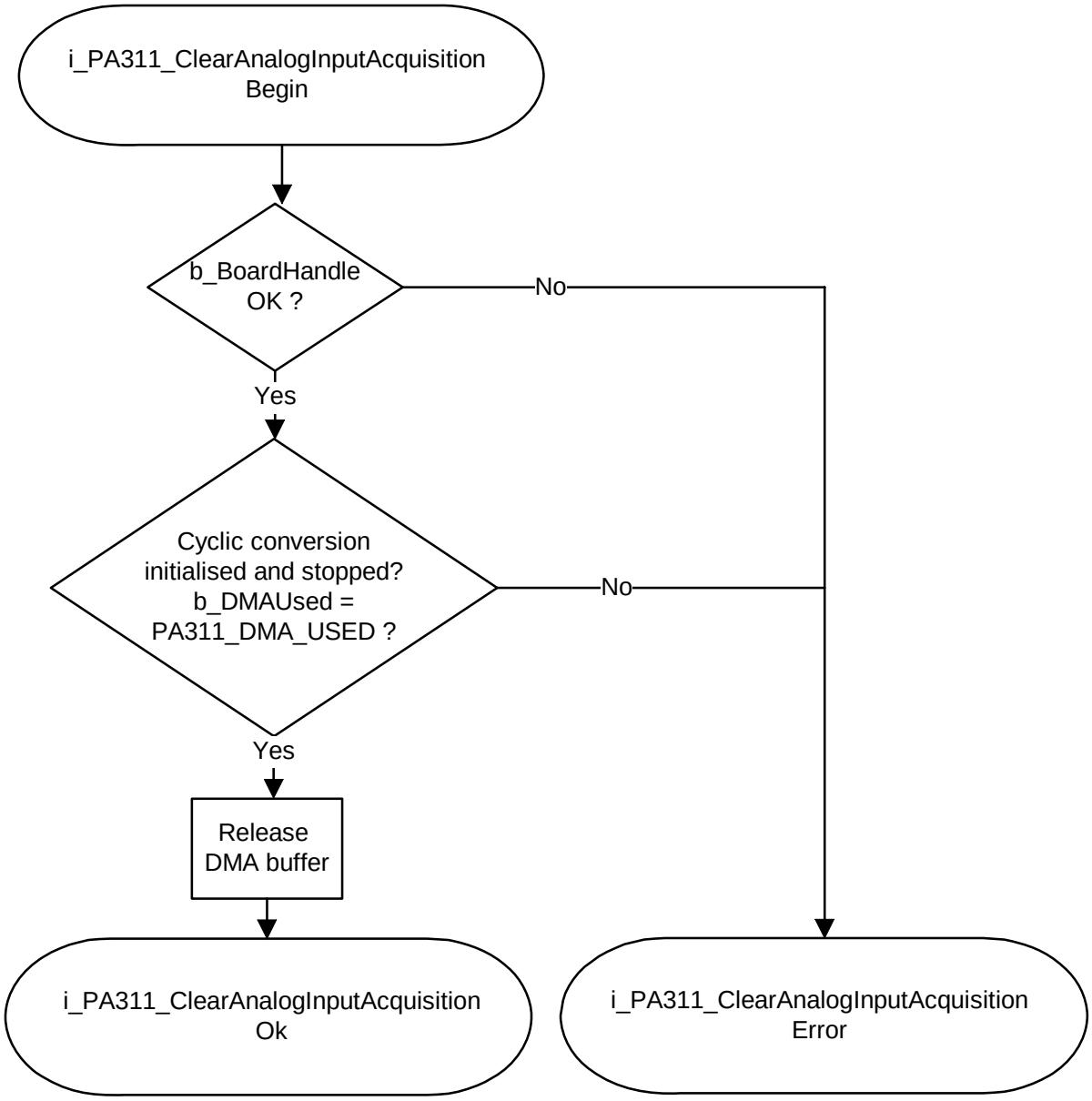
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PA311_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_PA311_InitAnalogInputAcquisition"

Input

b_BoardHandle

Function diagram



Output

<Return Value>

3.5 Analog outputs

1) i_PA311_Write1AnalogValue (...)

Syntax:
<Return value> = i_PA311_Write1AnalogValue
(BYTEb_BoardHandle,
BYTE b_ChannelNbr,
UINT ui_ValueToWrite)

Parameter:
- Input:

BYTE	b_BoardHandle	Handle of board **PA311**
BYTE	b_ChannelNbr	Number of the analog output (0 to 7)
UINT	ui_ValueToWrite	Analog output value to write - 0 to 65535 for 16-bit - 1 to 16383 for 14-bit

- Output:
No output signal has occurred.

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

Calling convention:
ANSI C :

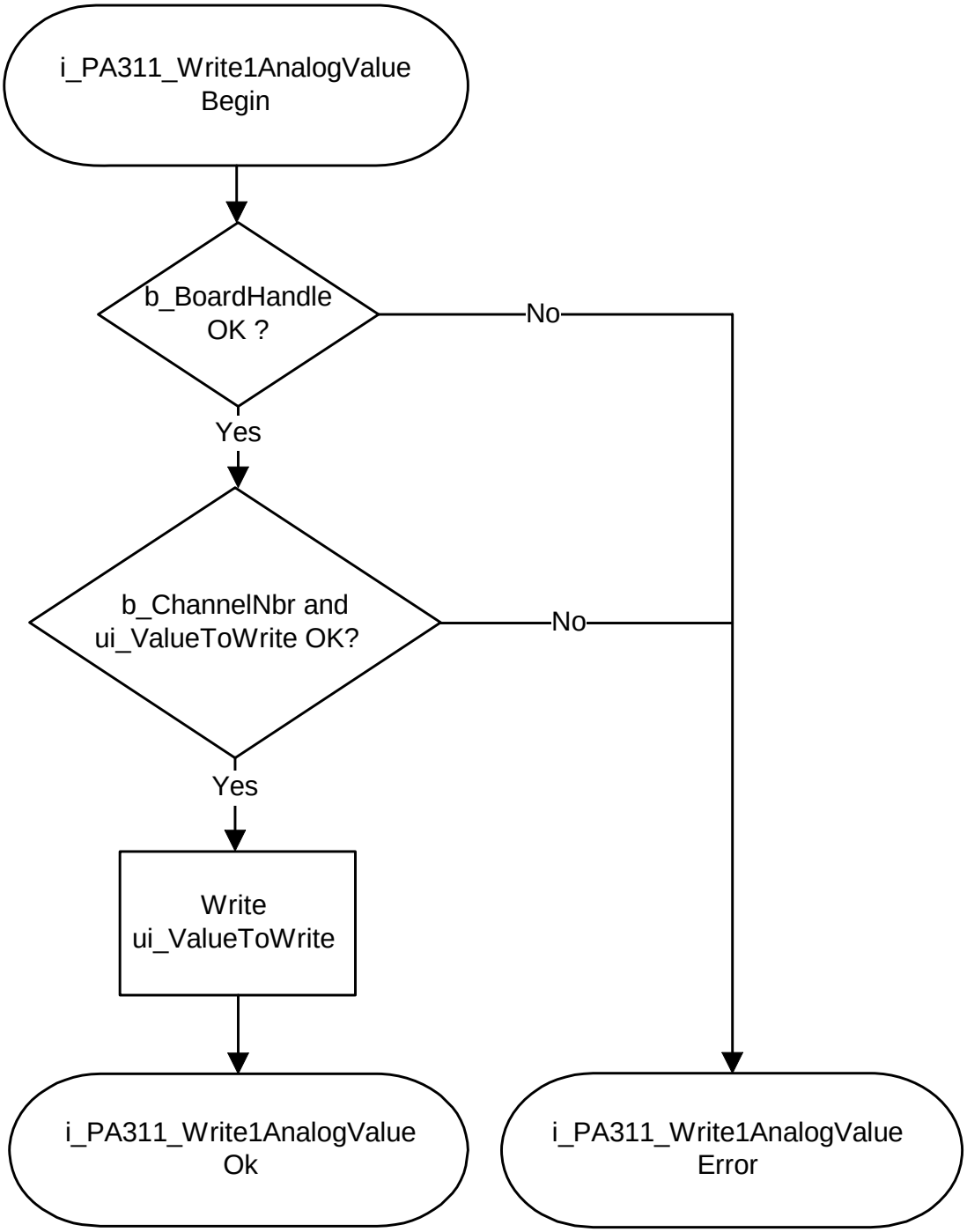
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PA311_Write1AnalogValue   (b_BoardHandle,
                                             1,
                                             4095);
```

Return value:
0: No error
-1: The handle parameter of the board is wrong
-2: Number of the analog output is wrong.
-3: Output value too high

Input

b_BoardHandle ui_ValueToWrite
b_ChannelNbr

Function diagram



Output

<Return Value>

2) i_PA311_WriteMoreAnalogValue (...)

Syntax:

```
<Return value> =  i_PA311_WriteMoreAnalogValue
                                    (BYTEb_BoardHandle,
                                     BYTE      b_FirstChannelNbr,
                                     BYTE      b_NbrOfChannel,
                                     PUINT     pui_ValueArray)
```

Parameter:

- Input:

BYTE	b_BoardHandle	Handle of board **PA311**
BYTE	b_FirstChannelNbr	Number of the first analog output (0 to 7)
BYTE	b_NbrOfChannel	Number of analog outputs to be written (1 to 8)
PUINT	pui_ValueArray	Table of analog output values to be written

- Output:

No output signal has occurred.

Task:

Writes several analog values on several analog outputs.
The variable *b_FirstChannelNbr* defines the first analog output.
The variable *b_NbrOfChannel* defines the number of analog outputs.

Calling convention:

ANSI C :

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

i_ReturnValue = i_PA311_WriteMoreAnalogValue    (b_BoardHandle,
                                                 0,
                                                 8,
                                                 ui_ValueArray);
```

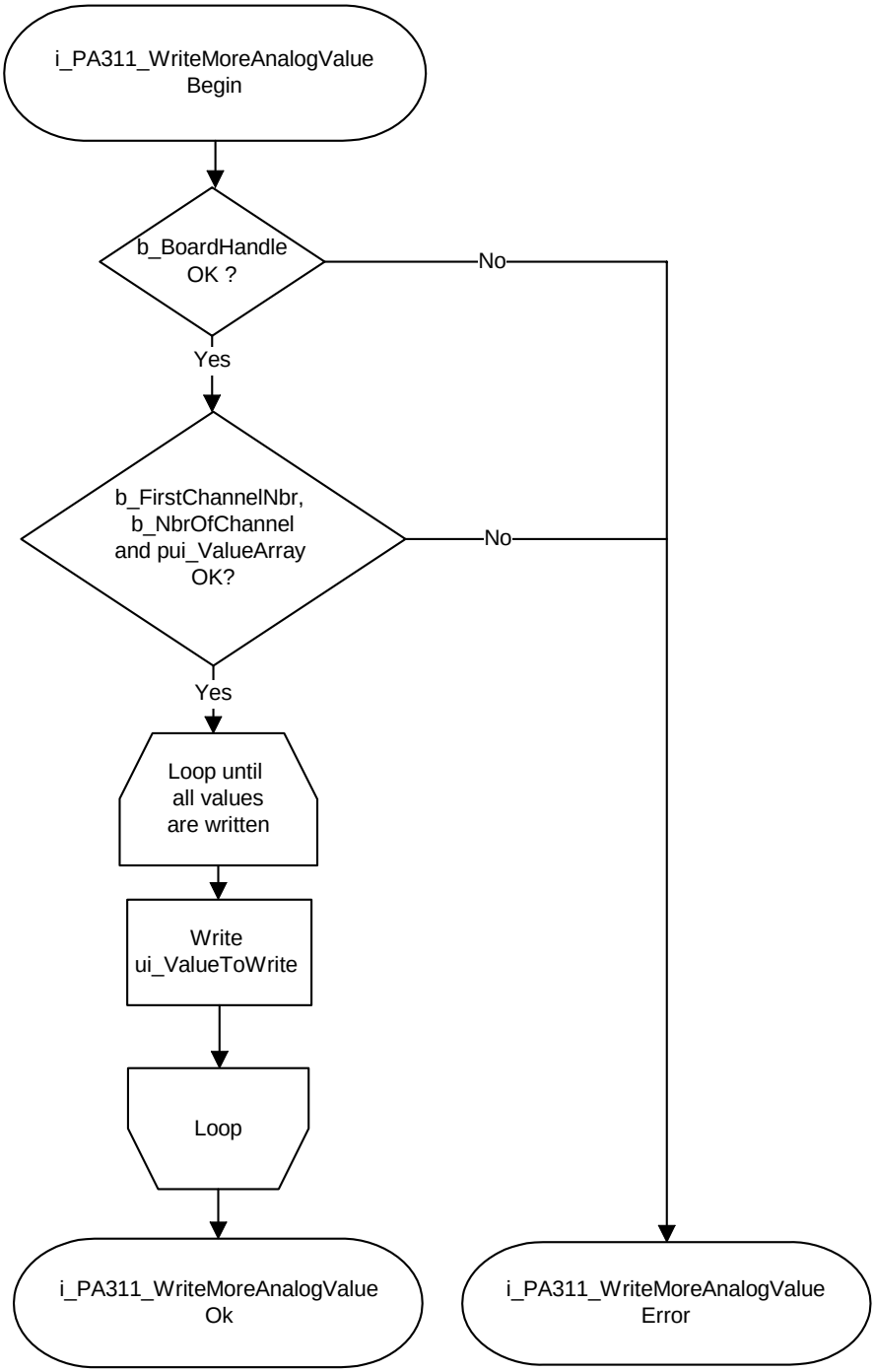
Return value:

0: No error
-1: The handle parameter of the board is wrong
-2: Number of the analog output is wrong
-3: The number of analog outputs you wish to write on is wrong
    See function "i_PA311_SetBoardInformation"
-4: One or several ouput value are too high.

Input

b_BoardHandle	b_NbrOfChannel
b_FirstChannelNbr	pui_ValueArray

Function diagram



Output

<Return Value>

3.6 Timer 2

1) i_PA311_InitTimer2 (...)

Syntax:
<Return value> = i_PA311_InitTimer2
(BYTEb_BoardHandle,
BYTE b_TimerInputFrequency,
LONG l_DelayValue,
BYTE b_InterruptFlag)

Parameter:
- Input:

BYTE	b_BoardHandle	Handle of board **PA 311**
BYTE	b_TimerInputFrequency	Input frequency of the timer: PA311_LOW_FREQUENCY: 27.901 kHz PA311_HIGH_FREQUENCY: 892.857 kHz
LONG	l_DelayValue	Time interval (Watchdog time) of the timer
BYTE	b_InterruptFlag	PA311_ENABLE: An interrupt is generated after the time interval. PA311_DISABLE: No interrupt is generated after the time interval.

- Output:
No interrupt signal has occurred.

Task:
Initialises timer 2 as an edge generator

Calling convention:
ANSI C :

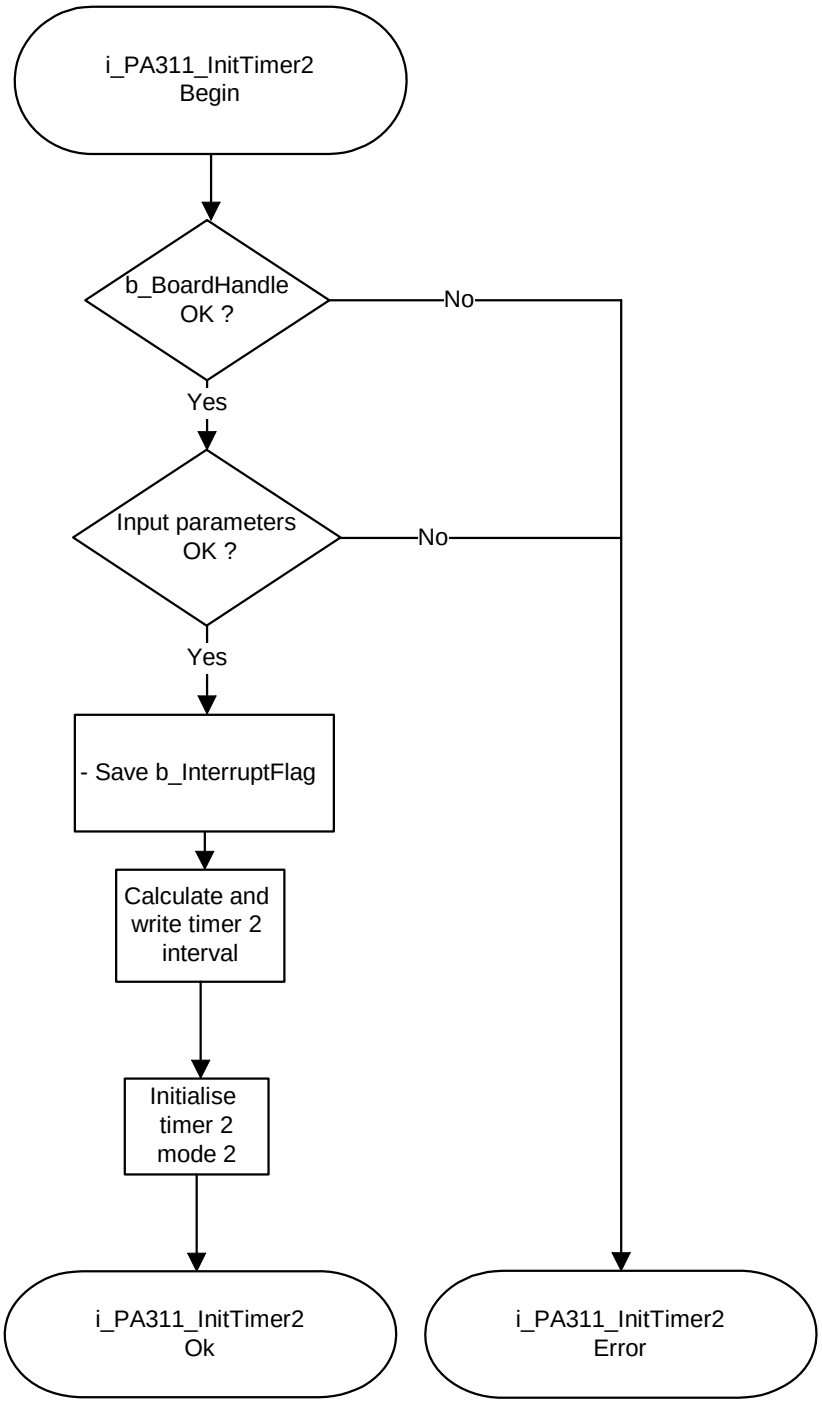
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PA311_InitTimer2   (b_BoardHandle,
                                      PA311_LOW_FREQUENCY
                                      1000,
                                      PA311_DISABLE);
```

Return value:
0: No error.
-1: The handle parameter of the board is wrong.
-2: User interrupt routine not installed. See function "i_PA311_SetBoardIntRoutine"
-3: Variable *b_InterruptFlag* is wrong
-4: The input frequency selected for timer 2 is wrong
-5: The time interval selected for timer 2 is wrong

Input

b_BoardHandle	l_DelayValue
b_TimerInputFrequency	b_InterruptFlag

Function diagram



Output

<Return Value>

2) i_PA311_StartTimer2 (...)

Syntax:
<Return value> = i_PA311_StartTimer2 (BYTE b_BoardHandle)

Parameter:
- Input:
BYTE b_BoardHandle : Handle of board **PA 311**
- Output:
No output signal has occurred.

Task:
Starts timer 2.

Calling convention:
ANSI C :

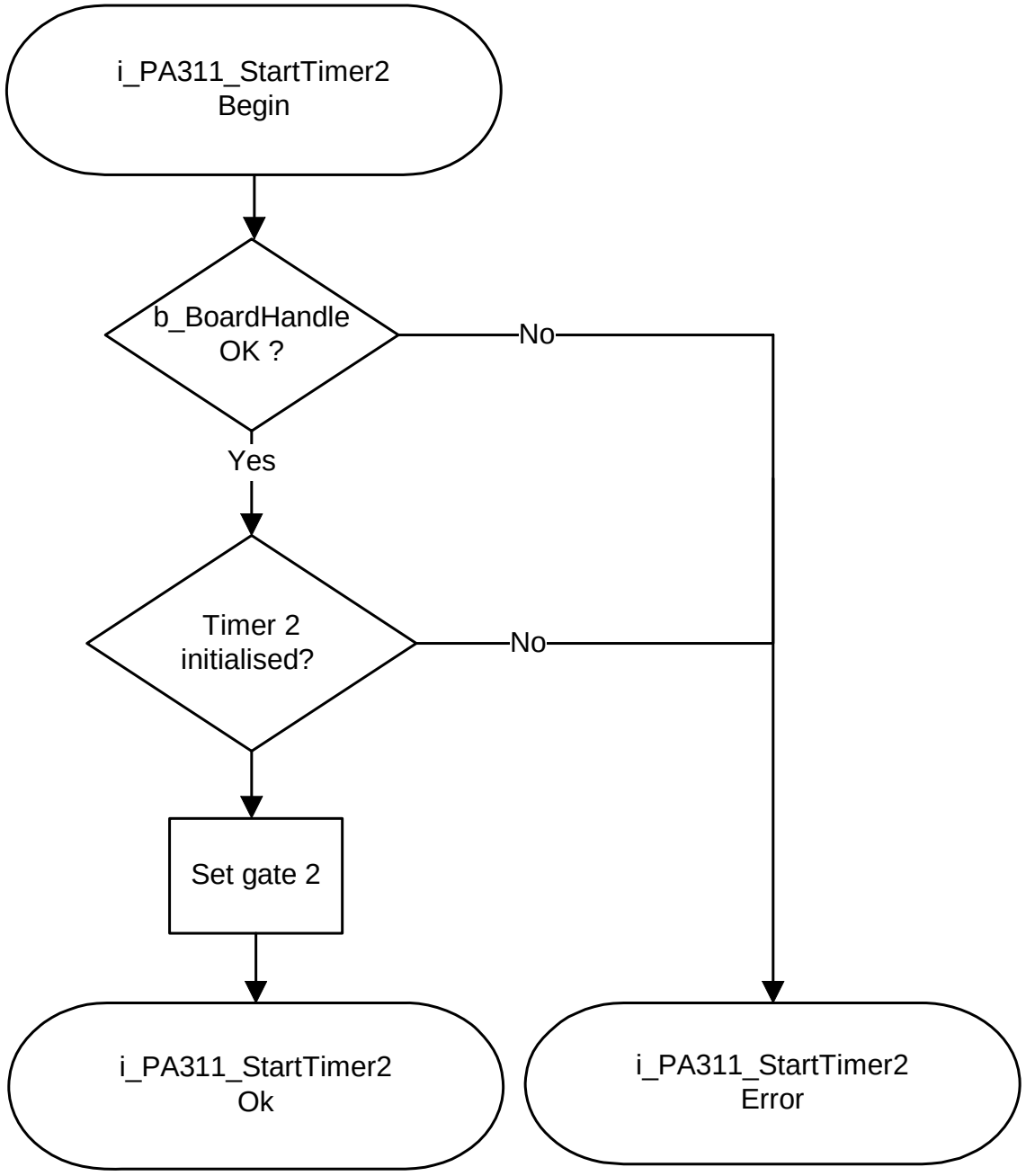
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PA311_StartTimer2     (b_BoardHandle);
```

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

Input

b_BoardHandle

Function diagram



Output

<Return Value>

3) i_PA311_StopTimer2 (...)

Syntax:
<Return value> = i_PA311_StopTimer2 (BYTEb_BoardHandle)

Parameter:
- Input
BYTE b_BoardHandle Handle of board **PA 311**
- Output:
No output signal has occurred.

Task:
Stops timer 2.

Calling convention:
ANSI C :

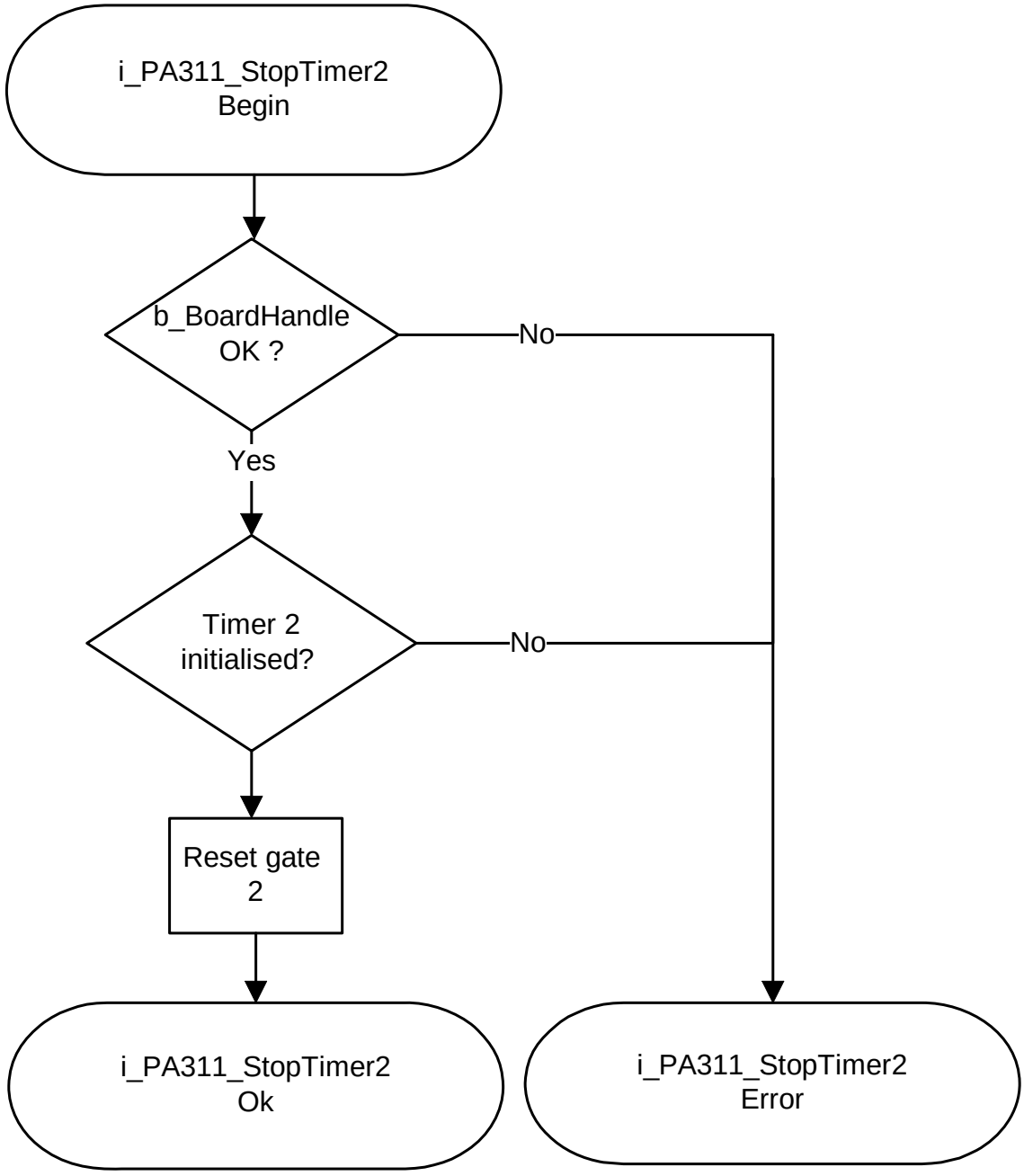
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PA311_StopTimer2      (b_BoardHandle);
```

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

Input

b_BoardHandle

Function diagram



Output

<Return Value>

4) i_PA311_ReadTimer2 (...)

Syntax:
<Return value> = i_PA311_ReadTimer2
(BYTE b_BoardHandle
PLONG pl_ReadValue)

Parameter:
- Input:

BYTE	b_BoardHandle	Handle of board **PA 311**

- Output:

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

Task:
Reads the current value of timer.

Calling convention:
ANSI C :

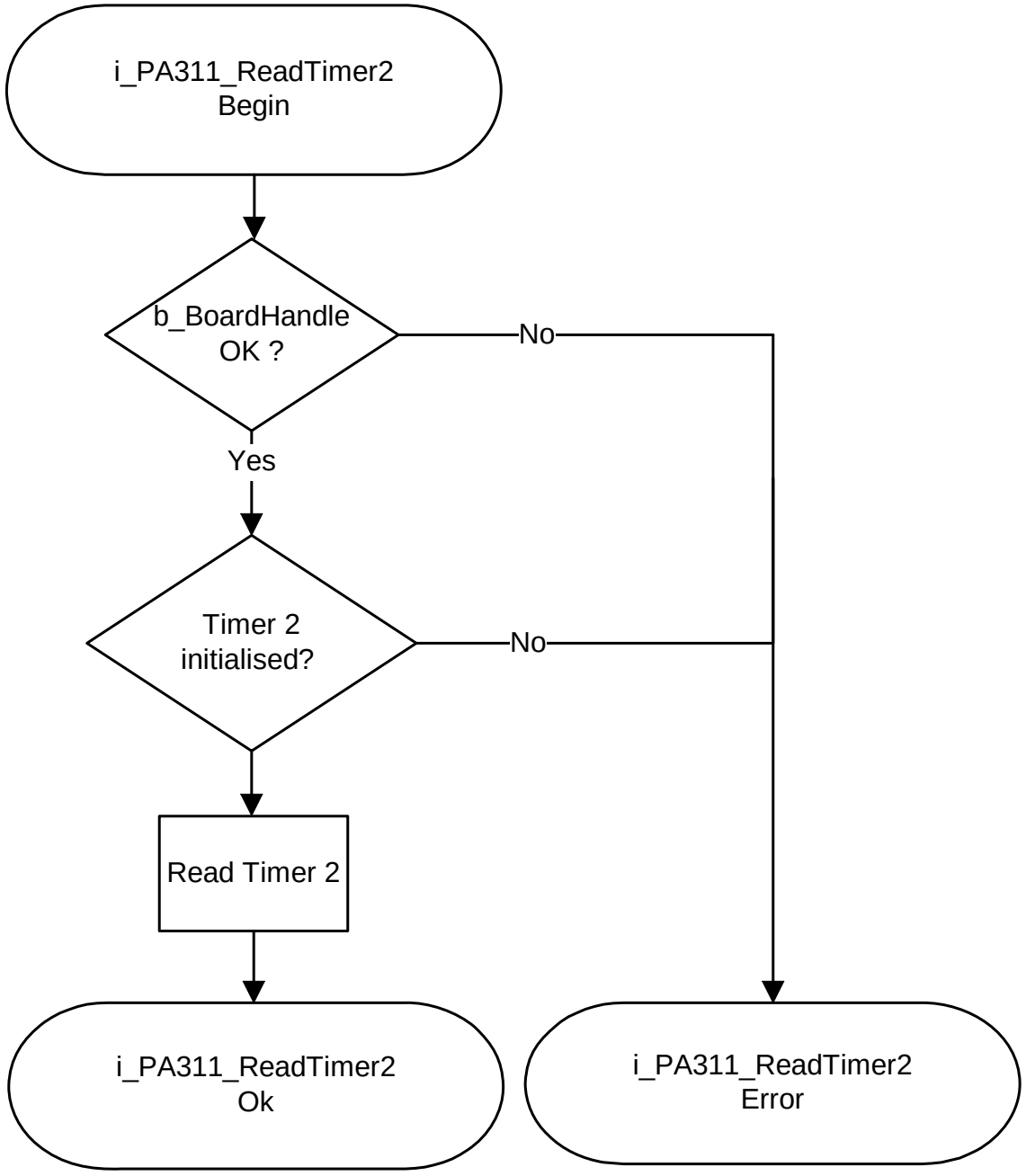
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
long            l_ReadValue;

i_ReturnValue = i_PA311_ReadTimer2      (b_BoardHandle,
                                         &l_ReadValue);
```

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

Input

b_BoardHandle pl_ReadValue

Function diagram



Output

pl_ReadValue

<Return Value>

5) i_PA311_WriteTimer2 (...)

Syntax:
<Return value> = i_PA311_WriteTimer2 (BYTEb_BoardHandle
LONG pui_ReadValue)

Parameter:
- Input:

BYTE	b_BoardHandle	Handle of board **PA 311**
LONG	l_ReadValue	New timer value (from 0 to FFFFFF Hex)

- Output:
No output signal has occurred.

Task:
Writes a new value in timer.

Calling convention:
ANSI C :

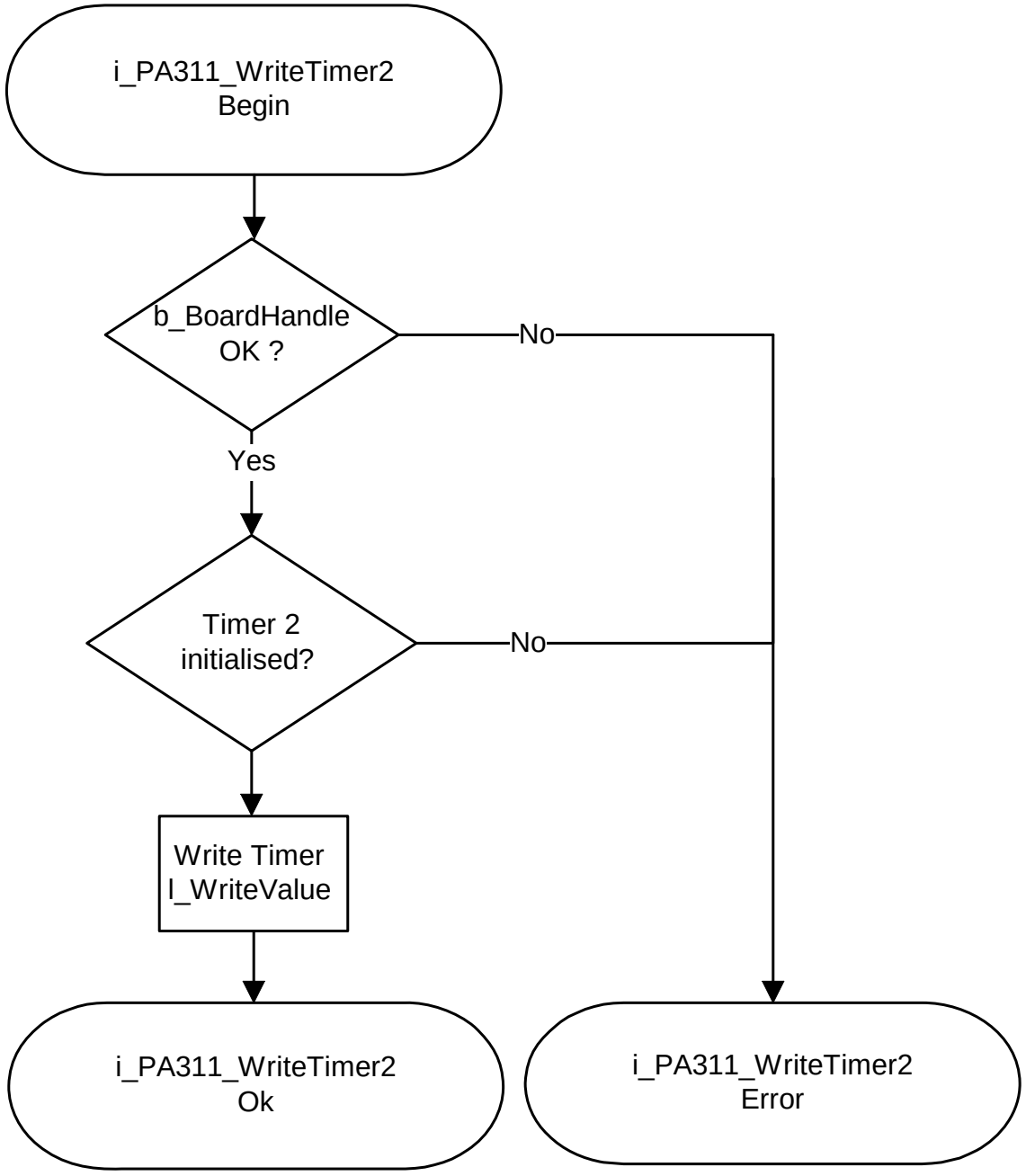
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PA311_WriteTimer2     (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>

3.7 PIO functions

1) i_PA311_InitPIO (...)

Syntax:

```
<Return value> = i_PA311_InitPIO    (BYTE   b_BoardHandle,
                                     BYTE   b_PortAMode,
                                     BYTE   b_PortBMode,
                                     BYTE   b_PortCLowMode,
                                     BYTE   b_PortCHighMode)
```

Parameter:

- Input:

BYTE	b_BoardHandle	Handle of board **PA311**
BYTE	b_PortAMode	Selection of the mode for port A 0 : Port A used for output 1 : Port A used for input
BYTE	b_PortBMode	Selection of the mode for port B 0 : used for output 1 : used for input
BYTE	b_PortCLowMode	Selection of low mode for Port C 0 : used for output 1 : used for input
BYTE	b_PortCHighMode	Selection of high mode for Port C 0 : used for output 1 : used for input

- Output:

No output signal has occurred.

Task:

Initialises the PIO (82C55A).

Calling convention:

ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PA311_InitPIO     (b_BoardHandle,
                                     0,
                                     0,
                                     0,
                                     0);
```

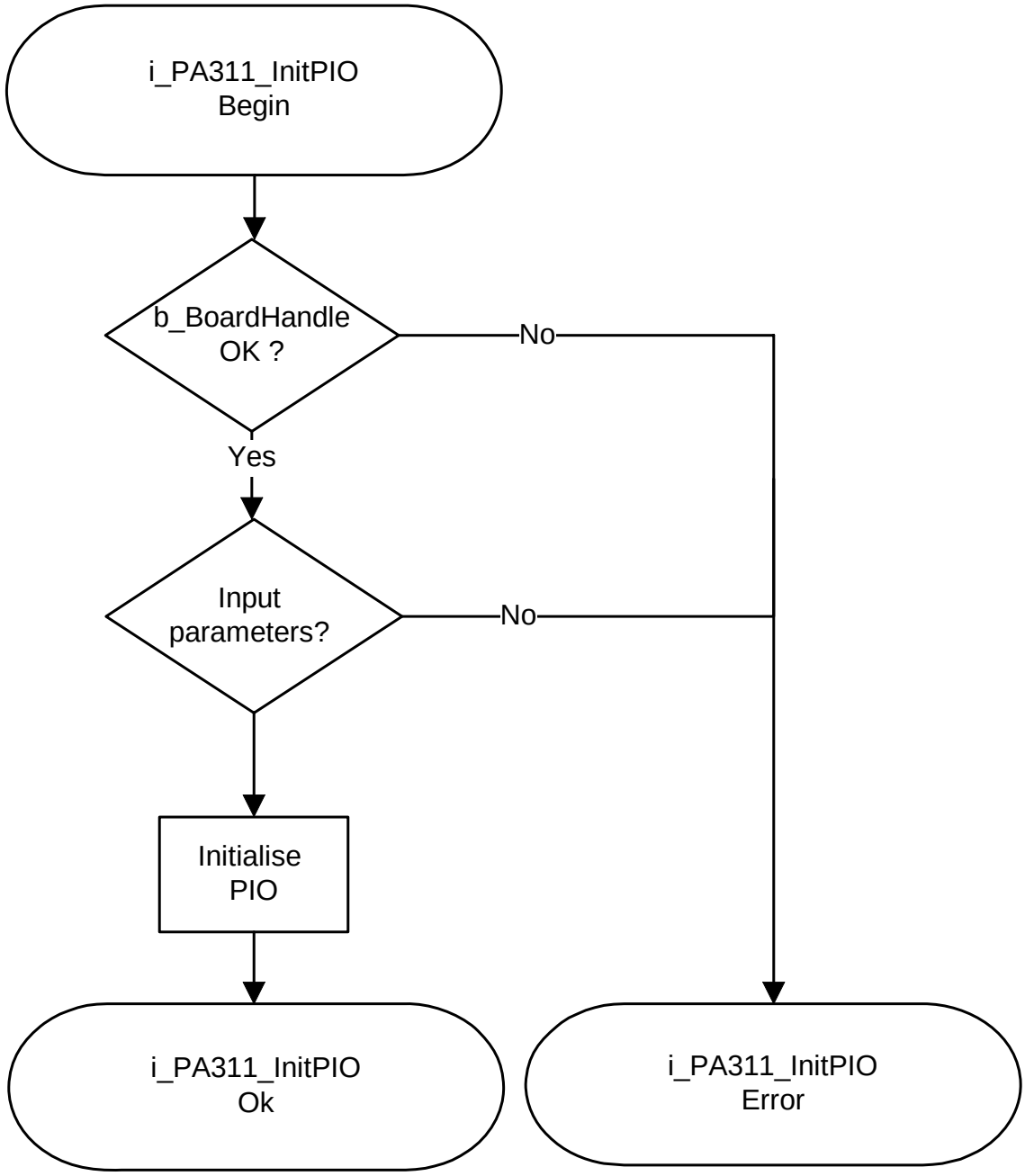
Return value:

 0: No error
-1: The handle parameter of the board is wrong
-2: Mode selection for port A is wrong
-3: Mode selection for port B is wrong
-4: Selection of low mode for port C is wrong
-5: selection of high mode for port C is wrong

Input

b_BoardHandle
b_PortAMode
b_PortBMode
b_PortCLowMode
b_PortCHighMode

Function diagram



Output

<Return Value>

2) i_PA311_ReadPIO (...)

Syntax:
<Return value> = i_PA311_ReadPIO
(BYTE b_BoardHandle,
BYTE b_SelectedPort,
PBYTE pb_PortValue)

Parameter:
- Input:

BYTE	b_BoardHandle	Handle of the board **PA311**
BYTE	b_SelectedPort	selection of the port to be read 0: Port A 1: Port B 2: Port C 3: Status register

- Output:

PBYTE	pb_PortValue	Port value

Task:
Reads the selected port value.

Calling convention:
ANSI C :

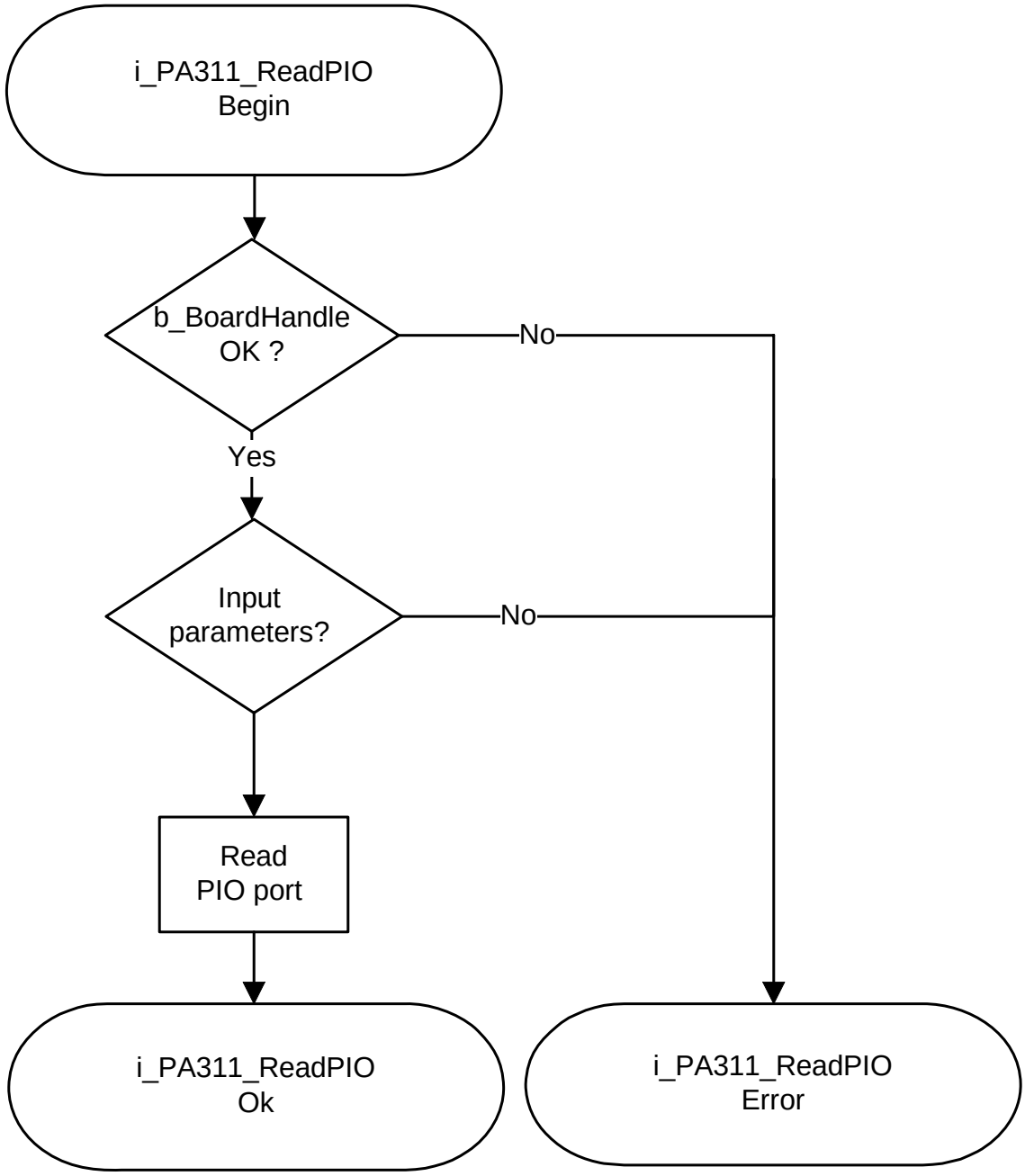
```
int             i_ReturnValue;
unsigned char   b_BoardHandle;
unsigned char   b_PortValue;

i_ReturnValue = i_PA311_ReadPort  (b_BoardHandle,
                                   0,
                                   &b_PortValue);
```

Return value:
0: No error.
-1: The handle parameter of the board is wrong.
-2: Port selection error.

Input

b_BoardHandle pb_PortValue
b_SelectedPort

Function diagram



Output

pb_PortValue

<Return Value>

3) i_PA311_WritePIO (...)

Syntax:

```
<Return value> = i_PA311_WritePIO   (BYTE    b_BoardHandle,
                                     BYTE    b_SelectedPort,
                                     BYTE    b_WriteValue)
```

Parameter:

- Input:

BYTE	b_BoardHandle	Handle of the board **PA311**
BYTE	b_SelectedPort	Selection of the port to be written on 0: Port A 1: Port B 2: Port C 3: Commando register
BYTE	b_WriteValue	Value to be written

- Output:

No output signal has occurred.

Task:

Write the value to the selected port.

Calling convention:

ANSI C :

```
int             i_ReturnValue;
unsigned char   b_BoardHandle;

i_ReturnValue = i_PA311_WritePort  (b_BoardHandle,
                                    0,
                                    0x55);
```

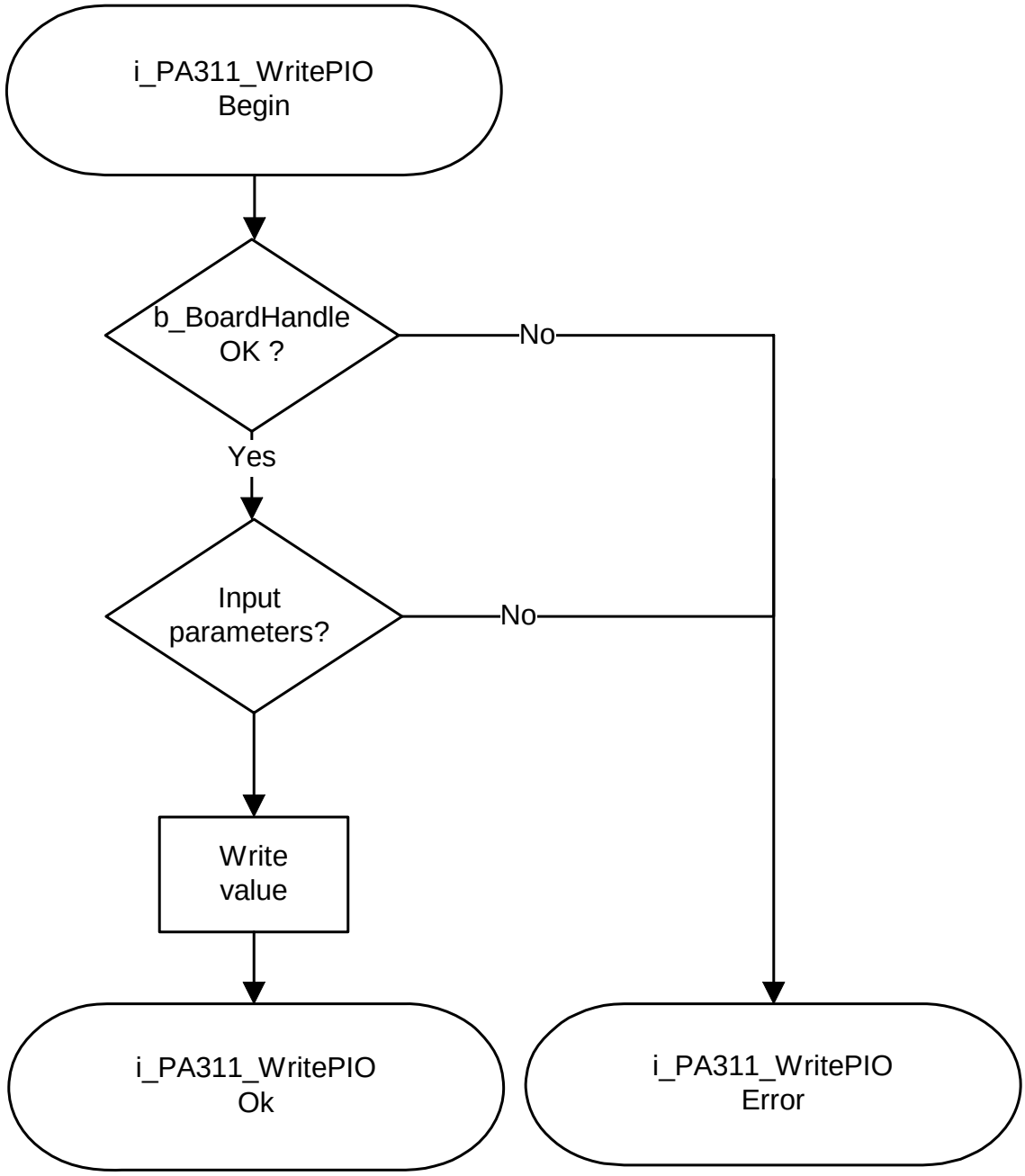
Return value:

0: No error.
-1: The handle parameter of the board is wrong.
-2: Port selection error.

Input

b_BoardHandle b_WriteValue
b_SelectedPort

Function diagram



Output

<Return Value>

3.8 Function to use in KERNEL Mode

1) i_PA311_KRNL_Write1AnalogValue (...)

Syntax:

```
<Return value> =  i_PA311_KRNL_Write1AnalogValue
                                  (UINT      ui_Address,
                                   BYTE      b_ChannelNbr,
                                   UINT      ui_ValueToWrite)
```

Parameter:

- Input:

UINT	ui_Address	Address of the board **PA311**
BYTE	b_ChannelNbr	Number of the analog output (0 to 7)
UINT	ui_ValueToWrite	Analog output value to be written 0 to 65535 (16-bit)

- Output:

No output signal has occurred.

Task:

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

Calling convention:

ANSI C :

```
int              i_ReturnValue;

i_ReturnValue = i_PA311_KRNL_Write1AnalogValue    (0x390,
                                                   0,
                                                   4095);
```

Return value:

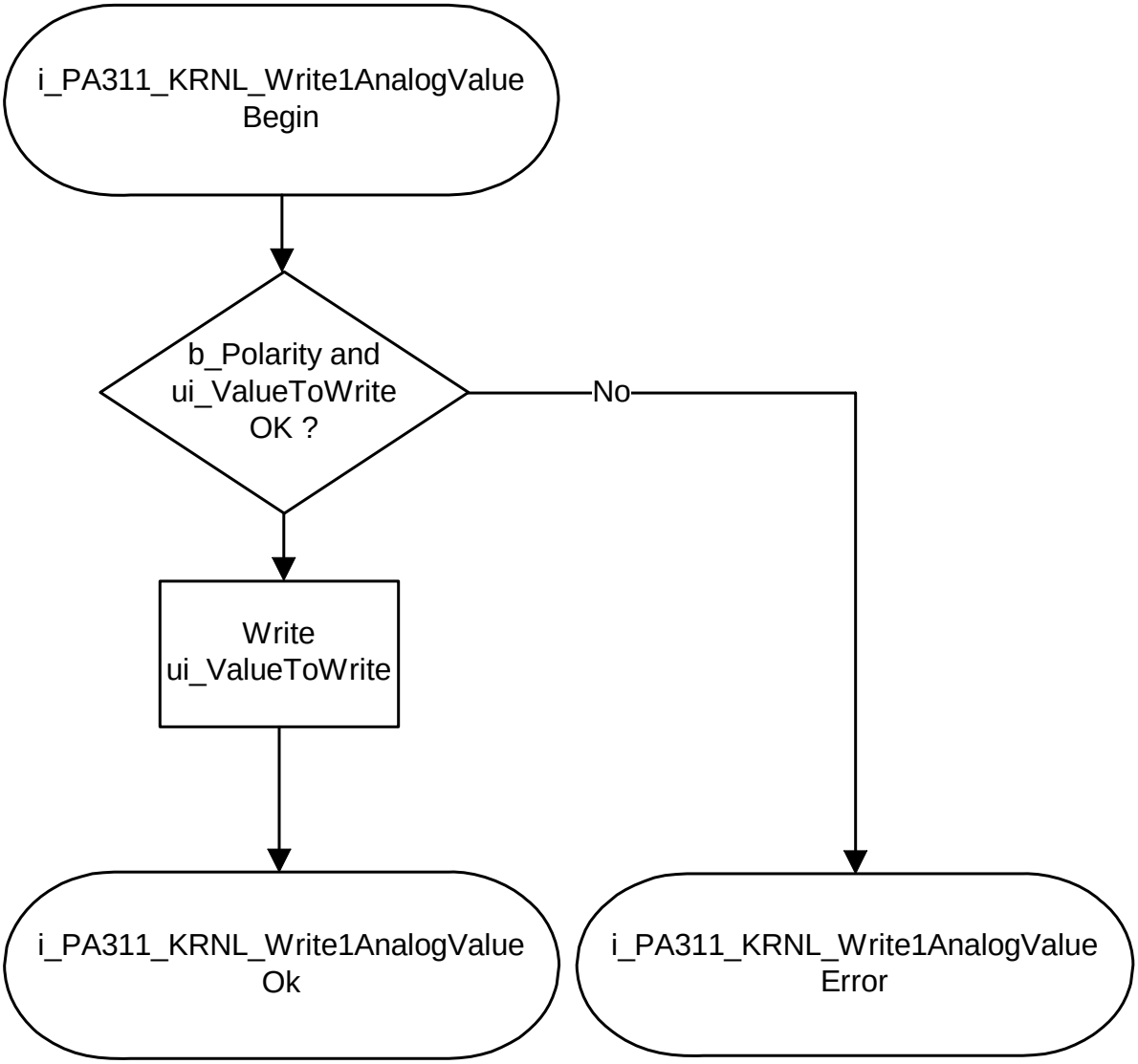
0: No error

-1: Output value too high

Input

ui_Address ui_ValueToWrite
b_ChannelNbr

Function diagram



Output

<Return Value>

2) i_PA311_KRNL_ReadPIO (...)

Syntax:
<Return value> = i_PA311_KRNL_ReadPIO
(UINT ui_Address,
BYTE b_SelectedPort,
PBYTE pb_PortValue)

Parameter:
- Input:

UINT	ui_Address	Address of the board **PA311**
BYTE	b_SelectedPort	Selection of the port to be read 0: Port A 1: Port B 2: Port C 3: Status register

- Output:

PBYTE	pb_PortValue	Port value

Task:
Reads the selected port value.

Calling convention:
ANSI C :

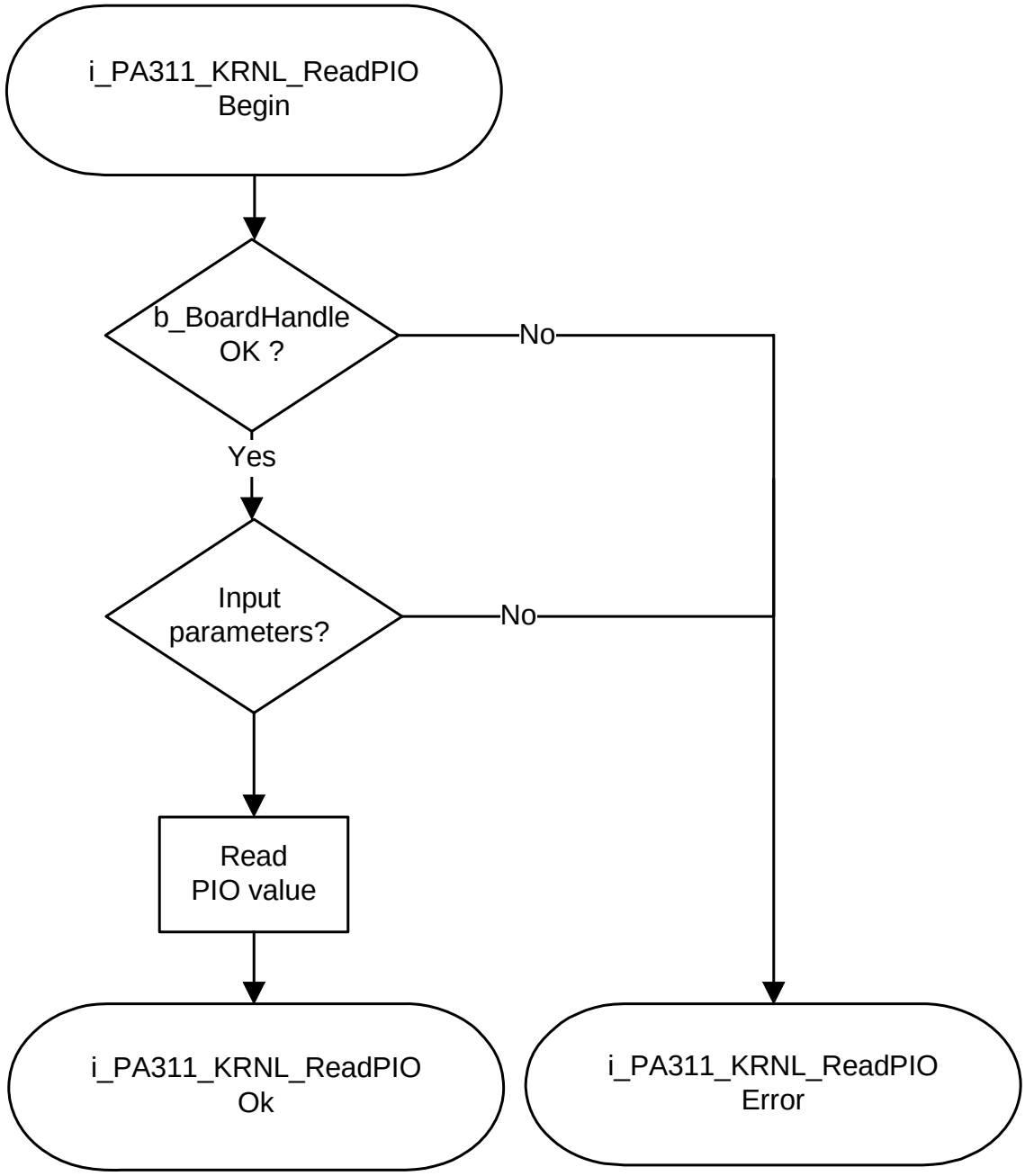
```
int             i_ReturnValue;
unsigned char   b_PortValue;

i_ReturnValue = i_PA311_KRNL_ReadPort (0x390,
                                       0,
                                       &b_PortValue);
```

Return value:
0: No error.
-1: Port selection error.

Input

ui_Address pb_PortValue
b_SelectedPort

Function diagram



Output

pb_PortValue

<Return Value>

3) i_PA311_KRNL_WritePIO (...)

Syntax:
<Return value> = i_PA311_KRNL_WritePIO
(UINT ui_Address,
BYTE b_SelectedPort,
BYTE b_WriteValue)

Parameter:
- Input:

UINT	ui_Address	Address of the board **PA311**
BYTE	b_SelectedPort	Slection of the port to be ion to read 0 : Port A 1 : Port B 2 : Port C 3 : Commando register
BYTE	b_WriteValue	Value to be written

- Output:
No output signal has occurred.

Task:
Writes the value to the selected port

Calling convention:
ANSI C :

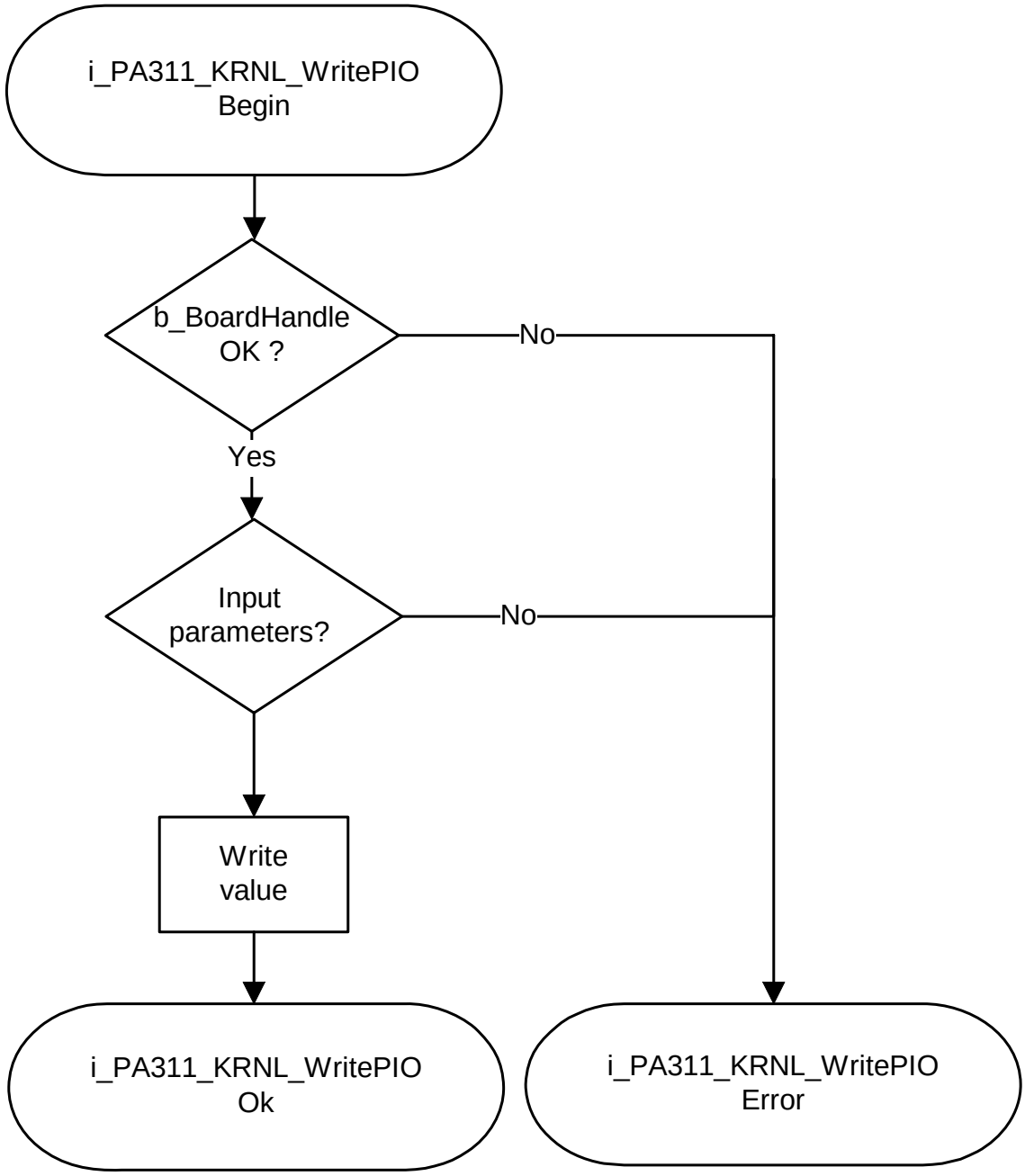
```
int                 i_ReturnValue;

i_ReturnValue = i_PA311_KRNL_WritePort (0x390,
                                        0,
                                        0x55);
```

Return value:
0: No error.
-1: Port selection error.

Input

ui_Address b_WriteValue
b_SelectedPort

Function diagram



Output

<Return Value>