



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



Technical description

ADDIALOG PA 3500

Standard software / API functions

1	INTRODUCTION	1
2	DIN 66001- GRAPHICAL SYMBOLS	3
3	SOFTWARE FUNCTIONS (API)	4
3.1	Initialisation	4
	1) i_PA3500_InitCompiler (..)	4
	2) i_PA3500_SetBoardInformation (...)	6
	3) i_PA3500_SetBoardInformationWin32 (...)	9
	4) i_PA3500_GetHardwareInformation (...)	11
	5) i_PA3500_ChangeBoardOperatingMode (...).....	13
	6) i_PA3500_CloseBoardHandle (..)	15
3.2	Interrupt	17
	1) i_PA3500_SetBoardIntRoutineDos (..)	17
	2) i_PA3500_SetBoardIntRoutineVBDos (..).....	20
	3) i_PA3500_SetBoardIntRoutineWin16 (..).....	23
	4) i_PA3500_SetBoardIntRoutineWin32 (..).....	26
	5) i_PA3500_TestInterrupt (..).....	33
	6) i_PA3500_ResetBoardIntRoutine (..).....	35
3.3	Analog output channels	37
	1) i_PA3500_Write1AnalogValue (...)	37
	2) i_PA3500_WriteMoreAnalogValue (...).....	40
	3) i_PA3500_StoreAnalogOutputValue (...).....	43
	4) i_PA3500_GetExternTriggerStatus (...).....	46
	5) i_PA3500_SimulateExternalTrigger (...).....	48
	6) i_PA3500_EnableTriggerInterrupt (...).....	50
	7) i_PA3500_DisableTriggerInterrupt (...).....	52
3.4	Watchdog	54
	1) i_PA3500_EnableWatchdog (...)	54
	2) i_PA3500_GetWatchdogStatus (...).....	56
	3) i_PA3500_TriggerWatchdog (...).....	58
	4) i_PA3500_DisableWatchdog (...)	60
3.5	Digital output channels	62
	1) i_PA3500_SetOutputMemoryOn (...)	62
	2) i_PA3500_SetOutputMemoryOff (...)	64
	3) i_PA3500_Set1DigitalOutputOn (...).....	66
	4) i_PA3500_Set1DigitalOutputOff (...).....	68
	5) i_PA3500_Set2DigitalOutputOn (...).....	70
	6) i_PA3500_Set2DigitalOutputOff (...).....	72
3.6	Digital input channels	74
	1) i_PA3500_Read1DigitalInput (...)	74
	2) i_PA3500_Read2DigitalInput (...)	76
	3) i_PA3500_EnableDigitalInputInterrupt (...)	78
	4) i_PA3500_DisableDigitalInputInterrupt (...)	80
	5) i_PA3500_ReadExternTriggerInput (...).....	82

3.7 Functions in KERNEL Mode.....84

- 1) i_PA3500_KRNL_Write1AnalogValue (...).....84
- 2) i_PA3500_KRNL_StoreAnalogOutputValue (...)86
- 3) i_PA3500_KRNL_GetExternTriggerStatus (...)88
- 4) i_PA3500_KRNL_SimulateExternalTrigger (...).....90
- 5) i_PA3500_KRNL_GetWatchdogStatus (...)92
- 6) i_PA3500_KRNL_TriggerWatchdog (...).....94
- 7) i_PA3500_KRNL_Set1DigitalOutputOn (...)96
- 8) i_PA3500_KRNL_Set2DigitalOutputOn (...)98
- 9) i_PA3500_KRNL_Read1DigitalInput (...).....100
- 10) i_PA3500_KRNL_Read2DigitalInput (...).....102
- 11) i_PA3500_KRNL_ReadExternTriggerInput (...).....104

Tables

- Table 1-1: Type Declaration for Dos and Windows 3.1X..... 1
- Table 1-2: Type Declaration for Windows 95/NT..... 1
- Table 1-3: Define value..... 2

1 INTRODUCTION

i

IMPORTANT!

Note the following conventions in the text:

Function: "i_PA3500_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



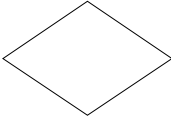

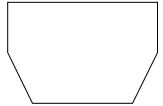
Define name	Decimal value	Hexadecimal value
DLL_COMPILER_C	0	0
DLL_COMPILER_VB	1	1
DLL_COMPILER_PASCAL	2	2
DLL_LABVIEW	3	3
DLL_COMPILER_VB_5	4	4
PA3500_DISABLE	0	0
PA3500_ENABLE	1	1
PA3500_CHANNEL0	0	0
PA3500_CHANNEL1	1	1
PA3500_CHANNEL2	2	2
PA3500_CHANNEL3	3	3
PA3500_CHANNEL4	4	4
PA3500_CHANNEL5	5	5
PA3500_CHANNEL6	6	6
PA3500_CHANNEL7	7	7
PA3500_UNIPOLAR	1	1
PA3500_BIPOLAR	0	0
PA3500_SYNCHRONOUS_MODE	1	1
PA3500_ASYNCHRONOUS_MODE	0	0
PA3500_SIMPLE_MODE	0	0
PA3500_TRIGGER_MODE	1	1
PA3500_SYNCHRONISATION_MODE	2	2

2 DIN 66001- GRAPHICAL SYMBOLS

This chapter describes all software functions (API) necessary for the operation of the **APCI-3001** 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)
	Terminator (eg. Beginning or end of a sequence, origin or place of data)
	Decision Selection unit (eg.: switch)
	Loop limit Beginning
	Loop limit End

3 SOFTWARE FUNCTIONS (API)

3.1 Initialisation

1) i_PA3500_InitCompiler (..)

Syntax:

<Return value> = i_PA3500_InitCompiler (BYTE b_CompilerDefine)

Parameters:

- Input:

BYTE b_CompilerDefine

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

- DLL_COMPILER_C:

The user programs in C.

- DLL_COMPILER_VB:

The user programs in Visual Basic for Windows.

- DLL_COMPILER_VB_5:

The user programs in Visual Basic 5 for Windows 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.

i

IMPORTANT!

This function is only available with a Windows environment.

Calling convention:

ANSI C:

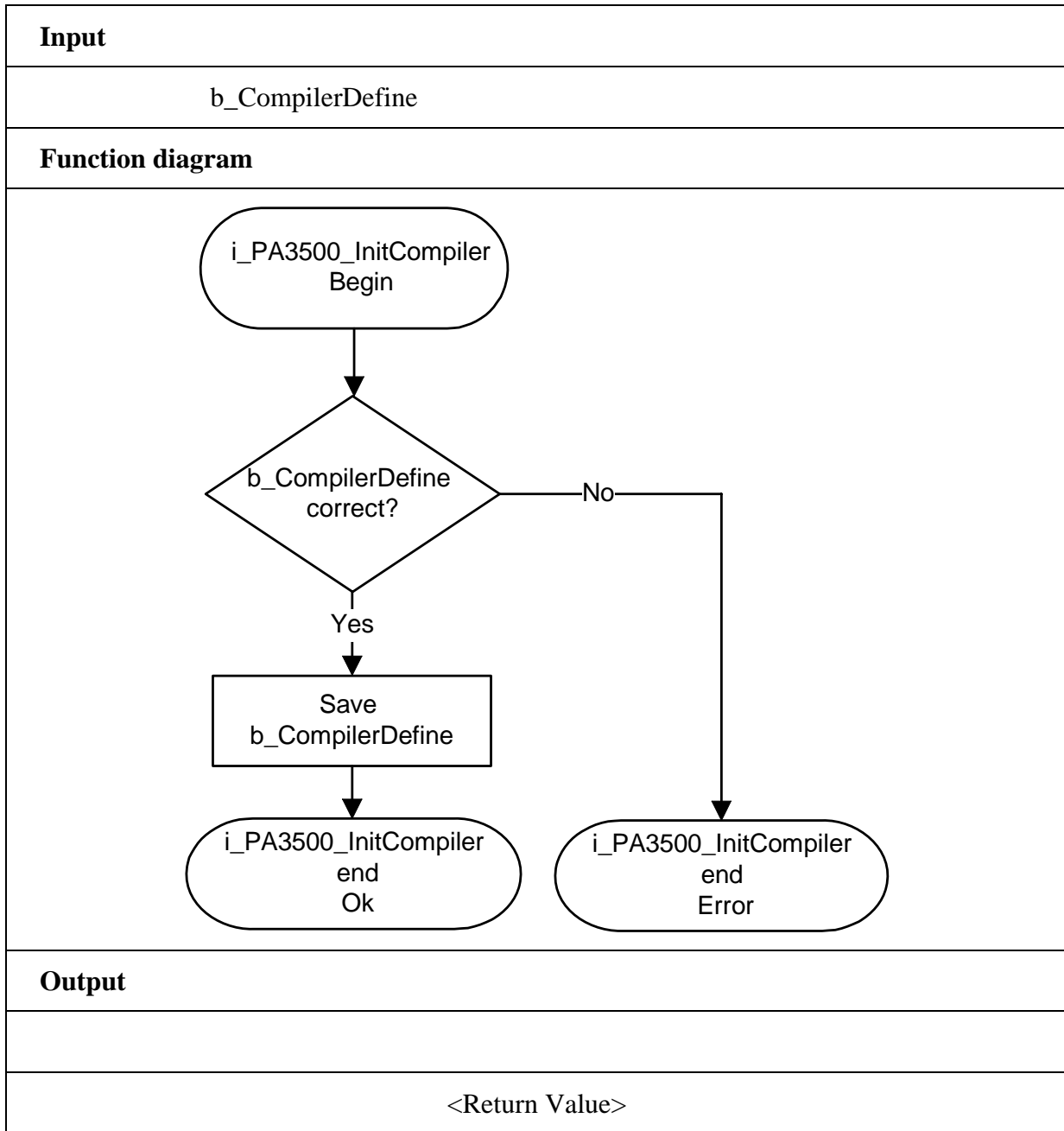
int i_ReturnValue;

i_ReturnValue = i_PA3500_InitCompiler (DLL_COMPILER_C);

Return value:

0: No error

-1: Compiler parameter is wrong



i**IMPORTANT!**

This function is only available for DOS and Windows 3.11 applications

2) i_PA3500_SetBoardInformation (...)**Syntax:**

```
<Return value> = i_PA3500_SetBoardInformation
                    (UINT    ui_BaseAddress,
                     BYTE    b_InterruptNumber,
                     BYTE    b_BoardOperatingMode,
                     PBYTE   pb_BoardHandle)
```

Parameters:**- Input:**

UINT	ui_BaseAddress	Base address of PA 3500 board
BYTE	b_InterruptNumber	Interrupt line of the board (IRQ 3,4,5,6,7,9,10,11,12,14,15). When 0: Interrupt lines are not used.
BYTE	b_BoardOperatingMode	Selects the board operating mode. See table 3-1.

- Output:

PBYTE	pb_BoardHandle	Handle ¹ of board PA 3500 to use the functions
-------	----------------	---

Task:

Verifies if the board **PA 3500** is present. Stores the following information:

- base address,
- the interrupt number,
- the board operating mode.

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;
```

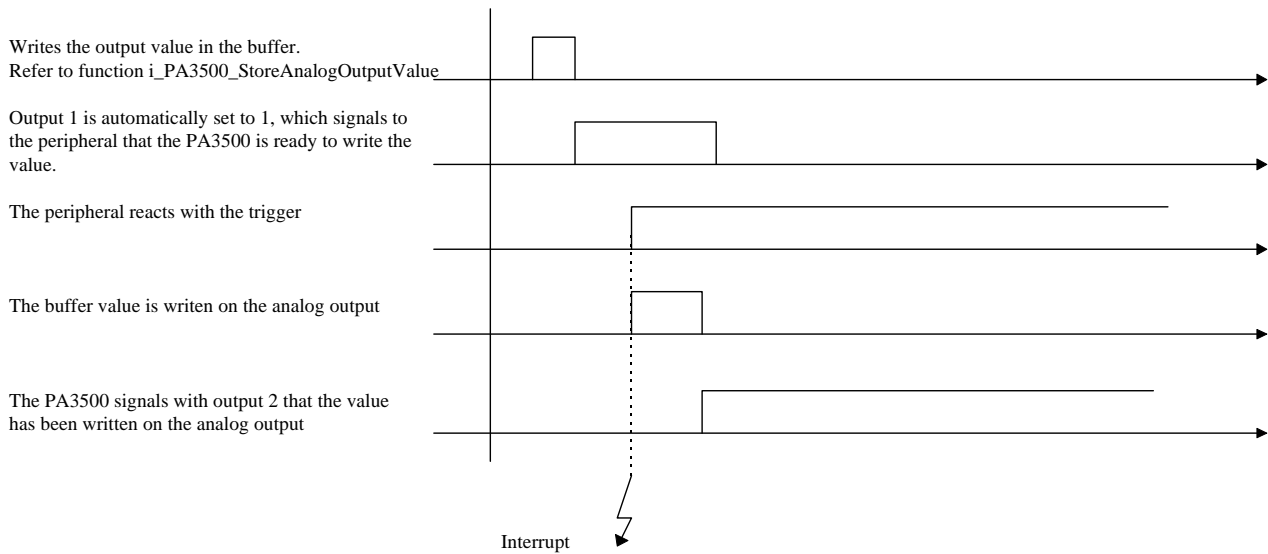
```
i_ReturnValue = i_PA3500_SetBoardInformation (0x390,
                                              10,
                                              PA3500_TRIGGER_MODE,
                                              &b_BoardHandle);
```

¹ Identification number of the board

Table 3-1: Board operating mode

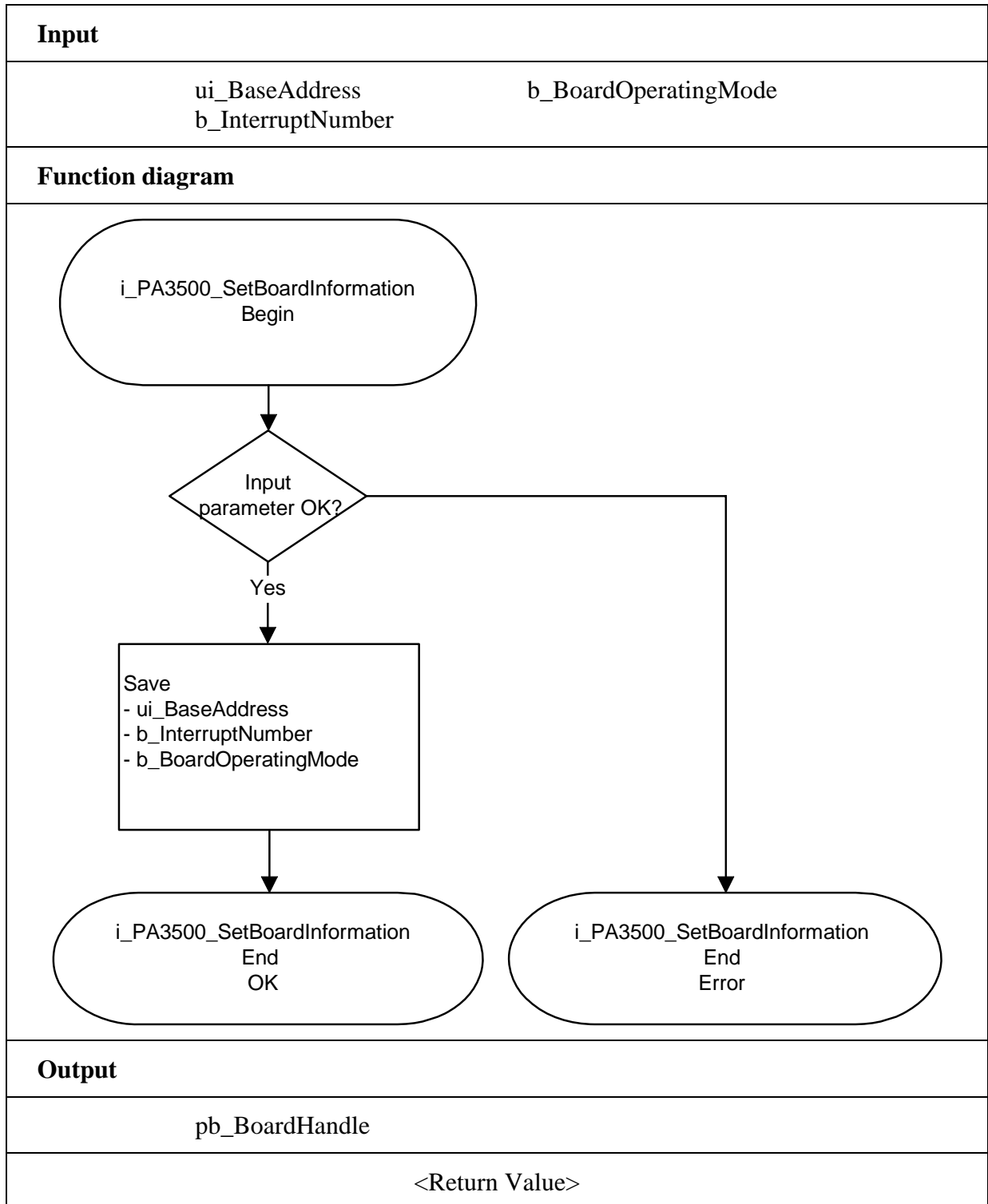
<i>b_BoardOperatingMode</i>	Description of the operating mode
PA3500_SIMPLE_MODE	This mode writes the analog value directly on the output channel.
PA3500_TRIGGER_MODE	This mode writes the analog value into a buffer. This buffer value is written on the output channel when an external trigger occurs.
PA3500_SYNCHRONISATION_MODE	This mode writes the analog value into a buffer. This buffer value is written on the output channel when an external trigger occurs. Digital channels 1 and 2 are used for the synchronisation with the peripheral. See figure below. Interrupts must be selected in this mode.

Synchronisation operating mode



Return value:

- 0: No error
- 1: Base address already used
- 2: Board no present
- 3: Wrong interrupt number
- 4: Interrupt is already used by another PA 3500
- 5: Operating mode is wrong
- 6: No handle is available for the board (up to 10 handles can be used)



i**IMPORTANT!**

This function is only available for Windows NT / 95 applications

3) i_PA3500_SetBoardInformationWin32 (...)**Syntax:**

```
<Return value> = i_PA3500_SetBoardInformationWin32
                    (PCHAR    pc_Identifier,
                     BYTE     b_BoardOperatingMode,
                     PBYTE    pb_BoardHandle)
```

Parameters:**- Input:**

PCHAR	pc_Identifier	Identifier string for the selection of a PA 3500 board. The identifier string is determined with the ADDIREG registration program.
BYTE	b_BoardOperatingMode	Selects the board operating mode. (see table 3-1)

- Output:

PBYTE	pb_BoardHandle	Handle ¹ of board PA 3500 to use the functions
-------	----------------	--

Task:

Calls all hardware information about the **PA 3500** from the **ADDIREG** registration program and stores the following information:

- the base address,
 - the interrupt number,
 - the board operating mode
- and verifies if board **PA 3500** is present.

A handle is returned to the user which allows to use the next functions.

Handles allow to operate several boards.

Calling convention:

ANSI C :

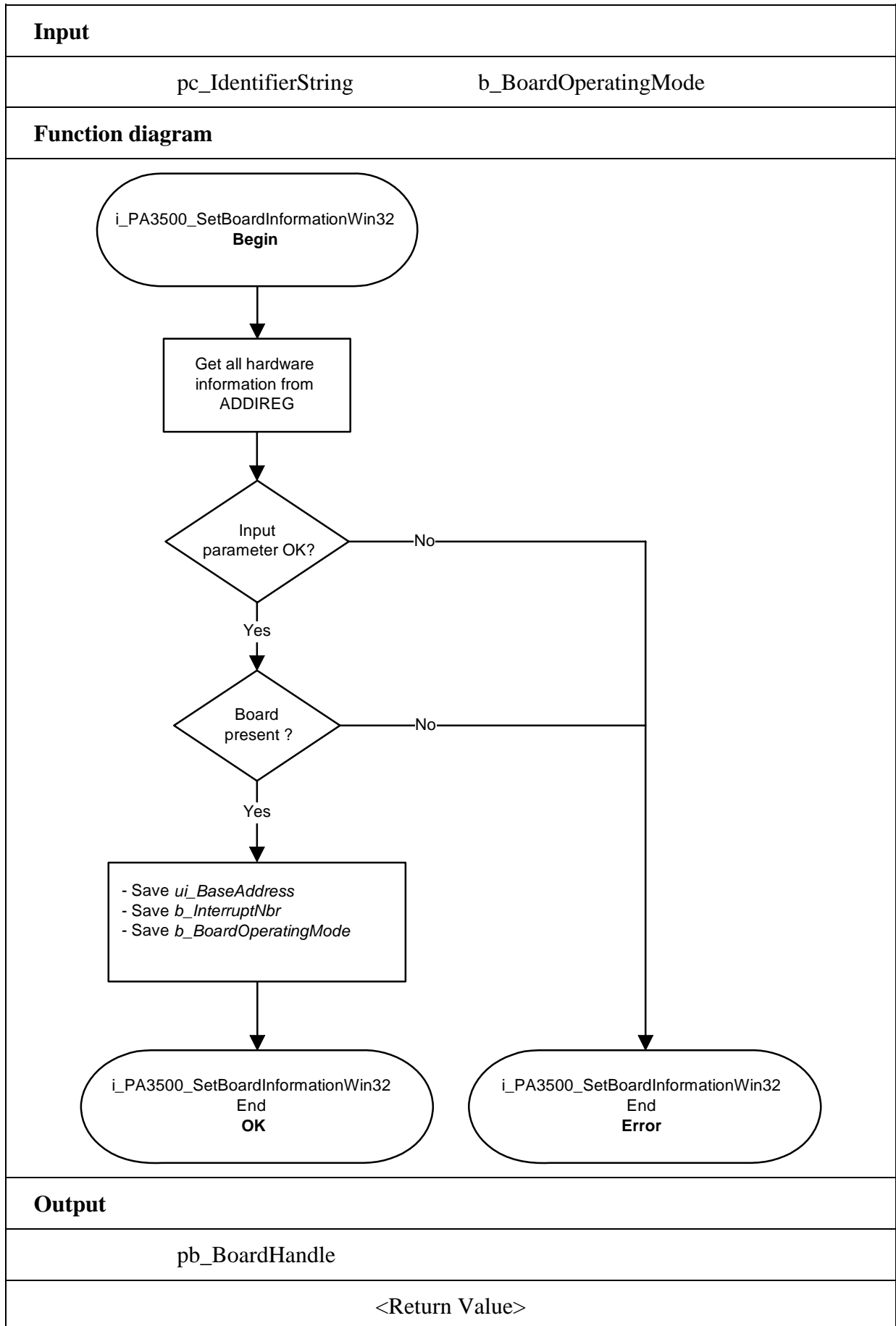
```
int          i_ReturnValue;
unsigned char b_BoardHandle;
```

```
i_ReturnValue = i_PA3500_SetBoardInformationWin32
                ("PA3500-00",
                 PA3500_SIMPLE_MODE,
                 &b_BoardHandle);
```

Return value:

- 0: No error
- 2: Board not present
- 4: Interrupt is already used by another PA 3500
- 5: Board operating mode is wrong
- 6: No handle is available for the board (up to 10 handles can be used)
- 7: Error when opening the driver under Windows NT/95

¹ Identification number of the board



4) i_PA3500_GetHardwareInformation (...)**Syntax:**

```
<Return value> = i_PA3500_GetHardwareInformation
                    (BYTE      b_BoardHandle,
                     PUINT     pui_BaseAddress,
                     PBYTE     pb_InterruptNbr,
                     PBYTE     pb_BoardOperatingMode,
                     PBYTE     pb_NbrOfAnalogOutput)
```

Parameters:**- Input:**

BYTE b_BoardHandle Handle of board **PA 3500**

- Output:

PUINT pui_BaseAddress **PA 3500** base address

PBYTE pb_InterruptNbr **PA 3500** interrupt channel.

PBYTE pb_BoardOperatingMode Board operating mode.

See table 3-1

PBYTE pb_NbrOfAnalogOutput Number of analog outputs

Task:

Returns the base address, the interrupt number, the number of analog outputs and the operating mode of the **PA 3500**.

Calling convention:ANSI C :

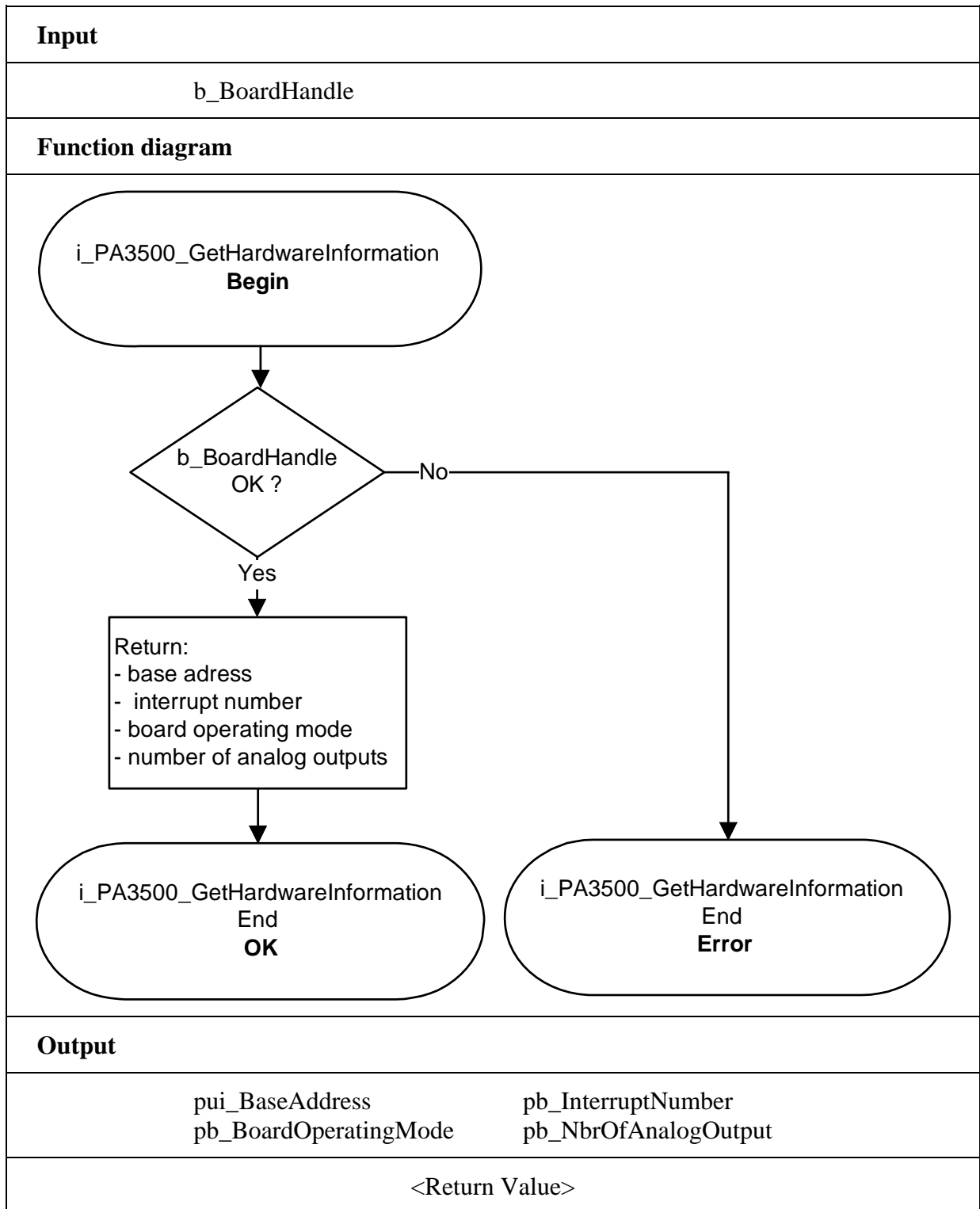
```
int          i_ReturnValue;
unsigned int  ui_BaseAddress;
unsigned char b_InterruptNbr;
unsigned char b_BoardHandle;
unsigned char b_BoardOperatingMode;
unsigned char b_OutputNumber;
```

```
i_ReturnValue = i_PA3500_GetHardwareInformation (b_BoardHandle,
                                                &ui_BaseAddress,
                                                &b_InterruptNbr,
                                                &b_BoardOperatingMode,
                                                &b_OutputNumber);
```

Return value:

0: No error

-1: The handle parameter of the board is wrong



5) i_PA3500_ChangeBoardOperatingMode (...)

Syntax:

```
<Return value> = i_PA3500_ChangeBoardOperatingMode  
                  (BYTE   b_BoardHandle,  
                  BYTE   b_BoardOperatingMode)
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of board PA 3500
BYTE	b_BoardOperatingMode	Selects the board operating mode.

- Output:

No output signal has occurred.

Task:

Changes the operating mode of the board.

Calling convention:

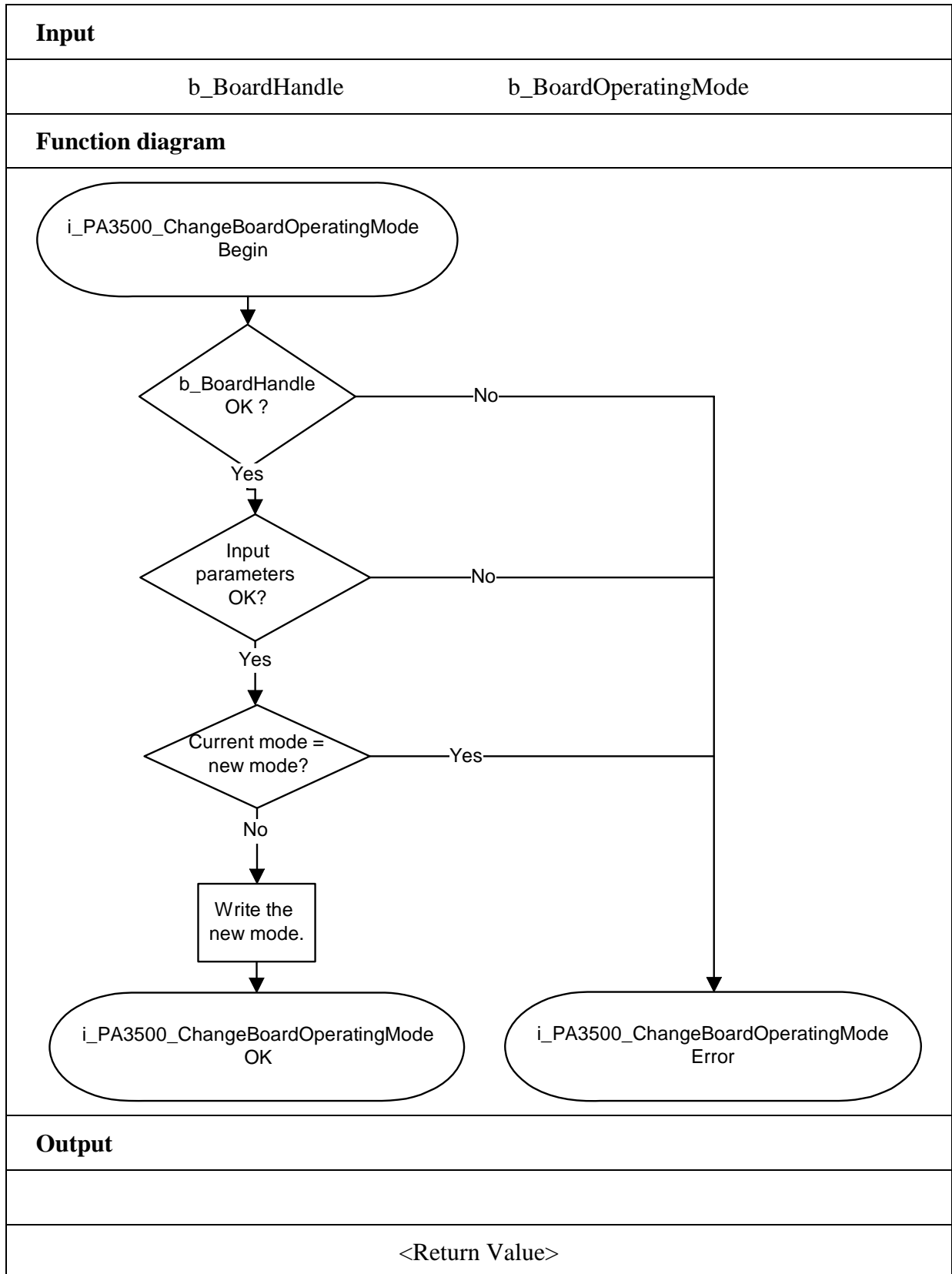
ANSI C :

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

```
i_ReturnValue = i_PA3500_ChangeBoardOperatingMode  
                (b_BoardHandle,  
                PA3500_SIMPLE_MODE)
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The board operating mode is wrong.
- 3: The board operating mode is already used.



6) i_PA3500_CloseBoardHandle (..)**i****IMPORTANT!**

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

Syntax:

<Return value> = i_PA3500_CloseBoardHandle (BYTE b_BoardHandle)

Parameters:**- Input:**

BYTE b_BoardHandle Handle of board **PA 3500**

- Output:

No output signal has occurred.

Task:

Releases the board handle and blocks the access to the board.

Calling convention:

ANSI C:

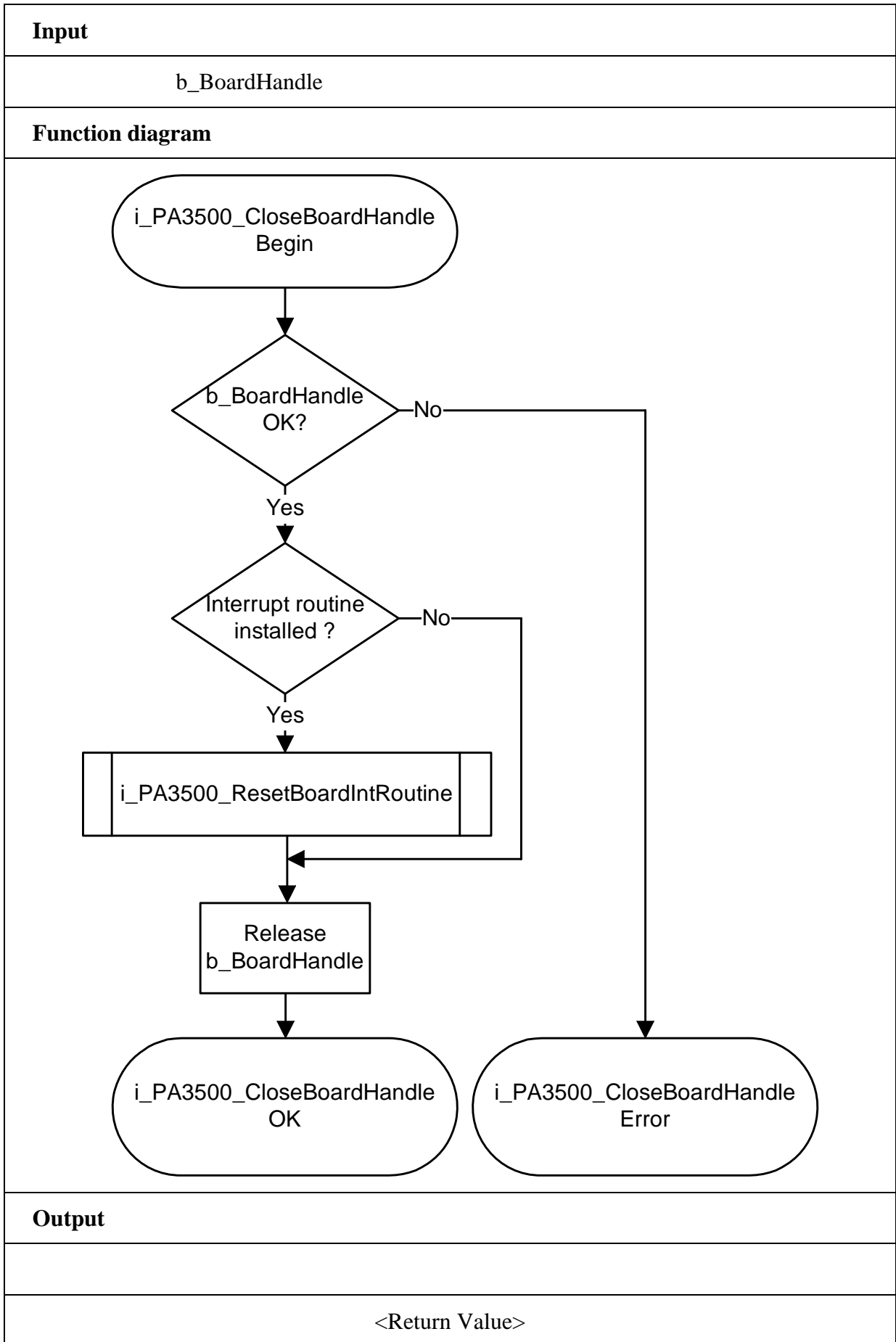
```
int                    i_ReturnValue;  
unsigned char    b_BoardHandle;
```

```
i_ReturnValue = i_PA3500_CloseBoardHandle        (b_BoardHandle);
```

Return value:

0: No error

-1: The handle parameter of the board is wrong



3.2 Interrupt

i

IMPORTANT!

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

1) i_PA3500_SetBoardIntRoutineDos (..)

Syntax:

```
<Return value> = i_PA3500_SetBoardIntRoutineDos
                (BYTE   b_BoardHandle,
                 VOID    v_FunctionName
                 (BYTE   b_BoardHandle,
                  BYTE   b_InterruptMask,
                  BYTE   b_LastChannelNumber))
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of board PA 3500
VOID	v_FunctionName	Name of the user interrupt routine

- Output:

No output signal has occurred.

Task:

This function must be called up for each **PA 3500** on which an interrupt is to be enabled. It installs one user interrupt function for all boards on which you have enabled the interrupt.

When the function is called up for the first time (first board):

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

If you operate several boards **PA 3500** which have to react to interrupts, call up the function as often as you operate boards **PA 3500**. The variable *v_FunctionName* is only relevant when the function is called up **for the first time**.

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,
                           BYTE   b_LastChannelNumber)
```

v_FunctionName Name of the user interrupt routine
b_BoardHandle Handle of the **PA 3500** which has generated the interrupt
b_InterruptMask Mask of the events which have generated the interrupt
b_LastChannelNumber Number of the last channel that has been triggered.

Table 3-3: Interrupt mask

Mask	Meaning	<i>b_LastChannelNumber</i>
0000 0001	Interrupt of digital input 1	Not used
0000 0010	Interrupt of digital input 2	Not used
0000 0100	Interrupt of external trigger	Number of the last channel that has been triggered (0 to 7)
0000 1000	Watchdog has run down	Not used

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

Calling convention:

ANSI C :

```

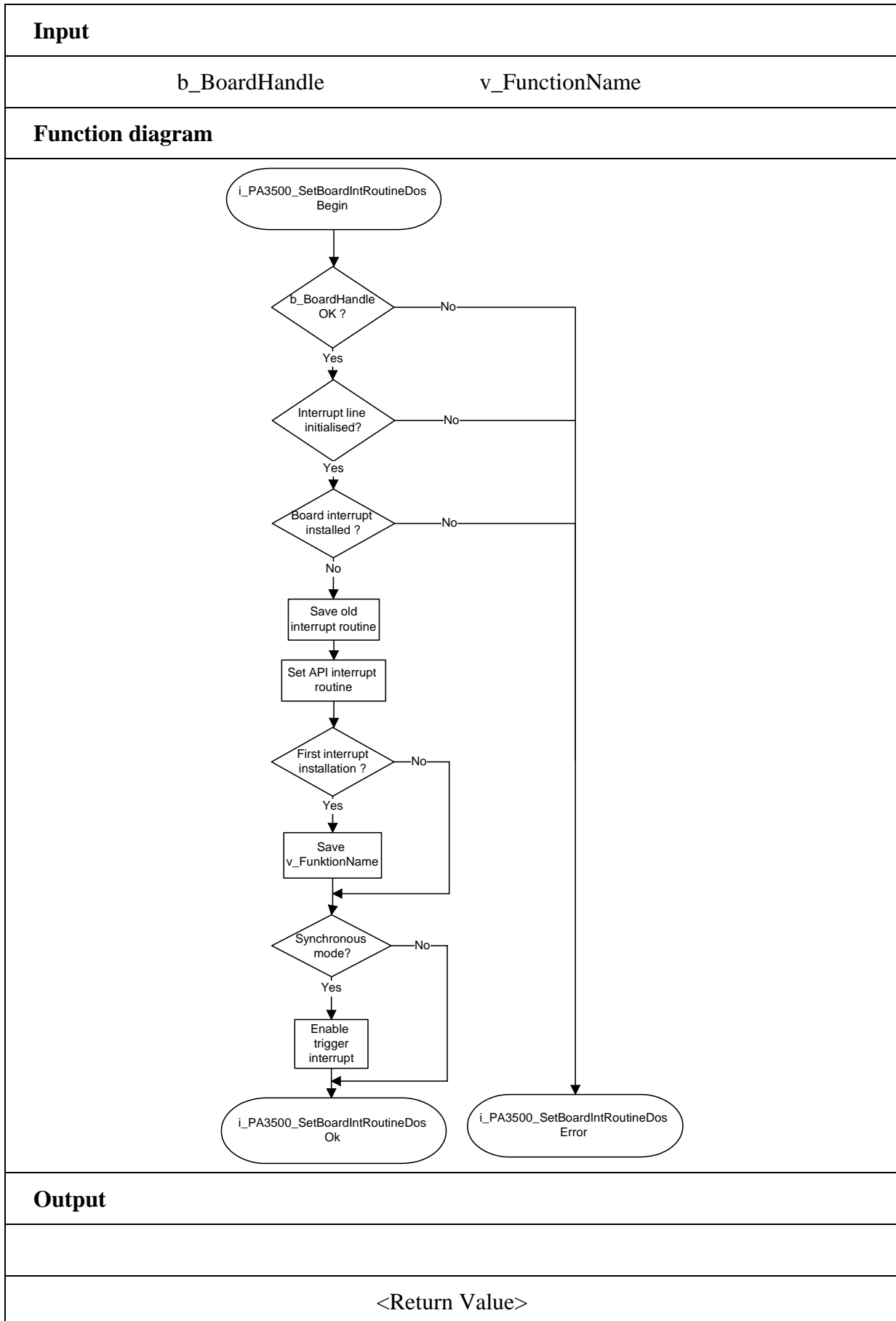
void    v_FunctionName    (unsigned char b_BoardHandle,
                          unsigned char b_InterruptMask,
                          unsigned char b_LastChannelNumber)
    {
        .
        .
    }

int      i_ReturnValue;
unsigned char b_BoardHandle;

i_ReturnValue = i_PA3500_SetBoardIntRoutineDos (b_BoardHandle,
                                                v_FunctionName );
  
```

Return value:

0: No error
 -1: The handle parameter of the board is wrong
 -2: No interrupt line has been initialised
 -3: Interrupt already installed



i**IMPORTANT!**

This function is only available for Visual Basic DOS.

2) i_PA3500_SetBoardIntRoutineVBDos (..)**Syntax:**

```
<Return value> = i_PA3500_SetBoardIntRoutineVBDos
                                     (BYTE   b_BoardHandle)
```

Parameters:**- Input:**

BYTE b_BoardHandle Handle of board **PA 3500**

- Output:

No output signal has occurred.

Task:

This function must be called up for each **PA 3500** on which an interrupt is to be enabled. It installs one user interrupt function for all boards on which you have enabled the interrupt.

When the function is called up for the first time (first board):

- interrupts are enabled for the selected board.

If you operate several boards **PA 3500** which have to react to interrupts, call up the function as often as you operate on boards **PA 3500**.

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_PA3500_TestInterrupt"

This function tests the interrupt of board **PA 3500**. It is used to obtain the values of *b_BoardHandle*, *b_InterruptMask* and *b_LastChannelNumber*.

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 i_LastChannelNumber As Integer
```

IntLabel:

```
i_ReturnValue = i_PA3500_TestInterrupt (i_BoardHandle, _
                                         i_InterruptMask, _
                                         i_LastChannelNumber)
```

```
.
.
.
```

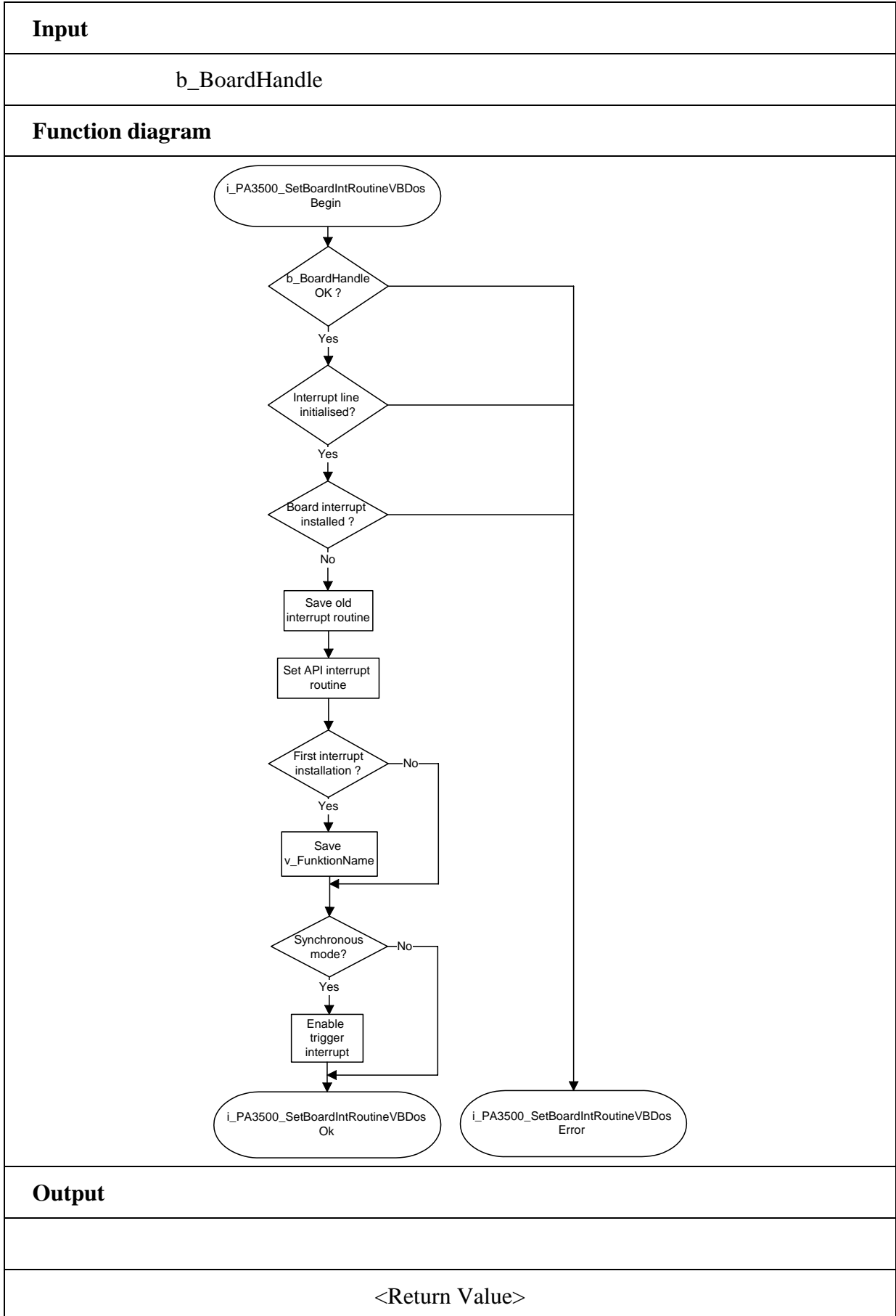
Return

```
ON UEVENT GOSUB IntLabel
UEVENT ON
```

```
i_ReturnValue = i_PA3500_SetBoardIntRoutineVBDos (b_BoardHandle)
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: No interrupt line has been initialised
- 3: Interrupt already installed



i**IMPORTANT!**

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

3) i_PA3500_SetBoardIntRoutineWin16 (..)**Syntax:**

```
<Return value> = i_PA3500_SetBoardIntRoutineWin16
                    (BYTE    b_BoardHandle,
                     VOID    v_FunctionName
                    (BYTE    b_BoardHandle,
                     BYTE    b_InterruptMask,
                     BYTE    b_LastChannelNumber))
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of board PA 3500
VOID	v_FunctionName	Name of the user interrupt routine

- Output:

No output signal has occurred.

Task:

This function must be called up for each **PA 3500** on which an interrupt is to be enabled. It installs one user interrupt function for all boards on which you have enabled the interrupt.

When the function is called up for the first time (first board):

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

If you operate several boards **PA 3500** which have to react to interrupts, call up the function as often as you operate boards **PA 3500**. 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,
                    BYTE    b_LastChannelNumber)
```

<i>v_FunctionName</i>	Name of the user interrupt routine
<i>b_BoardHandle</i>	Handle of the PA 3500 which has generated the interrupt
<i>b_InterruptMask</i>	Mask of the events which have generated the interrupt
<i>b_LastChannelNumber</i>	Number of the last channel that has been triggered.

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

i

IMPORTANT!

If you use Visual Basic for Windows the following parameters have no meaning. You must use the "i_PA3500_TestInterrupt" function.

Calling convention:

ANSI C :

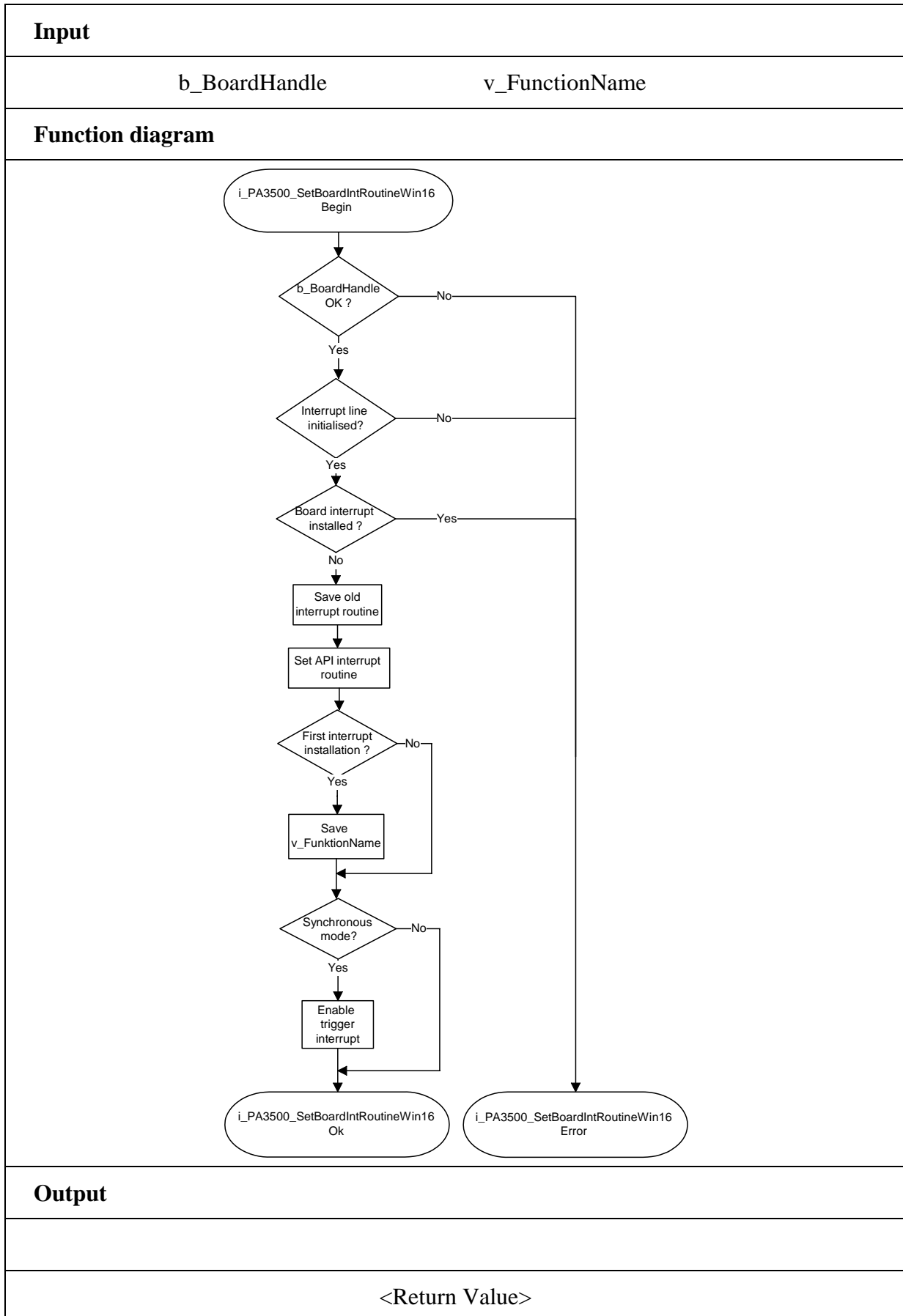
```
void    v_FunctionName    (unsigned char b_BoardHandle,
                          unsigned char b_InterruptMask,
                          unsigned char b_LastChannelNumber)
    {
        .
        .
    }
```

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

```
i_ReturnValue = i_PA3500_SetBoardIntRoutineWin16 (b_BoardHandle,
                                                  v_FunctionName );
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: No interrupt line has been initialised
- 3: Interrupt already installed



i**IMPORTANT!**

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

4) i_PA3500_SetBoardIntRoutineWin32 (..)**Syntax:**

```
<Return value> = i_PA3500_SetBoardIntRoutineWin32
    (BYTE    b_BoardHandle,
     BYTE    b_UserCallingMode,
     ULONG   ul_UserSharedMemorySize,
     VOID ** ppv_UserSharedMemory,
     VOID    v_FunctionName
    (BYTE    b_BoardHandle,
     BYTE    b_InterruptMask,
     BYTE    b_LastChannelNumber,
     BYTE    b_UserCallingMode,
     VOID *  pv_UserSharedMemory))
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of board PA 3500
BYTE	b_UserCallingMode	PA3500_SYNCHRONOUS_MODE: The user routine is directly called by the driver interrupt routine. PA3500_ASYNCHRONOUS_MODE: The user routine is called by the driver interrupt thread.
VOID	v_FunctionName	Name of the user interrupt routine
ULONG	ul_UserSharedMemorySize	Determines the size in bytes of the user shared memory. Only used if you have selected the PA3500_SYNCHRONOUS_MODE mode.

i**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 the PA3500_SYNCHRONOUS_MODE mode.
---------	----------------------	--

Task:**If you use Visual Basic 5.0:**

- Only the asynchronous mode is available.

i**Windows 32-bit information**

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

- The user application operates in ring 3. This ring does not give access to hardware.
- VXD and SYS driver operate in ring 0 and give access to hardware.
- Ring 0 has no direct access to the global variables of 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 3500** 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 `PA3500_SYNCHRONOUS_MODE` has been selected.

If you operate several boards **PA 3500** which have to react to interrupts, call up the function as often as you operate on boards **PA 3500**. The variable *v_FunctionName* is only relevant **for the first calling**.

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

- interrupts are enabled.

Interrupt

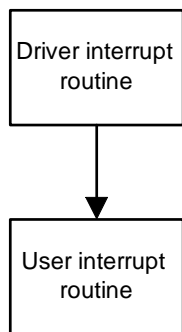
The user interrupt routine is called up by the system when an interrupt is generated.

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.

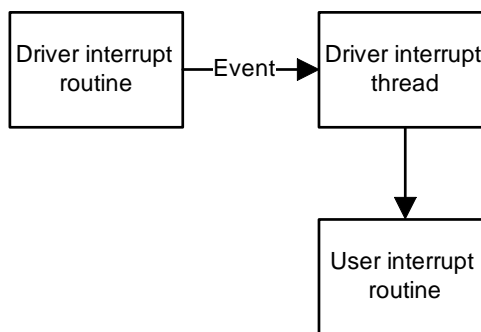
User interrupt routine can be called:

- directly by the driver interrupt routine (synchronous mode). The code of the user interrupt routine directly operates in ring 0.
 - by the driver interrupt thread (asynchronous mode). An event is generated and the interrupt thread calls up the user interrupt routine. The code of the user interrupt routine operates in ring 3.
- The driver interrupt thread has 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 3500 driver functions with the following extension "i_PA3500_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 all PA 3500 driver functions with the following extension: "i_PA3500_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 PA3500_SYNCHRONOUS_MODE you cannot have access to the 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,
                    BYTE      b_LastChannelNumber,
                    BYTE      b_UserCallingMode,
                    VOID *    pv_UserSharedMemory)
```

<i>v_FunctionName</i>	Name of the user interrupt routine
<i>b_BoardHandle</i>	Handle of the PA 3500 which has generated the interrupt
<i>b_InterruptMask</i>	Mask of the events which have generated the interrupt
<i>b_LastChannelNumber</i>	Number of the last channel that has been triggered
<i>b_UserCallingMode</i>	PA3500_SYNCHRONOUS_MODE: The user routine is directly called by the driver interrupt routine. PA3500_ASYNCHRONOUS_MODE: The user routine is called by driver interrupt thread
<i>pv_UserSharedMemory</i>	Pointer of the user shared memory.

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

i

IMPORTANT!

If you use Visual Basic 4 the following parameters have no meaning. You must use the "i_PA3500_TestInterrupt" function.

```
BYTE      b_UserCallingMode,
ULONG     ul_UserSharedMemorySize,
VOID **   ppv_UserSharedMemory,
VOID      v_FunctionName (BYTE      b_BoardHandle,
                        BYTE      b_InterruptMask,
                        BYTE      b_LastChannelNumber,
                        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 char b_LastChannelNumber,
                    unsigned char b_UserCallingMode,
                    void *        pv_UserSharedMemory)
```

```

    {
        str_UserStruct * ps_InterruptSharedMemory;

        ps_InterruptSharedMemory = (str_UserStruct *)
v_UserSharedMemory;
        .
        .
    }

int          i_ReturnValue;
unsigned char b_BoardHandle;

i_ReturnValue = i_PA3500_SetBoardIntRoutineWin32
                (b_BoardHandle,
                PA3500_SYNCHRONOUS_MODE,
                sizeof (str_UserStruct),
                (void **) &ps_UserSharedMemory,
                v_FunctionName);

```

i**IMPORTANT!**

Use the following parameters, when you use Visual Basic 5.0.

Visual Basic 5:

```

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

```

```
End Sub
```

```
Dim i_ReturnValue As Integer
Dim i_BoardHandle As Integer
```

```
...
```

```

i_ReturnValue = i_PA3500_SetBoardIntRoutineWin32
                (i_BoardHandle,
                PA3500_ASYNCHRONOUS_MODE,
                0,
                0,
                AddressOf v_FunctionName)

```

```
...
```

Return value:

0: No error

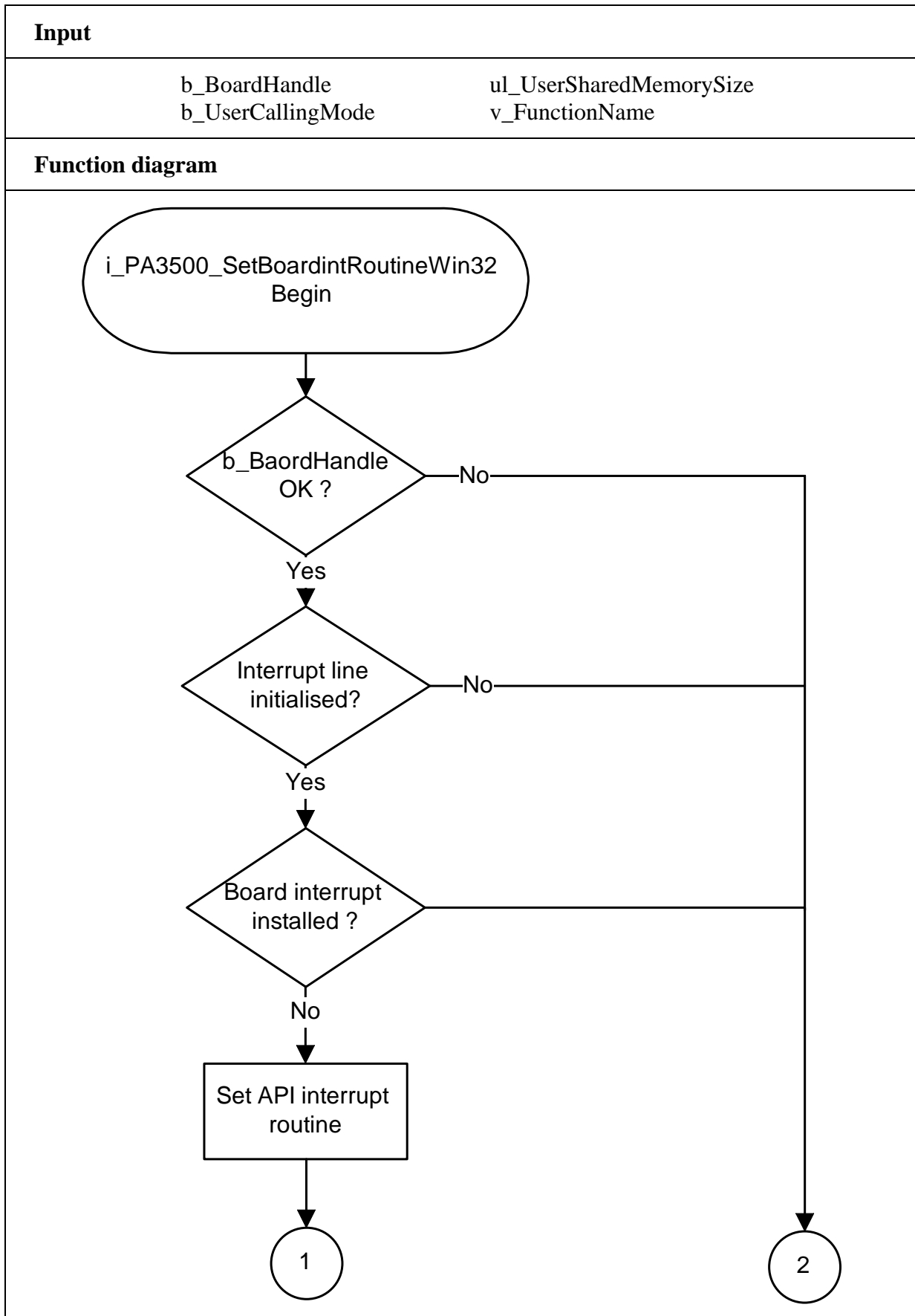
-1: The handle parameter of the board is wrong

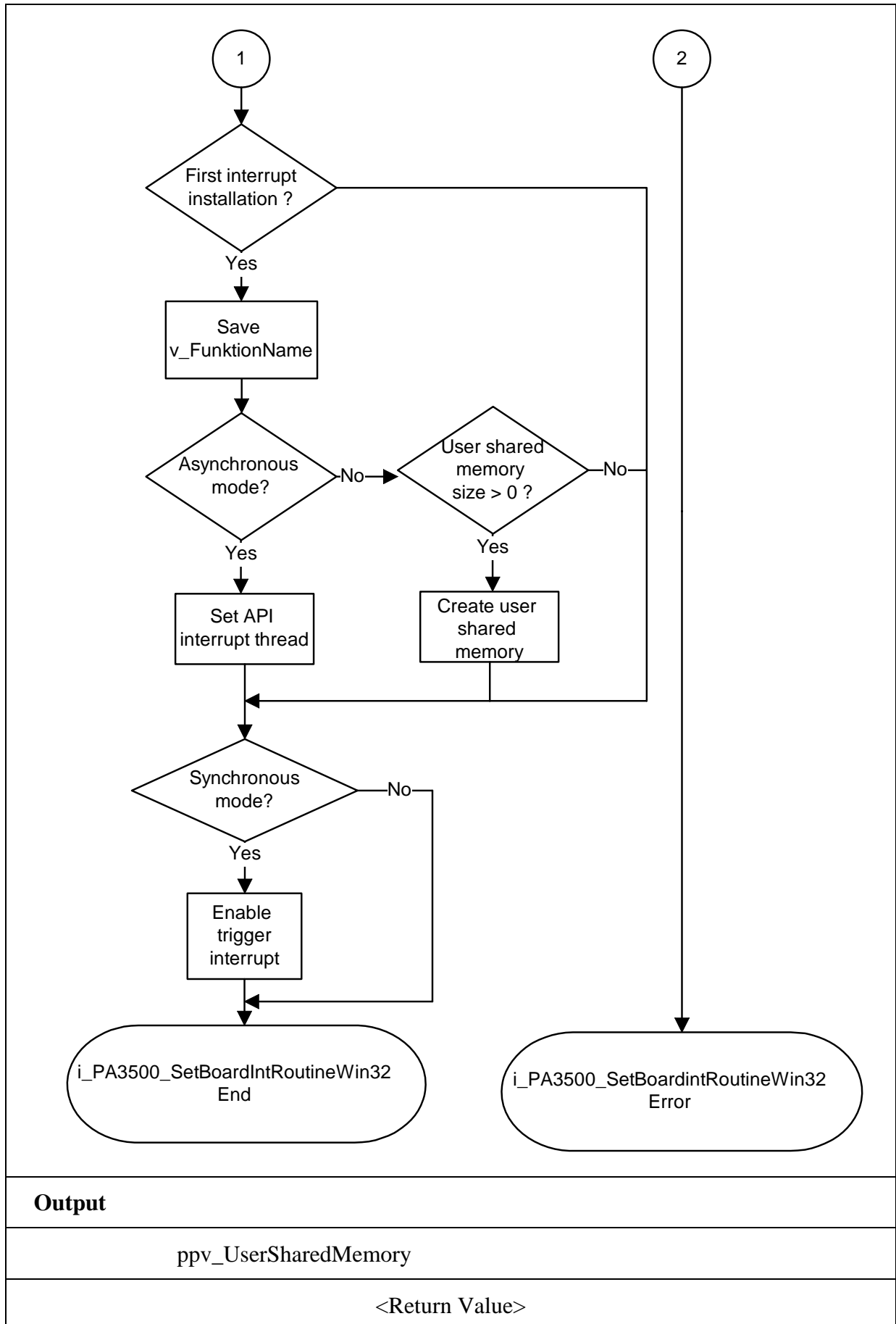
-2: No interrupt line has been initialised

-3: Interrupt already installed

-4: The calling mode selected for the user interrupt routine is wrong

-5: No memory available for the user shared memory





Output

ppv_UserSharedMemory

<Return Value>

5) i_PA3500_TestInterrupt (..)**Syntax:**

```
<Return value> = i_PA3500_TestInterrupt
                    (PBYTE   pb_BoardHandle,
                    PBYTE   pb_InterruptMask,
                    PBYTE   pb_LastChannelNumber)
```

Parameters:**- Input:**

No output signal has occurred.

- Output:

PBYTE	pb_BoardHandle	Handle of the board PA 3500 which has generated the interrupt
PBYTE	pb_InterruptMask	Mask of the events which have generated the interrupt
PBYTE	pb_LastChannelNumber	Number of the last channel that has been triggered

Task:

Verifies if a board **PA 3500** has generated an interrupt. If yes, the function returns the board handle and the interrupt source.

i**IMPORTANT!**

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

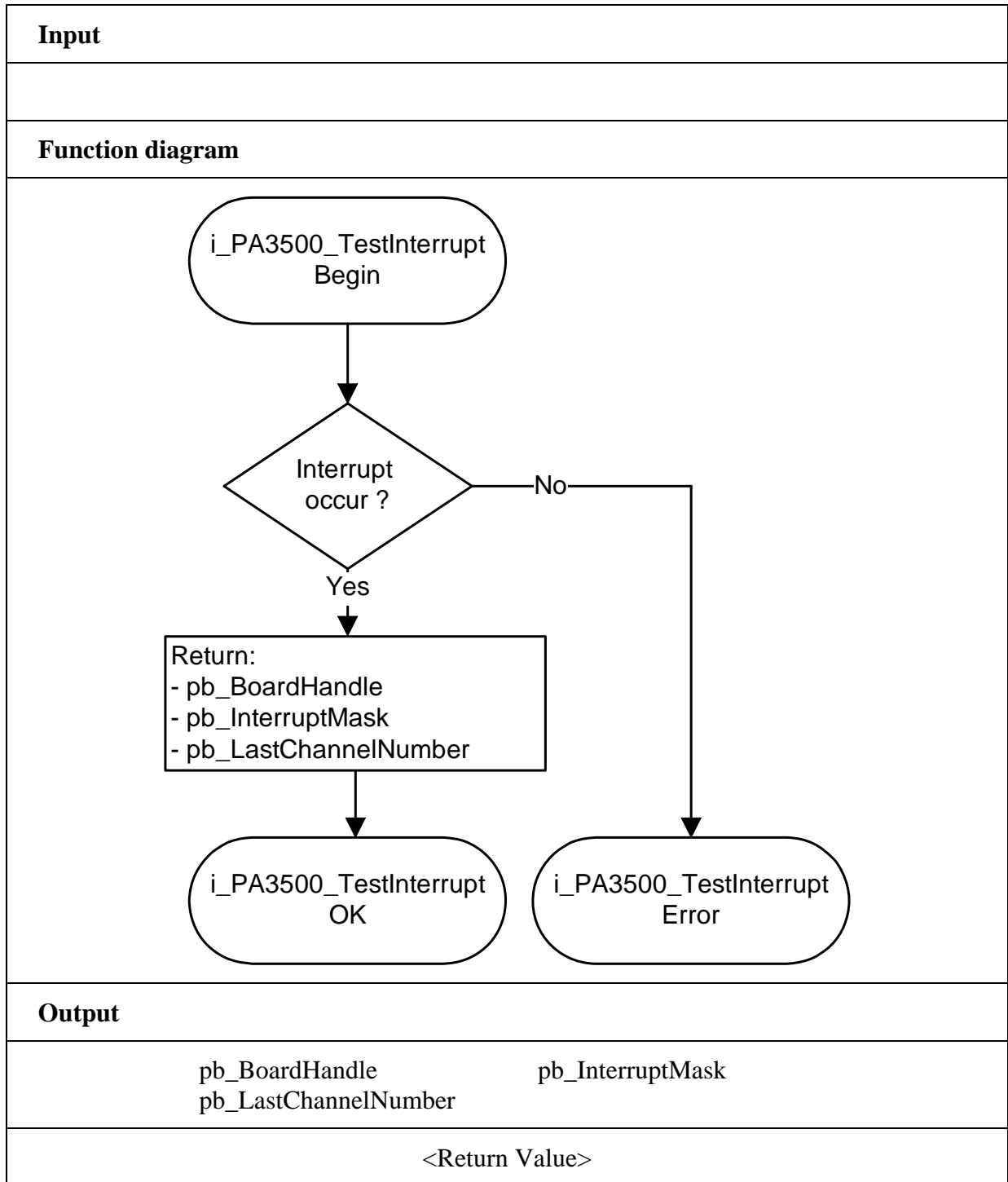
Calling convention:ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle;
unsigned char b_InterruptMask;
unsigned char b_InterruptMask;
unsigned char b_LastChannelNumber;
```

```
i_ReturnValue = i_PA3500_TestInterrupt
                    (&b_BoardHandle,
                    &b_InterruptMask,
                    &b_LastChannelNumber);
```

Return value:

```
-1   No interrupt
> 0  IRQ number
```



6) i_PA3500_ResetBoardIntRoutine (..)**Syntax:**

<Return value> = i_PA3500_ResetBoardIntRoutine (BYTE b_BoardHandle)

Parameters:**- Input:**

BYTE b_BoardHandle Handle of board **PA 3500**

- Output:

No output signal has occurred.

Task:

Stops the interrupt management of board **PA 3500**.

Deinstalls the user interrupt routine if the management of interrupts of all boards **PA 3500** is stopped.

Calling convention:ANSI C:

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

```
i_ReturnValue = i_PA3500_ResetBoardIntRoutine    (b_BoardHandle);
```

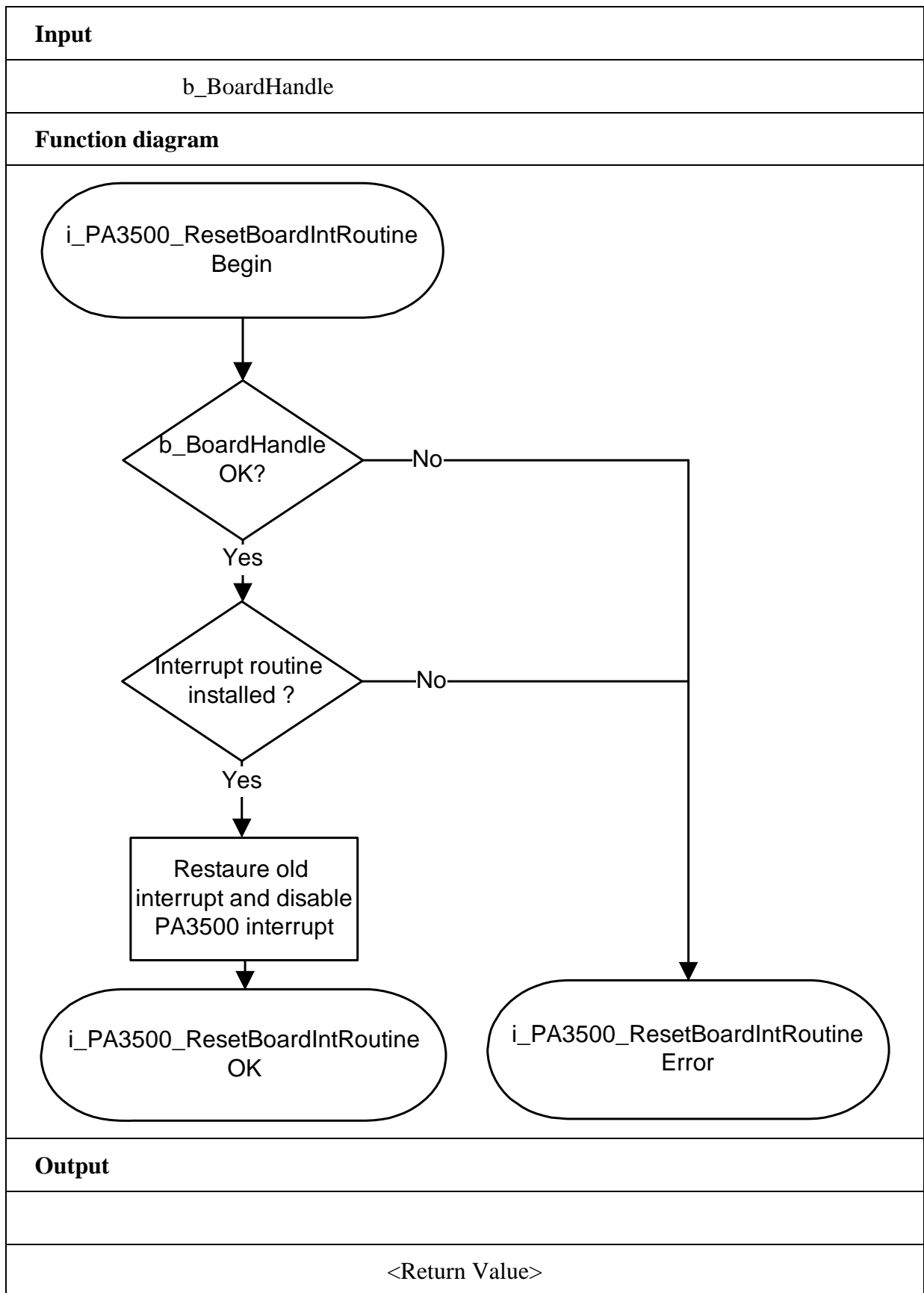
Return value:

0: No error

-1: The handle parameter of the board is wrong

-2: No interrupt function initialised with function

"i_PA3500_SetBoardIntRoutineXXX"



3.3 Analog output channels

1) i_PA3500_Write1AnalogValue (...)

i

IMPORTANT!

This function is only available in PA3500_SIMPLE_MODE.

Syntax:

```
<Return value> = i_PA3500_Write1AnalogValue
                    (BYTE      b_BoardHandle,
                     BYTE      b_ChannelNbr,
                     BYTE      b_Polarity,
                     UINT       ui_ValueToWrite)
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of board PA 3500
BYTE	b_ChannelNbr	Number of the selected analog output (0 to 7).
BYTE	b_Polarity	Polarity selection for the analog output - PA3500_UNIPOLAR: Unipolar (0-10 V) - PA3500_BIPOLAR: Bipolar (± 10 V)
UINT	ui_ValueToWrite	Analog output value to be written . In UNIPOLAR mode: (0 to 8192) In BIPOLAR mode: (0 to 16383) See table 3-4.

- Output:

No output signal has occurred.

Task:

Writes one analog value directly on the analog output.

If:

- you are in SIMPLE mode,
 - the reset flag of the watchdog is enabled,
 - the watchdog is activated and in alarm status,
- the function returns an error code and does not write the value on the output.

The functions "i_PA3500_DisableWatchdog" and "i_PA3500_EnableWatchdog" are to be used to deactivate the alarm status of the watchdog.

Table 3-4: Output voltage

Value	Polarity selection	Output voltage
0	PA3500_UNIPOLAR	0 V
	PA3500_BIPOLAR	-10 V
8192	PA3500_UNIPOLAR	10 V
	PA3500_BIPOLAR	0 V
16383	PA3500_BIPOLAR	10 V

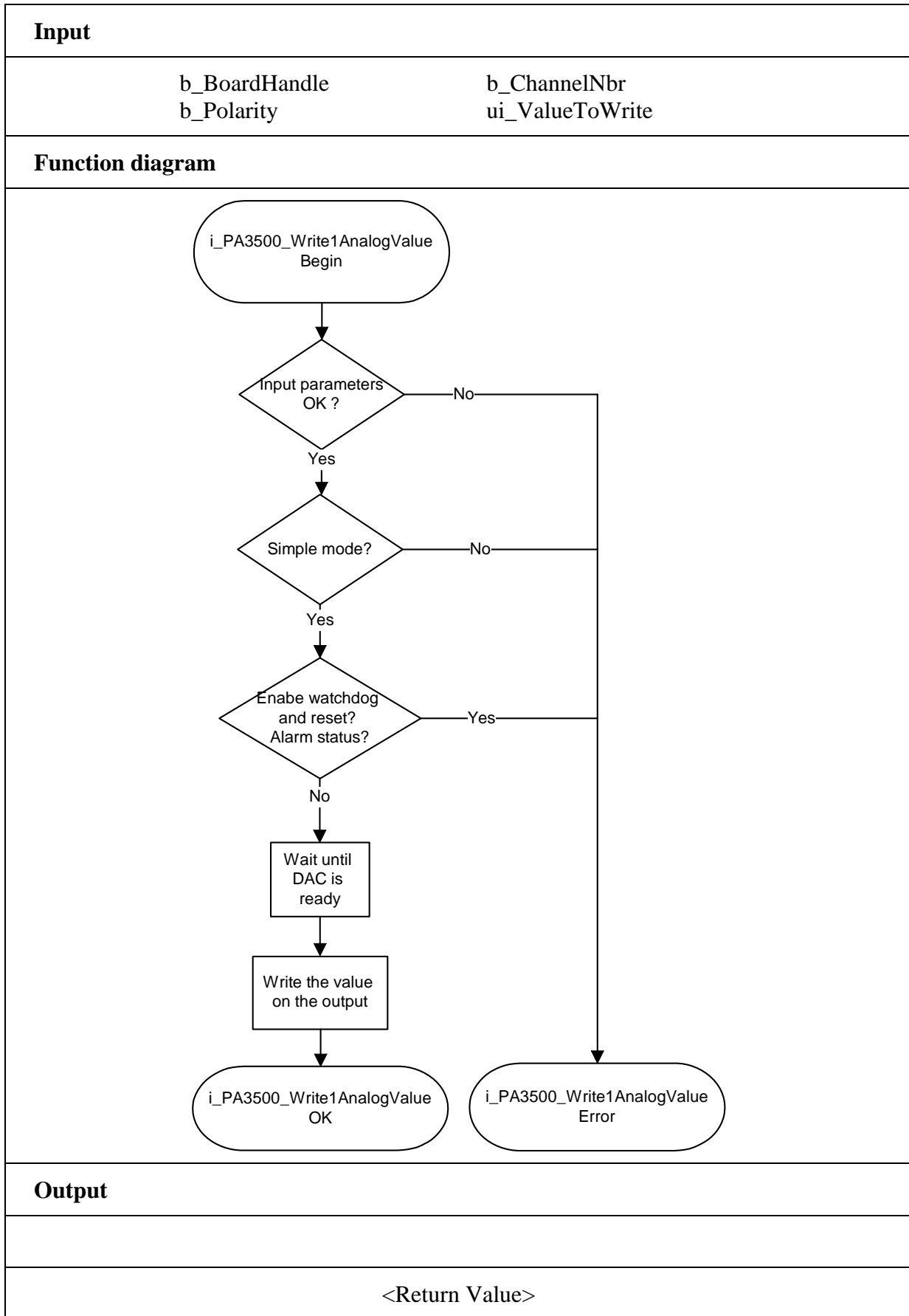
Calling convention:ANSI C:

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

```
i_ReturnValue = i_PA3500_Write1AnalogValue (b_BoardHandle,
                                             PA3500_CHANNEL0,
                                             PA3500_UNIPOLAR,
                                             4095);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The number of the analog output is wrong.
- 3: The polarity selected is wrong.
- 4: The output value is wrong.
- 5: Operating mode error. See Function "i_PA3500_SetBoardInformationXX" or "i_PA3500_ChangeBoardOperatingMode".
- 6: The watchdog is in alarm status. Please use the function "i_PA3500_TriggerWatchdog".
- 7: Time has run down



2) i_PA3500_WriteMoreAnalogValue (...)

i

IMPORTANT!

This function is only available in PA3500_SIMPLE_MODE.

Syntax:

```
<Return value> = i_PA3500_WriteMoreAnalogValue
                    (BYTE    b_BoardHandle,
                     BYTE    b_FirstChannelNbr,
                     BYTE    b_NbrOfChannel,
                     PBYTE   pb_PolarityArray,
                     PUINT   pui_WriteValueArray)
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of board PA 3500
BYTE	b_FirstChannelNbr	Number of the first analog output (0 to 7)
BYTE	b_NbrOfChannel	Number of analog outputs you wish to write on
PBYTE	pb_PolarityArray	Polarity array
PUINT	pui_ValueToWrite	Table of values to be written on the analog outputs

- Output:

No output signal has occurred.

Task:

Writes the values on several analog outputs.

The variable *b_FirstChannelNbr* passes the number of the first analog output.

The variable *b_NbrOfChannel* passes the number of analog outputs.

If the watchdog is activated and in alarm status, the function returns an error code and does not write the value on the output.

The function "i_PA3500_TriggerWatchdog" is to be used to deactivate the alarm status from the watchdog.

Example:

```
b_FirstChannelNbr = 2, b_NbrOfChannel = 3
pui_ValueArray [0] = 0
pui_ValueArray [1] = 2048
pui_ValueArray [2] = 4095
pb_PolarityArray [0] = PA3500_UNIPOLAR
pb_PolarityArray [1] = PA3500_BIPOLAR
pb_PolarityArray [2] = PA3500_UNIPOLAR
```

The value 0 (0 V) is written on the analog output 2.

The value 2048 (-5 V) is written on the analog output 3.

The value 4095 (5 V) is written on the analog output 4.

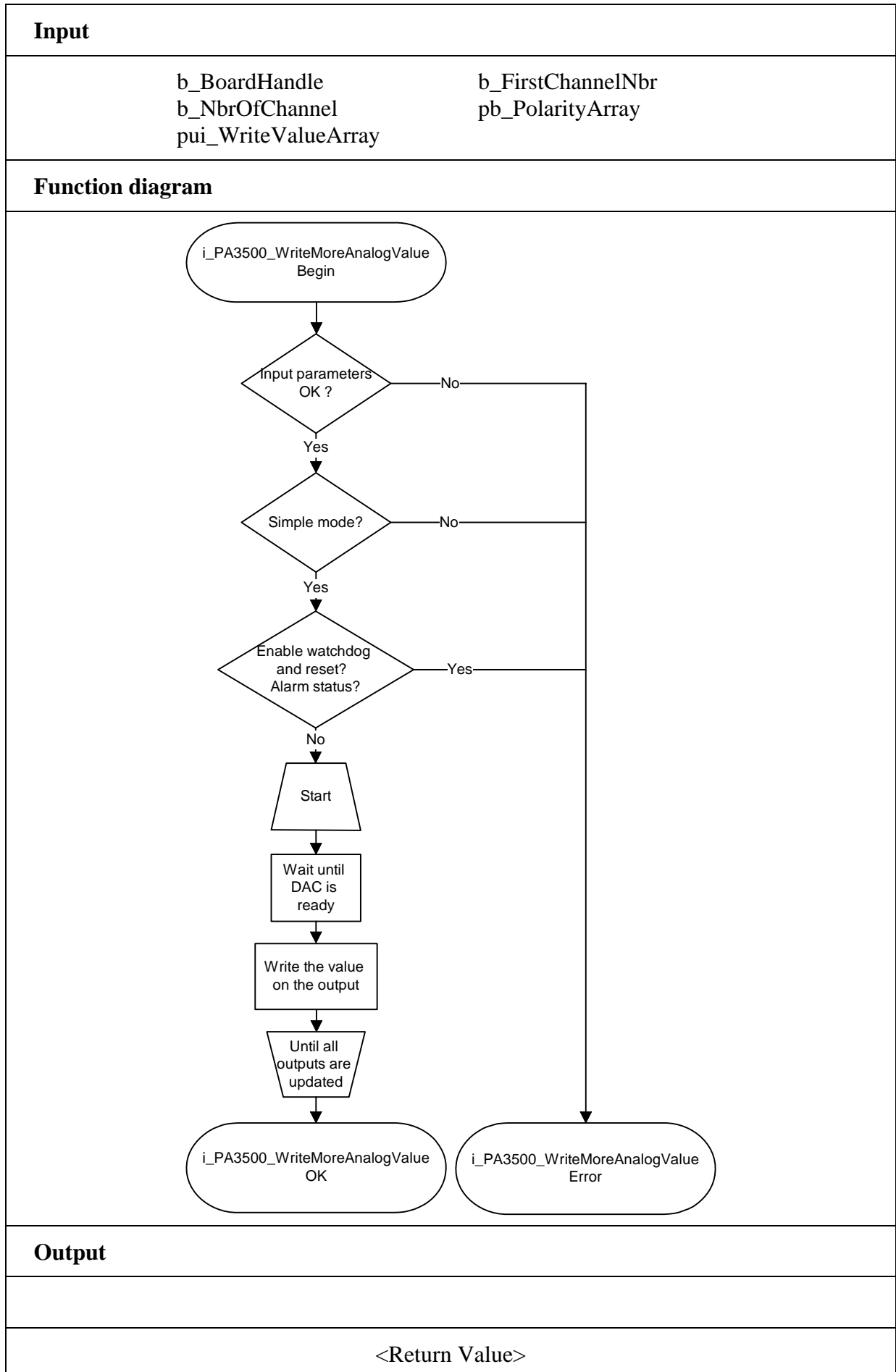
Calling convention:ANSI C :

```
int          i_ReturnValue;
unsigned char b_BoardHandle, b_PolarityArray[4];
unsigned int  ui_WriteValueArray[4];
```

```
i_ReturnValue = i_PA3500_WriteMoreAnalogValue (b_BoardHandle,
                                                PA3500_CHANNEL0,
                                                4,
                                                b_PolarityArray,
                                                ui_WriteValueArray);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The number of the first analog output is wrong.
See function "i_PA3500_SetBoardInformationXXX"
- 3: The number of analog outputs you wish to write on is wrong
See function "i_PA3500_SetBoardInformationXXX"
- 4: One or several polarity selections are wrong
- 5: One or several output values are too high
- 6: Operating mode error. See function
"i_PA3500_SetBoardInformationXXX" or
"i_PA3500_ChangeBoardOperatingMode"
- 7: The watchdog is in alarm status. Please use the function
"i_PA3500_TriggerWatchdog".
- 8: Time has run down



3) i_PA3500_StoreAnalogOutputValue (...)

i

IMPORTANT!

This function is only available in PA3500_TRIGGER_MODE or PA3500_SYNCHRONISATION_MODE.

Syntax:

```
<Return value> = i_PA3500_StoreAnalogOutputValue
                    (BYTE    b_BoardHandle,
                    BYTE    b_EraseLastStorage,
                    BYTE    b_ChannelNbr,
                    BYTE    b_Polarity,
                    UINT    ui_ValueToWrite)
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of board PA 3500
BYTE	b_EraseLastStorage	Select the function progress (see table 3-5). - PA3500_ENABLE: Overwrites the last storage and quits the function with a warning code. - PA3500_DISABLE: Quits the function with an error code.
BYTE	b_ChannelNbr	Number of the selected analog output channel (0 to 7)
BYTE	b_Polarity	Polarity selection of the analog output channel. - PA3500_UNIPOLAR: unipolar (0-10 V) - PA3500_BIPOLAR: bipolar (± 10 V)
UINT	ui_ValueToWrite	Analog output value to be written In unipolar mode: (0 to 8192) In bipolar mode: (0 to 16383).

- Output:

No output signal has occurred.

Task:

Stores the analog value and the polarity selection of an output in the output buffer. The output value buffer writes on the output when an external trigger occurs or when the external trigger is simulated with the function "i_PA3500_SimulateExternalTrigger".

Calling convention:

ANSI C :

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

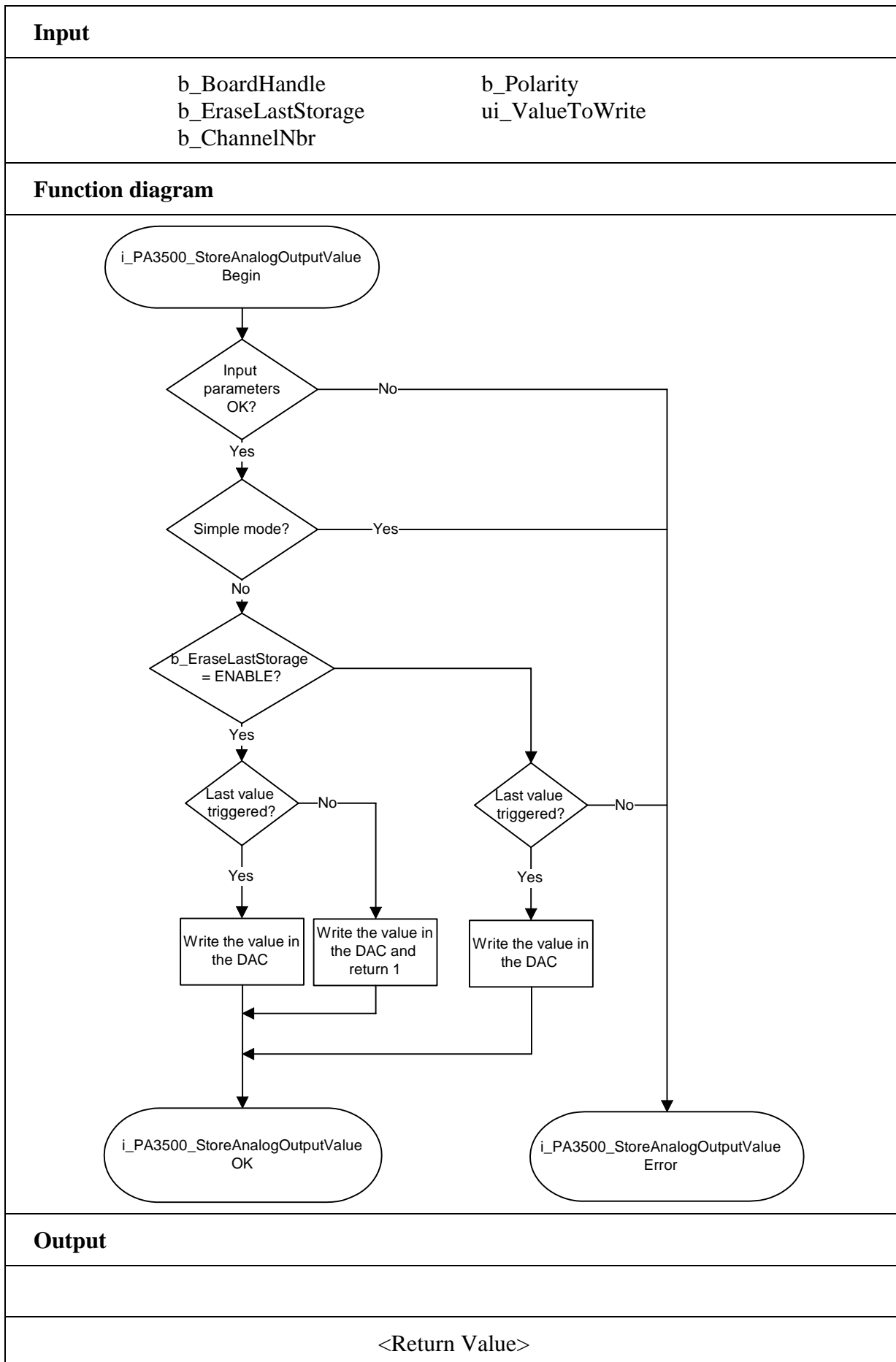
```
i_ReturnValue = i_PA3500_StoreAnalogOutputValue (b_BoardHandle,
                                                  PA3500_DISABLE,
                                                  PA3500_CHANNEL0,
                                                  PA3500_UNIPOLAR,
                                                  4095);
```

Table 3-5: Function progress

Last stored value triggered?	<i>b_EraseLastStorage</i> selection	Function course
Yes	PA3500_ENABLE	Normal progress
	PA3500_DISABLE	Normal progress
No	PA3500_ENABLE	Overwrites the last: - stored value - polarity selection - channel selection. Quits the function with warning code
	PA3500_DISABLE	Quits the function with error code

Return value:

- 1: Warning: last stored information is overwritten
- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The number of the analog output channel is wrong
- 3: The polarity selection of the analog output channel is wrong
- 4: The analog output value is wrong
- 5: The function course selection is wrong
- 6: The last stored information is not triggered
- 7: Board operating mode error. See function "i_PA3500_SetBoardInformationXXX" or "i_PA3500_ChangeBoardOperatingMode"
- 8: You have selected the PA3500_SYNCHRONISATION_MODE and not initialised the interrupt routine. See "i_PA3500_SetBoardIntRoutineXXX"



4) i_PA3500_GetExternTriggerStatus (...)**i****IMPORTANT!**

This function is only available in **PA3500_TRIGGER_MODE** or **PA3500_SYNCHRONISATION_MODE**.

Syntax:

```
<Return value> = i_PA3500_GetExternTriggerStatus
                    (BYTE    b_BoardHandle,
                     PBYTE   pb_ExternTriggerStatus)
```

Parameters:**- Input:**

BYTE b_BoardHandle Handle of board **PA 3500**

- Output:

PBYTE pb_ExternTriggerStatus Returns the trigger status.
 0: The last stored value is not triggered
 1: The last stored value is triggered

Task:

Tests whether the last stored information has been triggered or not.

Calling convention:

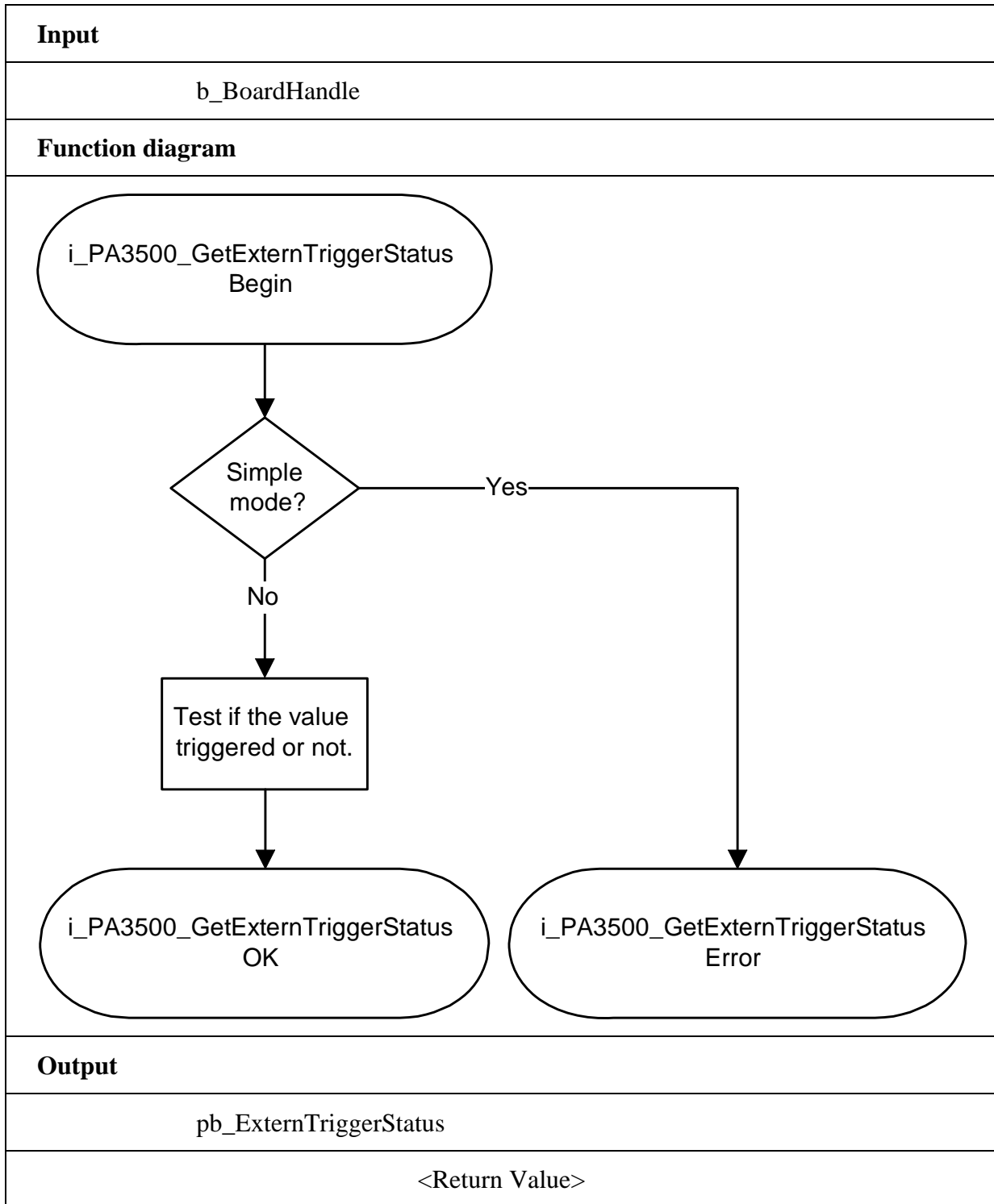
ANSI C:

```
int          i_ReturnValue;
unsigned char b_BoardHandle, b_ExternTriggerStatus;
```

```
i_ReturnValue = i_PA3500_GetExternTriggerStatus (b_BoardHandle,
                                                &b_ExternTriggerStatus);
```

Return value:

0: No error
 -1: The handle parameter of the board is wrong
 -2: Board operating mode error. See function
 "i_PA3500_SetBoardInformationXXX" or
 "i_PA3500_ChangeBoardOperatingMode"



5) i_PA3500_SimulateExternalTrigger (...)**i****IMPORTANT!**

This function is only available in PA3500_TRIGGER_MODE or PA3500_SYNCHRONISATION_MODE.

Syntax:

```
<Return value> = i_PA3500_SimulateExternalTrigger
                    (BYTE    b_BoardHandle)
```

Parameters:**- Input:**

BYTE b_BoardHandle Handle of board PA 3500

- Output:

No output signal has occurred.

Task:

Simulates the external trigger. This operation writes the last stored information on the analog output.

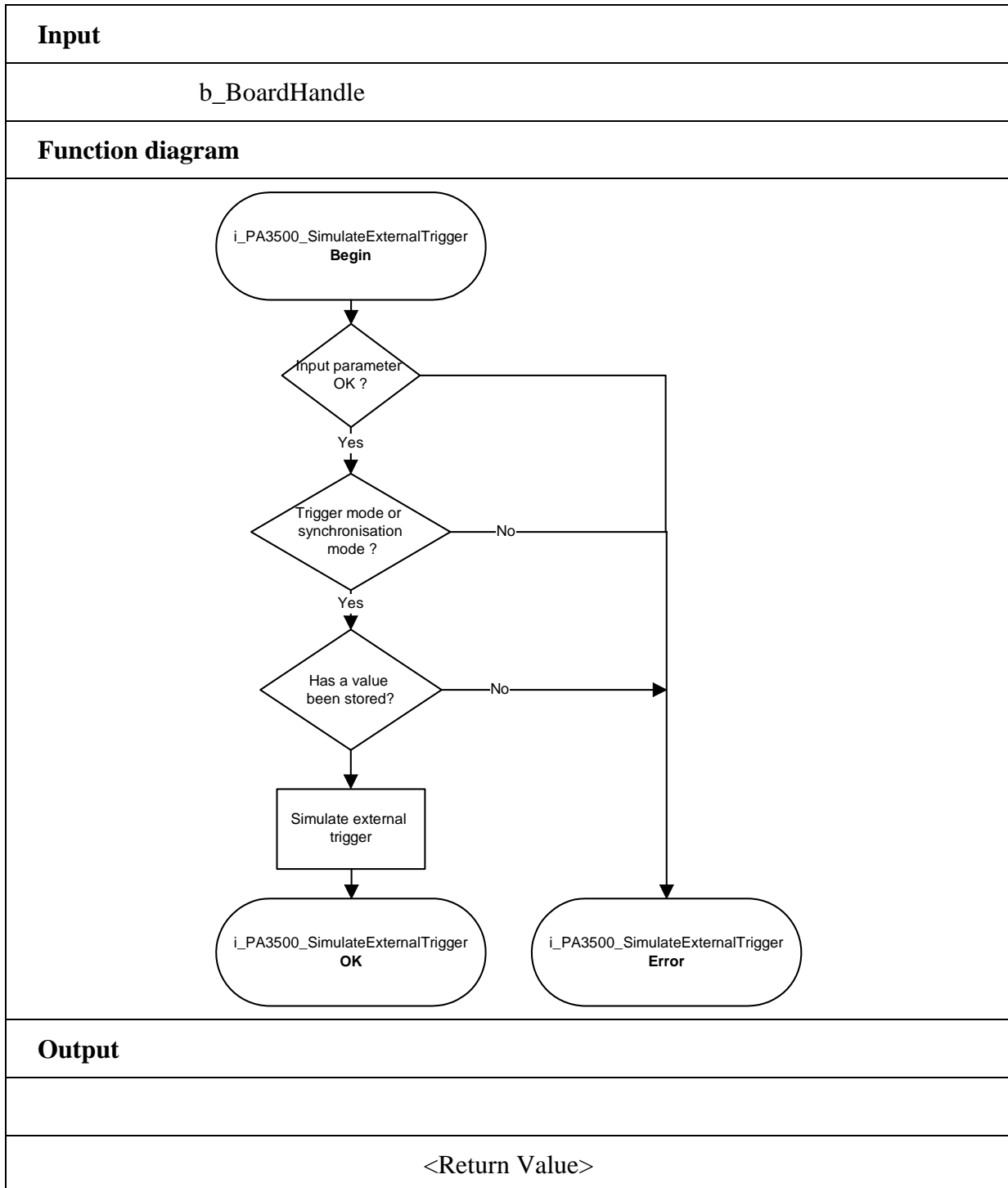
Calling convention:ANSI C:

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

```
i_ReturnValue = i_PA3500_SimulateExternalTrigger (b_BoardHandle);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: No information stored. See function "i_PA3500_StoreAnalogOutputValue"
- 3: Board operating mode error. See function "i_PA3500_SetBoardInformationXXX" or "i_PA3500_ChangeBoardOperatingMode"



6) i_PA3500_EnableTriggerInterrupt (...)**i****IMPORTANT!**

This function is only available in PA3500_SIMPLE_MODE or PA3500_TRIGGER_MODE.

Syntax:

```
<Return value> = i_PA3500_EnableTriggerInterrupt
                    (BYTE    b_BoardHandle)
```

Parameters:**- Input:**

BYTE b_BoardHandle Handle of board PA 3500

- Output:

No output signal has occurred.

Task:

Enables the trigger interrupt.

Calling convention:

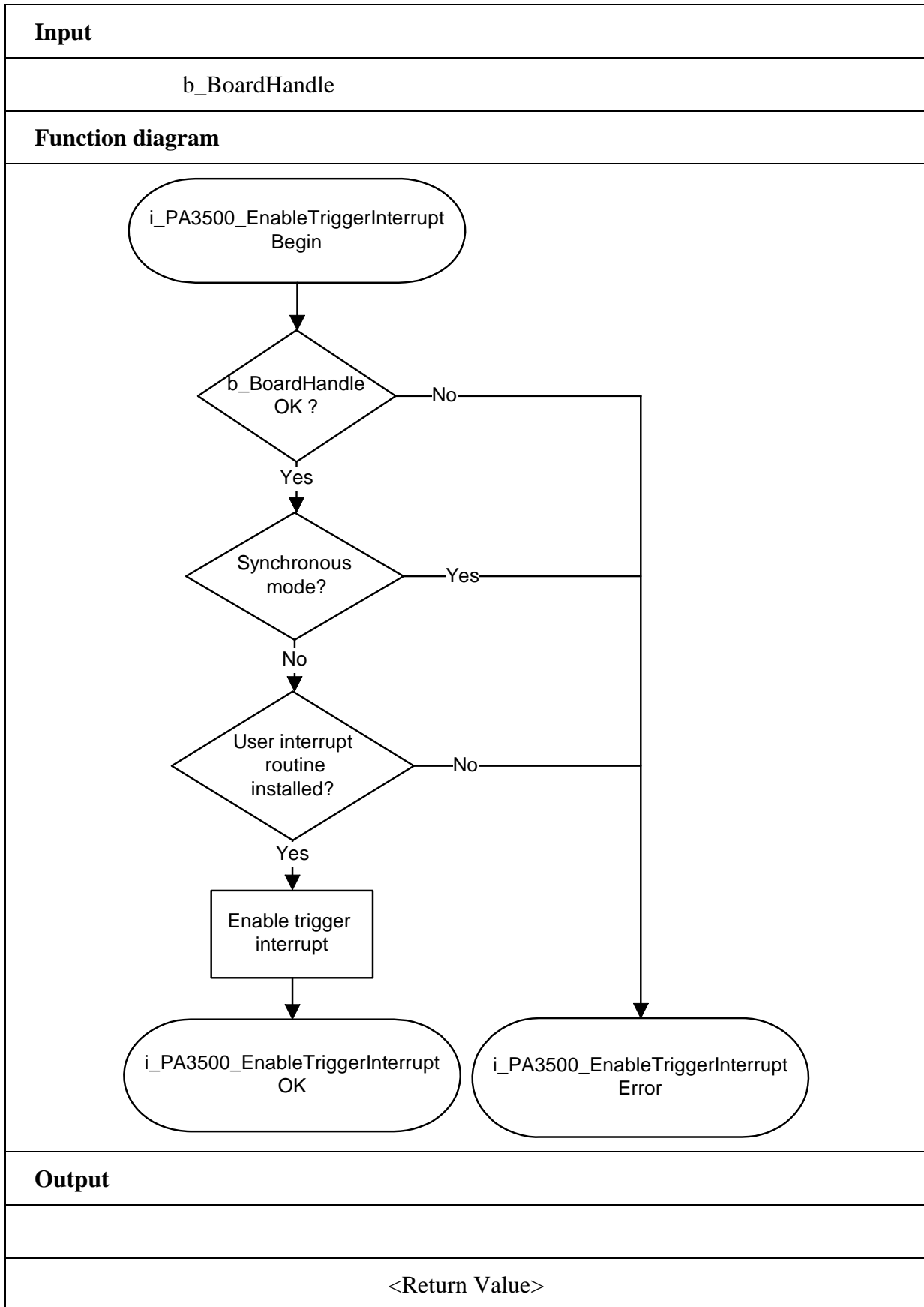
ANSI C:

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

```
i_ReturnValue = i_PA3500_EnableTriggerInterrupt (b_BoardHandle);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The user interrupt routine has not been installed .
See "i_PA3500_SetBoardInformationXX".
- 3: Board operating mode error. See function
"i_PA3500_SetBoardInformationXXX" or
"i_PA3500_ChangeBoardOperatingMode"



7) i_PA3500_DisableTriggerInterrupt (...)**i****IMPORTANT!**

This function is only available in PA3500_SIMPLE_MODE or PA3500_TRIGGER_MODE.

Syntax:

```
<Return value> = i_PA3500_DisableTriggerInterrupt
                    (BYTE    b_BoardHandle)
```

Parameters:**- Input:**

BYTE b_BoardHandle Handle of board PA 3500

- Output:

No output signal has occurred.

Task:

Disables the trigger interrupt.

Calling convention:

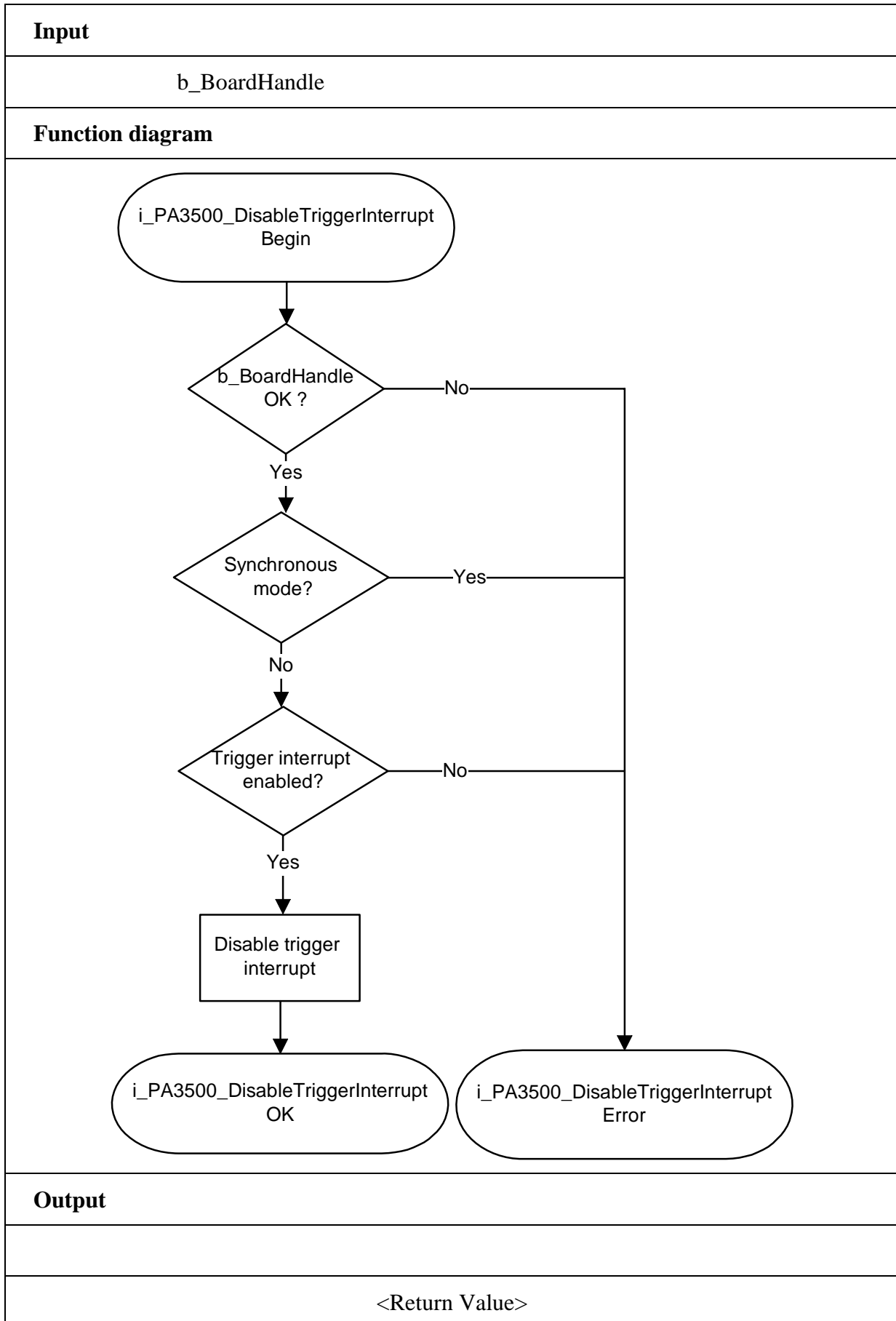
ANSI C:

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

```
i_ReturnValue = i_PA3500_DisableTriggerInterrupt (b_BoardHandle);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The trigger interrupt has not been enabled
See "i_PA3500_EnableTriggerInterrupt."
- 3: Board operating mode error. See function
"i_PA3500_SetBoardInformationXXX" or
"i_PA3500_ChangeBoardOperatingMode"



3.4 Watchdog

1) i_PA3500_EnableWatchdog (...)

Syntax:

```
<Return value> = i_PA3500_EnableWatchdog
                    (BYTE    b_BoardHandle,
                    BYTE    b_InterruptFlag,
                    BYTE    b_ResetFlag)
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of board PA 3500
BYTE	b_InterruptFlag	Enables or disables watchdog interrupt. PA3500_ENABLE: When the watchdog runs down, an interrupt is generated. PA3500_DISABLE: No interrupt.
BYTE	b_ResetFlag	PA3500_ENABLE: The analog and digital outputs are reset when the alarm of the watchdog is set (available in SIMPLE mode) PA3500_DISABLE: The outputs are not reset when the alarm of the watchdog is set.

- Output:

No output signal has occurred.

Task:

Enables the digital and analog watchdog. The watchdog time is 4.69 s. Every function writing on an analog output triggers the watchdog.

Calling convention:

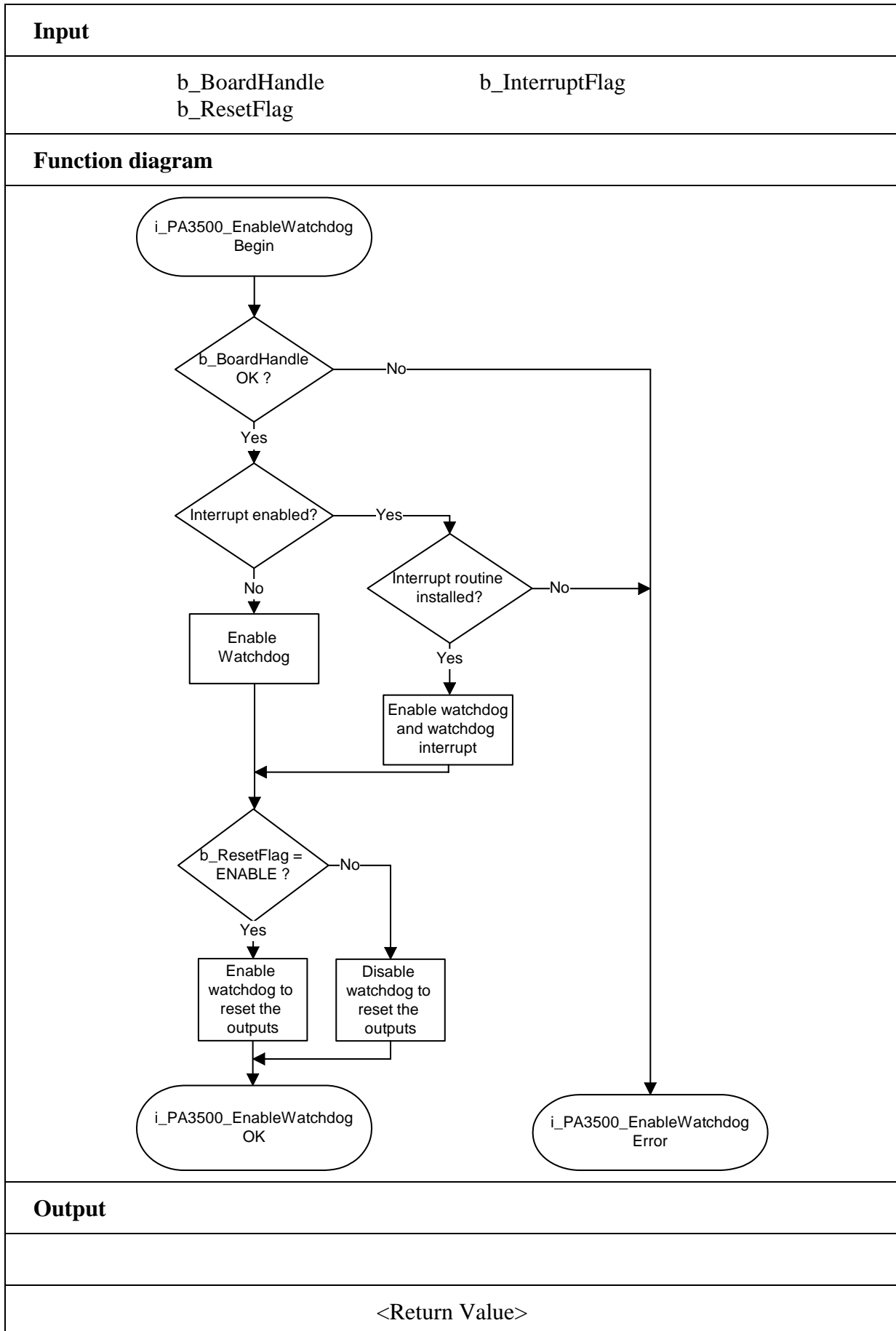
ANSI C:

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

```
i_ReturnValue = i_PA3500_EnableWatchdog (b_BoardHandle,
                                         PA3500_DISABLE,
                                         PA3500_ENABLE );
```

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_PA3500_SetBoardIntRoutineXX"
- 3: The reset flag parameter is wrong.
- 4: The interrupt flag parameter is wrong.
- 5: You cannot enable the reset flag in synchronisation mode.



2) i_PA3500_GetWatchdogStatus (...)

Syntax:

```
<Return value> = i_PA3500_GetWatchdogStatus  
                    (BYTE    b_BoardHandle,  
                    PBYTE   pb_WatchdogStatus)
```

Parameters:

- Input:

BYTE b_BoardHandle Handle of board **PA 3500**

- Output:

PBYTE pb_WatchdogStatus Returns the watchdog status.
0: Watchdog is active
1: Watchdog has run down

Task:

Tests if the watchdog has run down.

Calling convention:

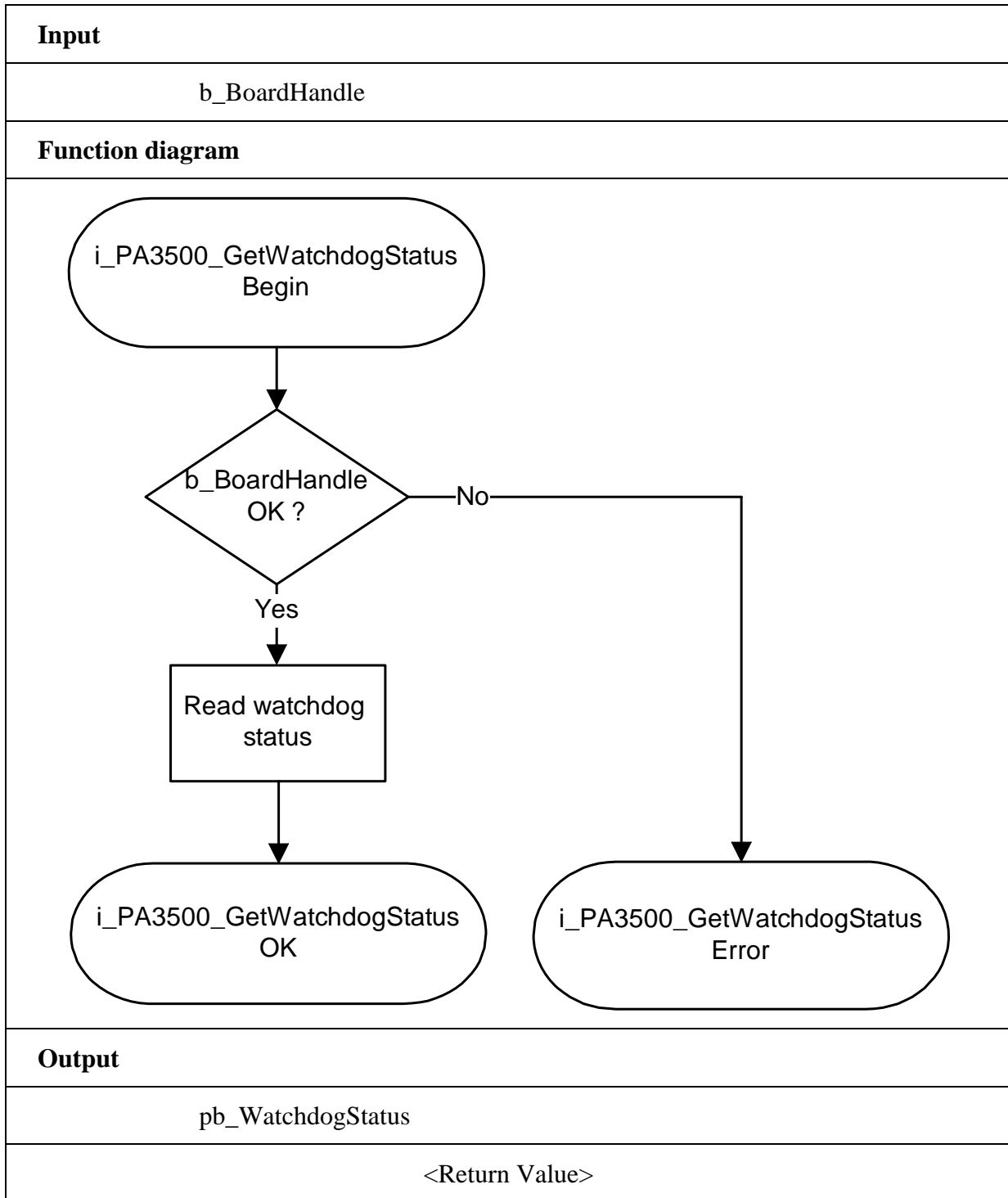
ANSI C:

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

```
i_ReturnValue = i_PA3500_GetWatchdogStatus (b_BoardHandle,  
                                             &b_WatchdogStatus);
```

Return value:

0: No error
-1: The handle parameter of the board is wrong
-2: The watchdog is not enabled. See function "i_PA3500_EnableWatchdog".



3) i_PA3500_TriggerWatchdog (...)

Syntax:

```
<Return value> = i_PA3500_TriggerWatchdog  
                  (BYTE    b_BoardHandle)
```

Parameters:**- Input:**

BYTE b_BoardHandle Handle of board **PA 3500**

- Output:

No output signal has occurred.

Task:

Triggers the watchdog.

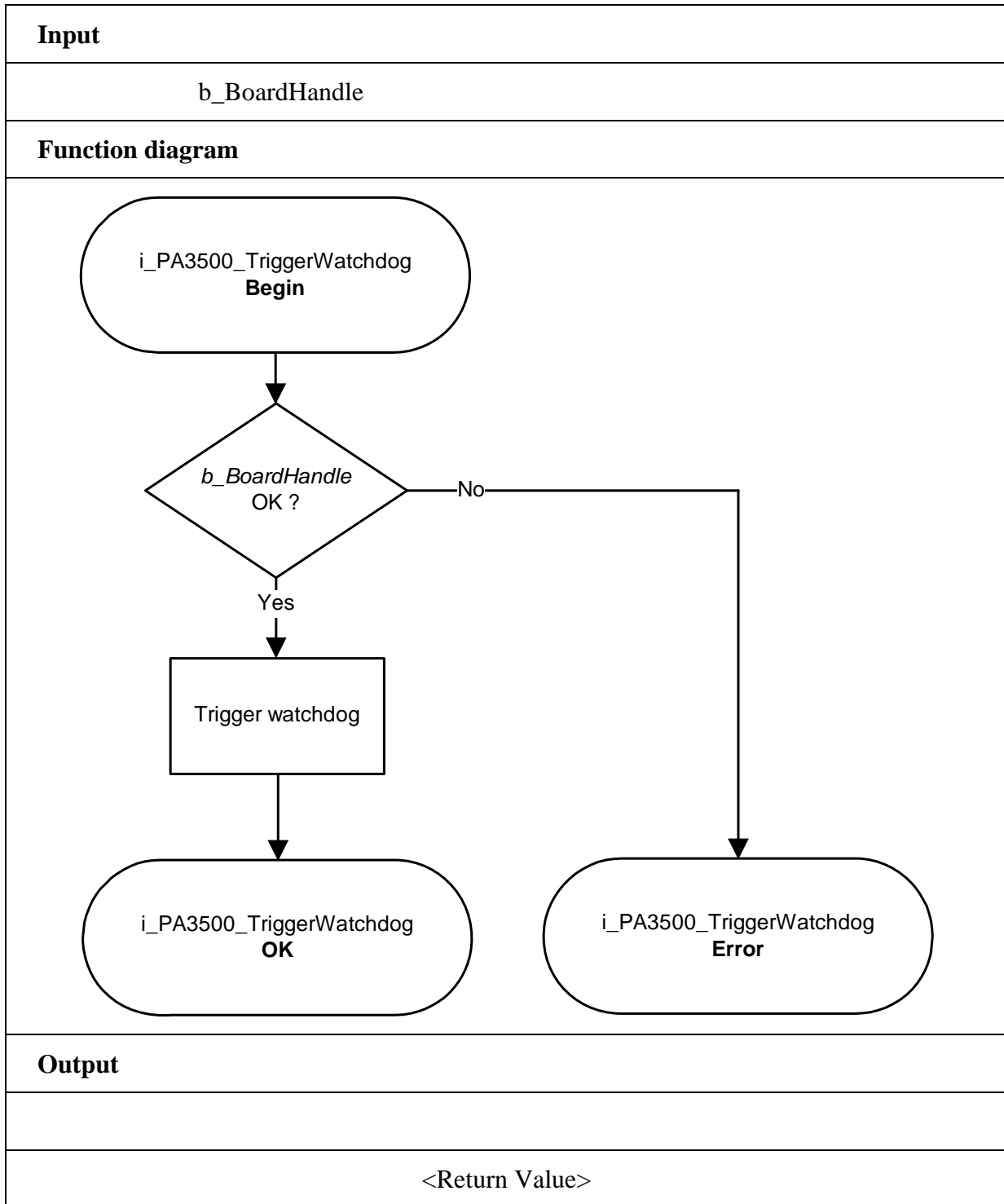
Calling convention:ANSI C:

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

```
i_ReturnValue = i_PA3500_TriggerWatchdog (b_BoardHandle);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The watchdog is not enabled. See function "i_PA3500_EnableWatchdog".



4) i_PA3500_DisableWatchdog (...)

Syntax:

```
<Return value> = i_PA3500_DisableWatchdog  
                  (BYTE    b_BoardHandle)
```

Parameters:**- Input:**

BYTE b_BoardHandle Handle of board **PA 3500**

- Output:

No output signal has occurred.

Task:

Disables the digital and analog watchdog.

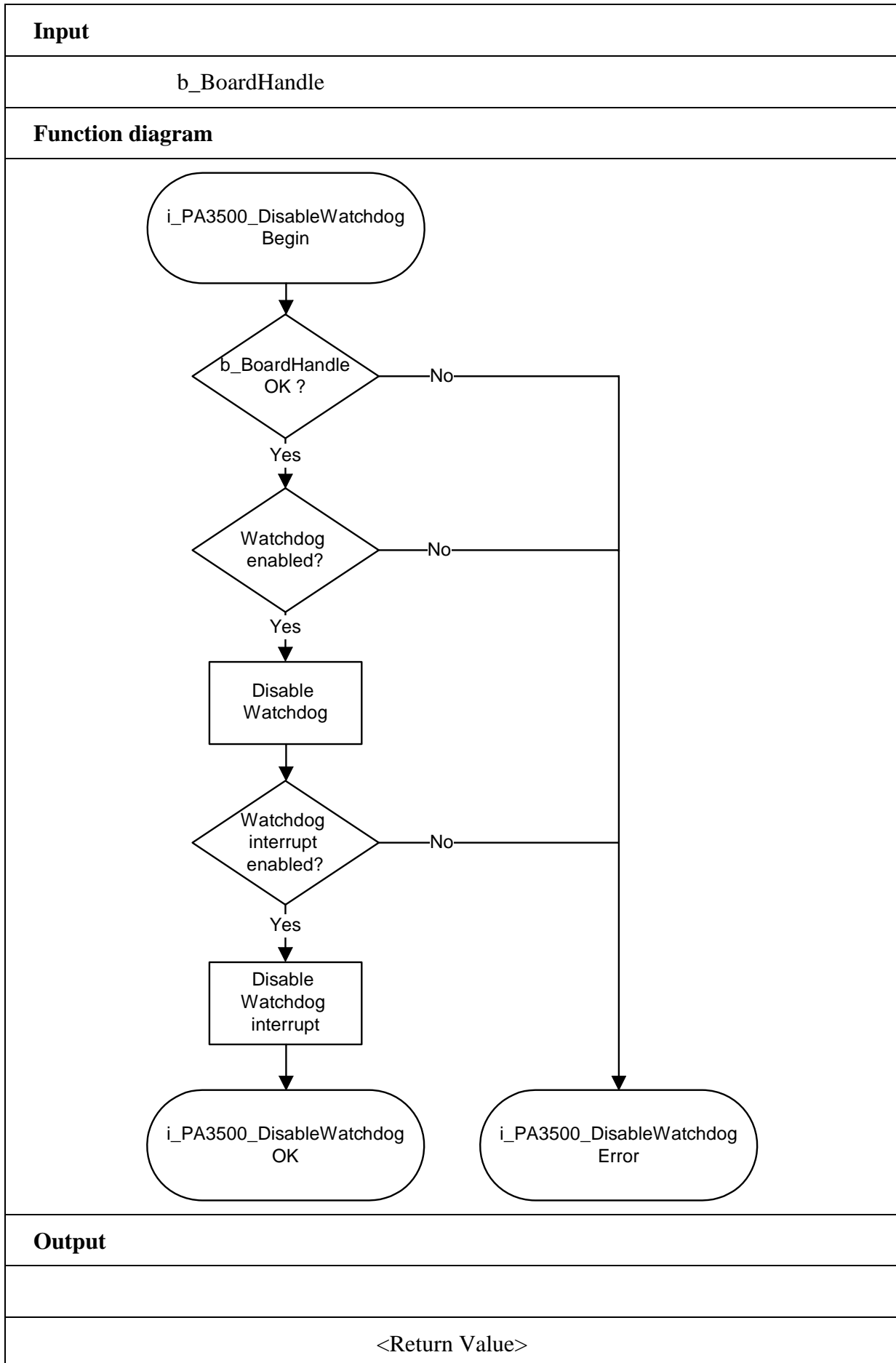
Calling convention:ANSI C:

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

```
i_ReturnValue = i_PA3500_DisableWatchdog (b_BoardHandle);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The watchdog is not enabled. See function "i_PA3500_EnableWatchdog".



3.5 Digital output channels

i

IMPORTANT!

The following functions are not available in the mode PA3500_SYNCHRONISATION_MODE.

1) i_PA3500_SetOutputMemoryOn (...)

Syntax:

```
<Return value> = i_PA3500_SetOutputMemoryOn
                    (BYTE    b_BoardHandle)
```

Parameters:

- Input:

BYTE b_BoardHandle Handle of board PA 3500

- 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_PA3500_SetXDigitalOutputOn" are not reset. You can reset them with the function "i_PA3500_SetXDigitalOutputOff".

Calling convention:

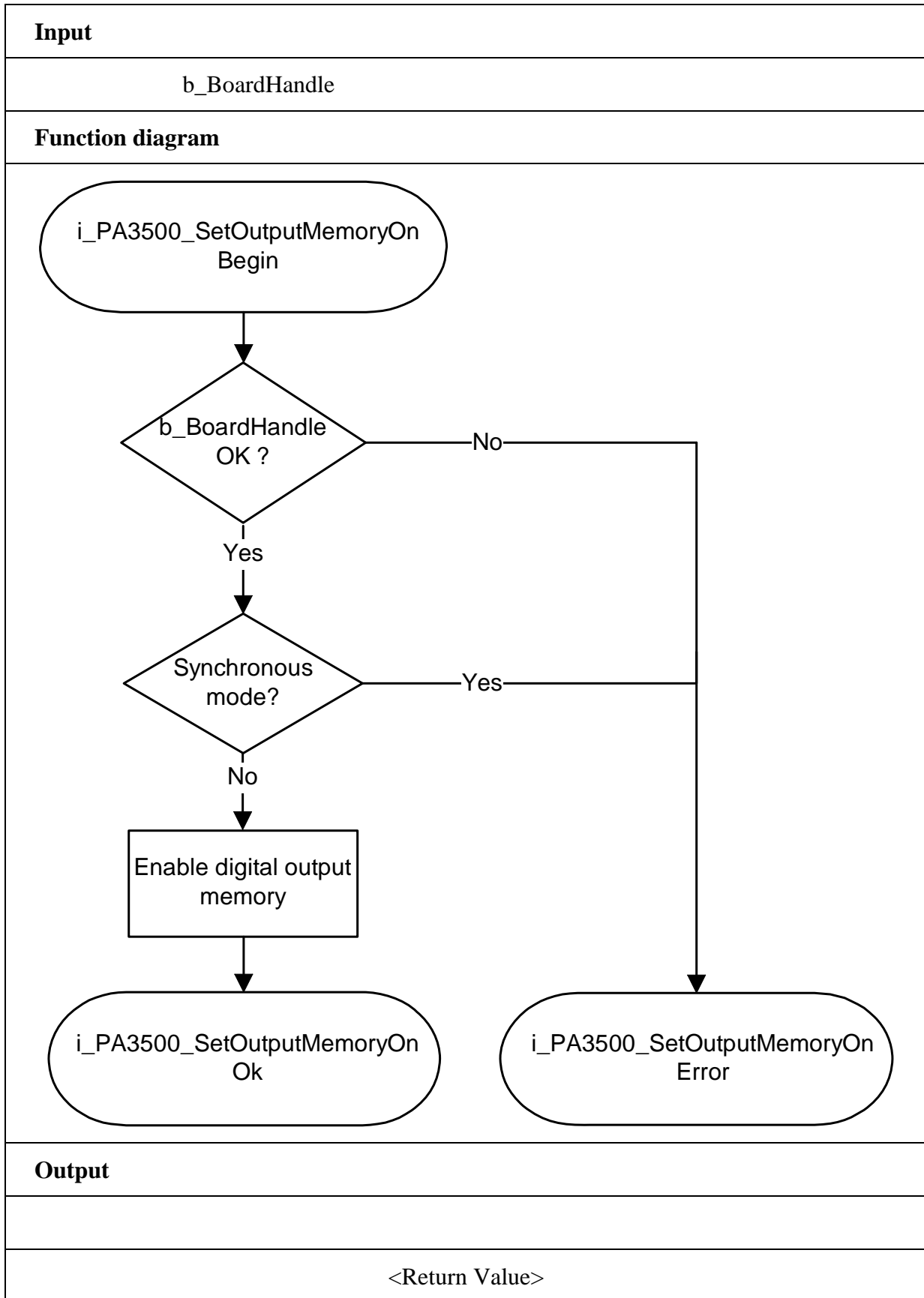
ANSI C:

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

```
i_ReturnValue = i_PA3500_SetOutputMemoryOn (b_BoardHandle);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: Board operating mode error. See function "i_PA3500_SetBoardInformation" or "i_PA3500_ChangeBoardOperatingMode".



2) i_PA3500_SetOutputMemoryOff (...)

Syntax:

<Return value> = i_PA3500_SetOutputMemoryOff
(BYTE b_BoardHandle)

Parameters:

- Input:

BYTE b_BoardHandle Handle of board **PA 3500**

- Output:

No output signal has occurred.

Task:

Deactivates the digital output memory.

Calling convention:

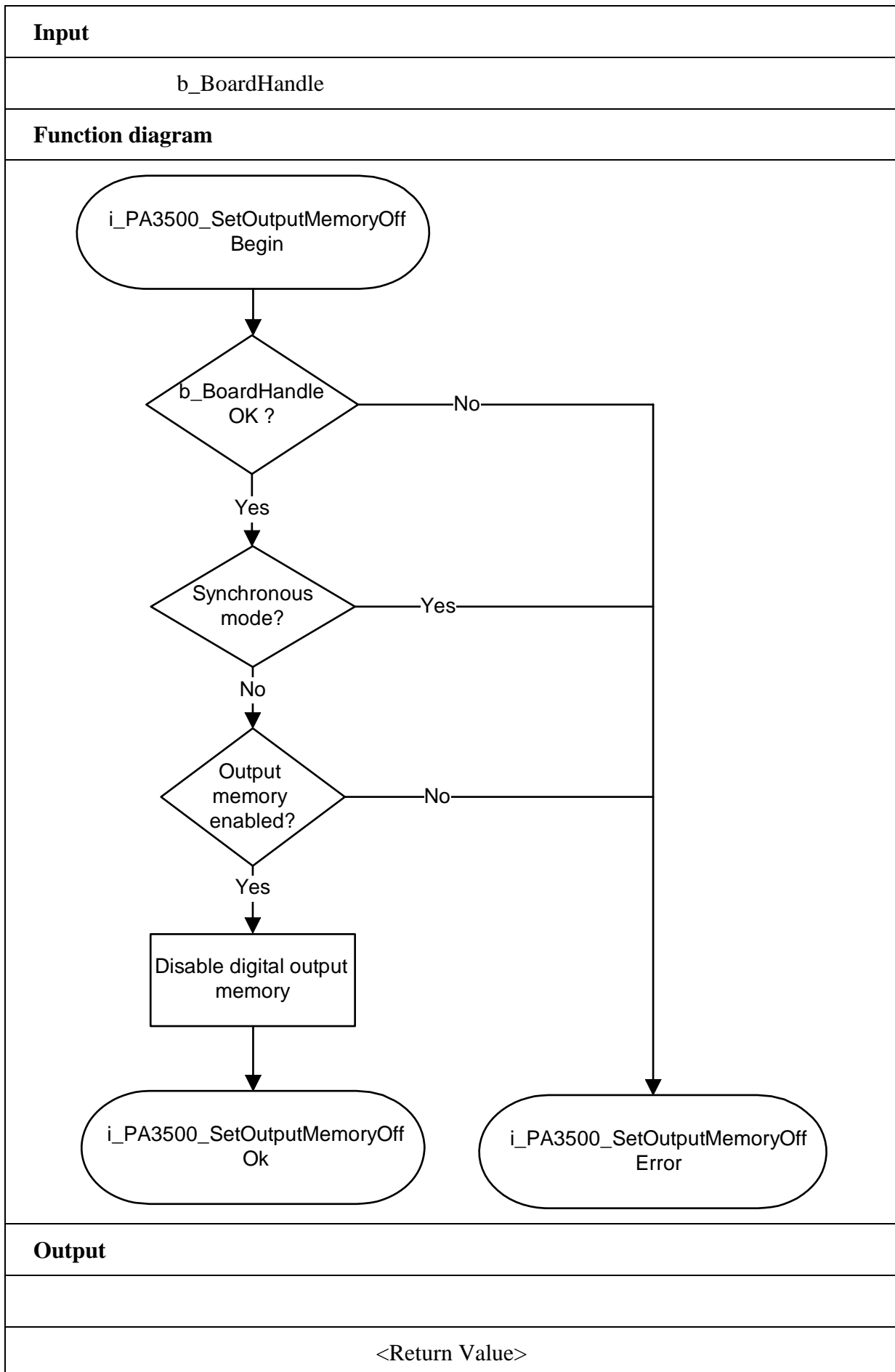
ANSI C:

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

```
i_ReturnValue = i_PA3500_SetOutputMemoryOff (b_BoardHandle);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The output memory has not been activated. See function "i_PA3500_SetOutputMemoryOn".
- 3: Board operating mode error. See function "i_PA3500_SetBoardInformation" or "i_PA3500_ChangeBoardOperatingMode".



3) i_PA3500_Set1DigitalOutputOn (...)**Syntax:**

```
<Return value> = i_PA3500_Set1DigitalOutputOn
                                     (BYTE   b_BoardHandle,
                                     BYTE   b_Channel)
```

Parameters:**- Input:**

```
BYTE   b_BoardHandle      Handle of board PA 3500
BYTE   b_Channel          Number of the output to be set (1 or 2).
```

- 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.

Example**- if the digital output memory is switched ON**

see function "i_PA3500_SetOutputMemoryOn (...)"

Enter: b_Channel= 1

»»» Output 1 is set. Output 2 holds its status.

- if the digital output memory is switched OFF

see function "i_PA3500_SetOutputMemoryOff (...)"

Enter: b_Channel= 1

»»» Output 1 is set. Output 2 is reset.

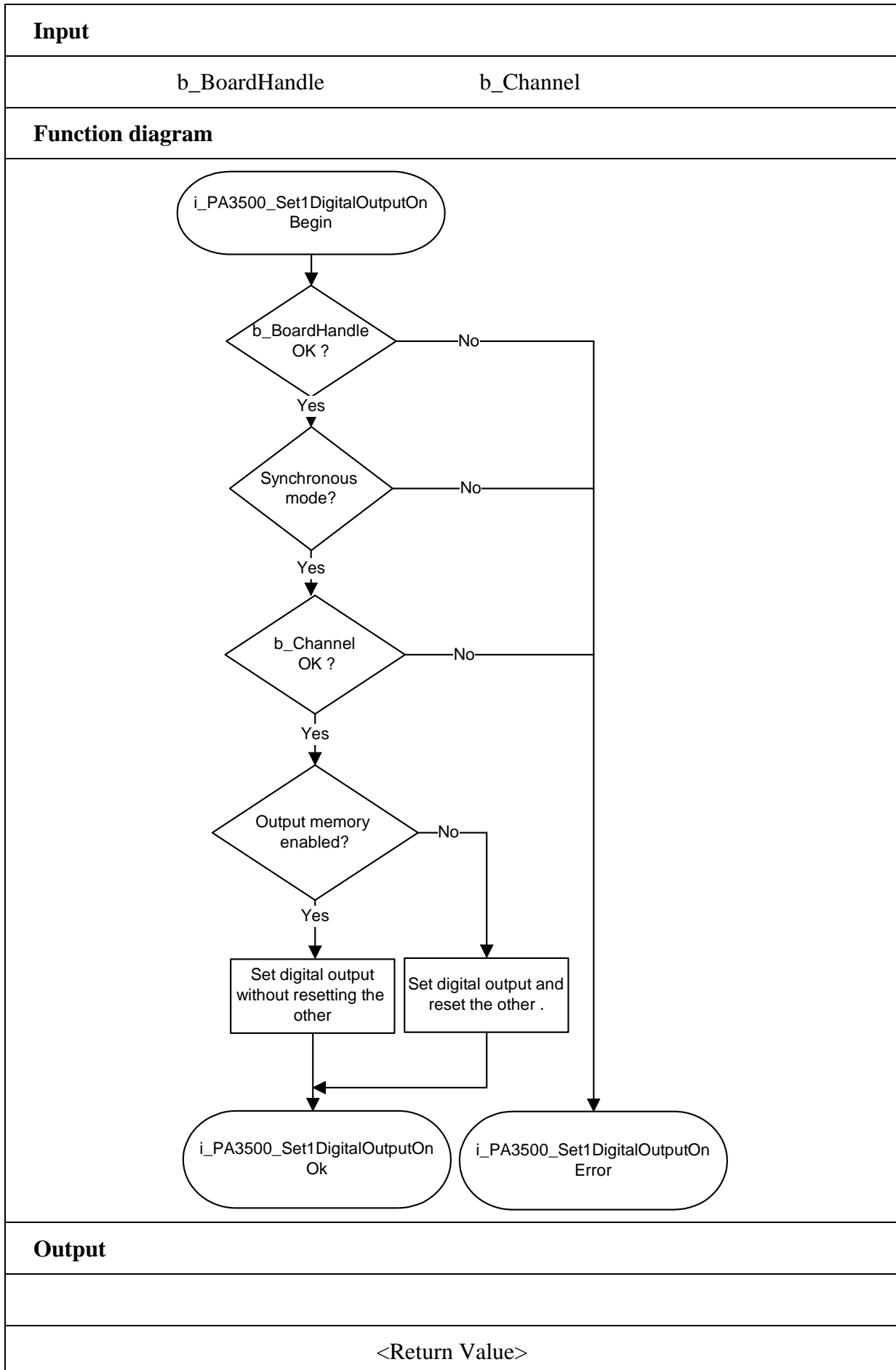
Calling convention:ANSI C:

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

```
i_ReturnValue = i_PA3500_Set1DigitalOutputOn (b_BoardHandle,1);
```

Return value:

```
0: No error
-1: The handle parameter of the board is wrong
-2: The output number is wrong.
-3: Board operating mode error.
```



4) i_PA3500_Set1DigitalOutputOff (...)**Syntax:**

```
<Return value> = i_PA3500_Set1DigitalOutputOff
                                     (BYTE    b_BoardHandle,
                                     BYTE    b_Channel)
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of board PA 3500
BYTE	b_Channel	Number of the output channel to be reset (1 or 2).

- 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.

i**IMPORTANT!**

**You can use this function only if the digital output memory is ON.
See function i_PA3500_SetOutputMemoryOn(..).**

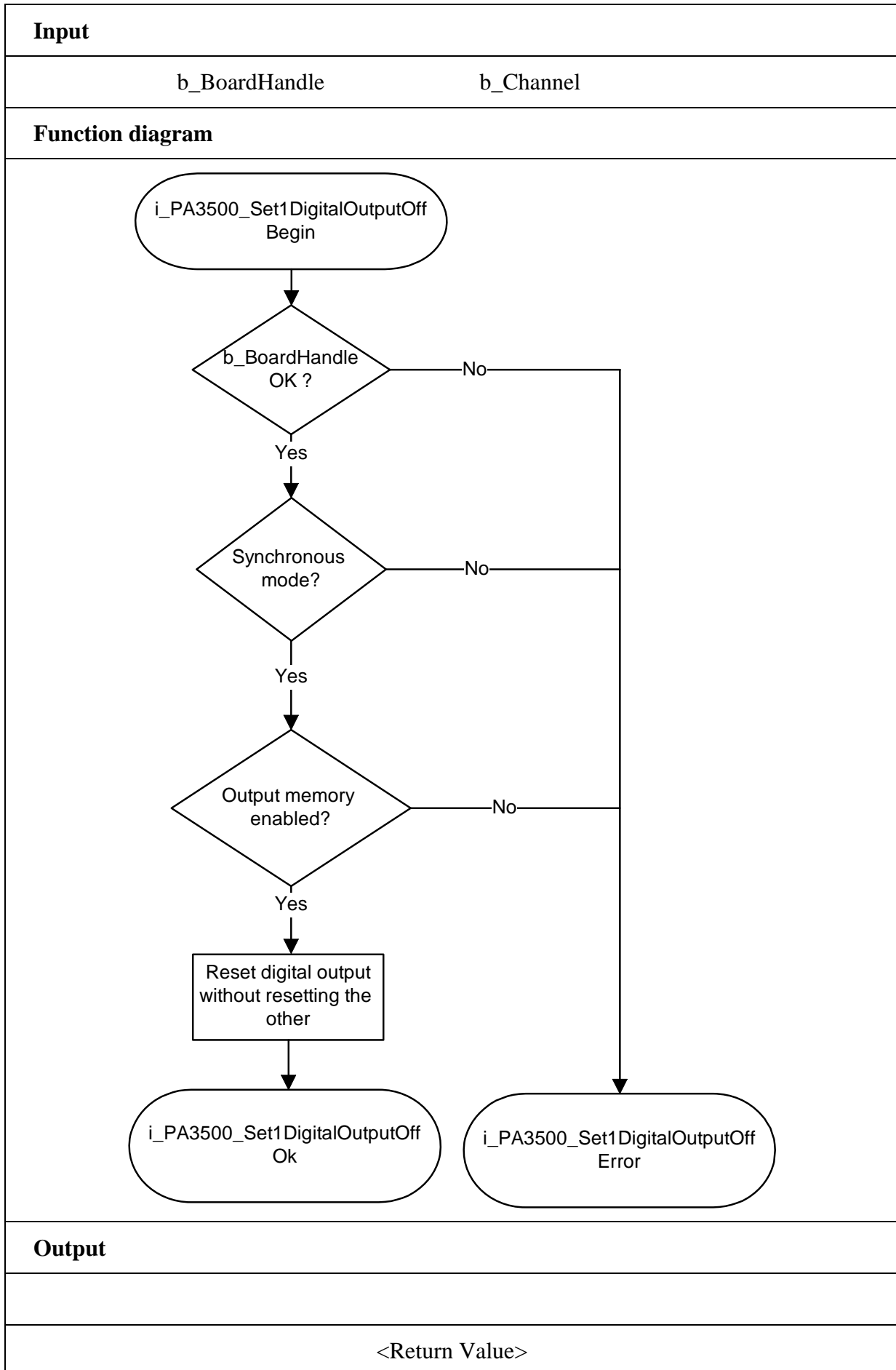
Calling convention:ANSI C:

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

```
i_ReturnValue = i_PA3500_Set1DigitalOutputOff (b_BoardHandle,1);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The output number is wrong.
- 3: Digital output memory off.
Use the function "i_PA3500_SetOutputMemoryOn" first.
- 4: Board operating mode error. See function "i_PA3500_SetBoardInformation" or "i_PA3500_ChangeBoardOperatingMode".



5) i_PA3500_Set2DigitalOutputOn (...)

Syntax:

```
<Return value> = i_PA3500_Set2DigitalOutputOn
                                     (BYTE    b_BoardHandle,
                                     BYTE    b_Value)
```

Parameters:

- Input:

```
BYTE    b_BoardHandle    Handle of board PA 3500
BYTE    b_Value          Output Value (0 to 3)
```

- Output:

No output signal has occurred.

Task:

Sets one or both outputs. Setting an output means setting an output high.

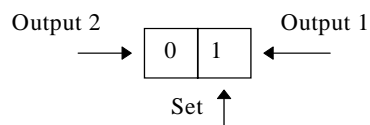
If you have switched off the digital output memory (OFF), the other output is set to "0".

Example:

- if the digital output memory is switched ON

see function "i_PA3500_SetOutputMemoryOn (...)"

Enter: b_Value = 1

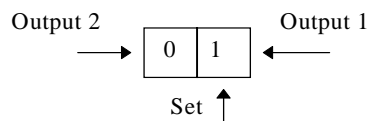


»»» Output 1 is set. Output 2 holds its status.

- if the digital output memory is switched OFF

see function "i_PA3500_SetOutputMemoryOff (...)"

Enter: b_Value = 1



»»» Output 1 is set. Output 2 is reset.

Calling convention:

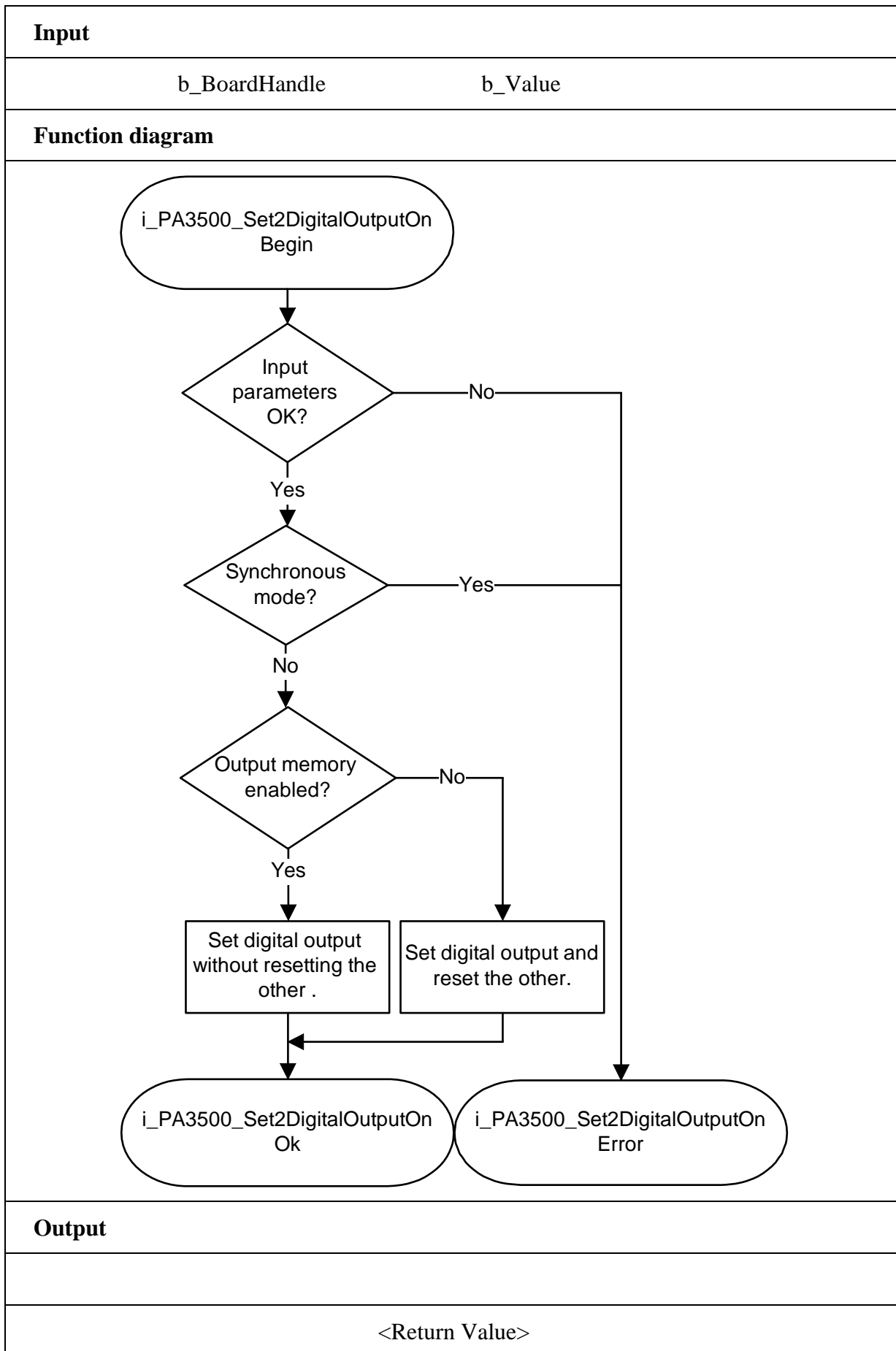
ANSI C:

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

```
i_ReturnValue = i_PA3500_Set2DigitalOutputOn (b_BoardHandle,1);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The output value is wrong.
- 3: Board operating mode error. See function "i_PA3500_SetBoardInformation" or "i_PA3500_ChangeBoardOperatingMode".



6) i_PA3500_Set2DigitalOutputOff (...)**Syntax:**

```
<Return value> = i_PA3500_Set2DigitalOutputOff
                                     (BYTE    b_BoardHandle,
                                     BYTE    b_Value)
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of board PA 3500
BYTE	b_Value	Output value (0 to 3)

- Output:

No output signal has occurred.

Task:

Resets one or both outputs. Resetting an output means setting an output low.

i**IMPORTANT!**

You can use this function only if the digital output memory is ON. See function i_PA3500_SetOutputMemoryOn(..).

Calling convention:

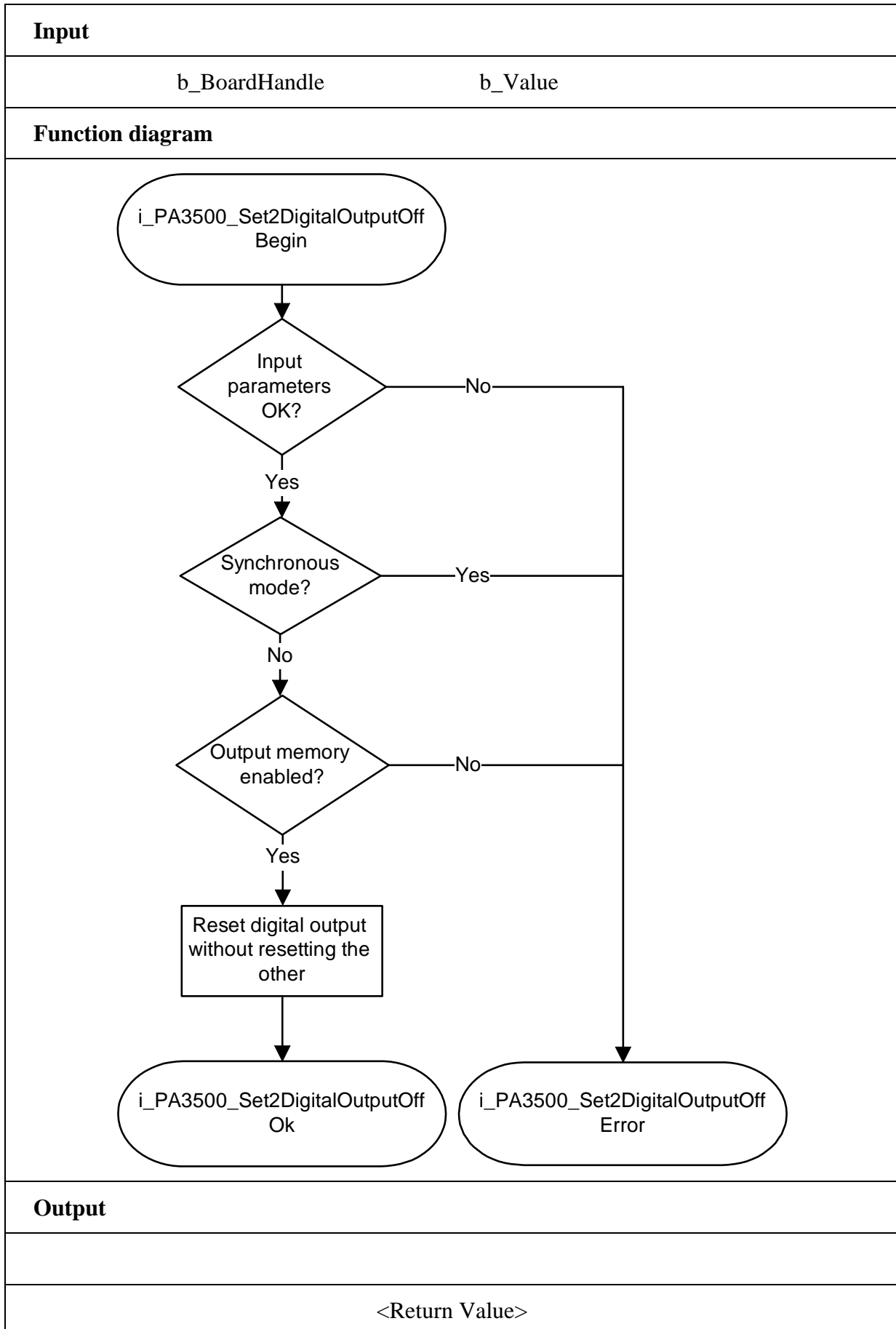
ANSI C:

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

```
i_ReturnValue = i_PA3500_Set2DigitalOutputOff (b_BoardHandle,1);
```

Return value:

- 0: No error
- 1: The handle parameter of the board is wrong
- 2: The output value is wrong.
- 3: Digital output memory off.
Use the function "i_PA3500_SetOutputMemoryOn" first.
- 4: Board operating mode error. See function "i_PA3500_SetBoardInformation" or "i_PA3500_ChangeBoardOperatingMode".



3.6 Digital input channels

1) i_PA3500_Read1DigitalInput (...)

Syntax:

```
<Return value> = i_PA3500_Read1DigitalInput
                    (BYTE b_BoardHandle,
                     BYTE b_Channel,
                     PBYTE pb_ChannelValue)
```

Parameters:

- Input:

BYTE	b_BoardHandle	Handle of board PA 3500
BYTE	b_Channel	Number of the input to be read (1 or 2).

- Output:

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

Task:

Indicates the status of an input . The variable b_Channel passes the input to be read (1 or 2). A value is returned with the variable pb_ChannelValue: 0 (Low), 1 (High).

Calling convention:

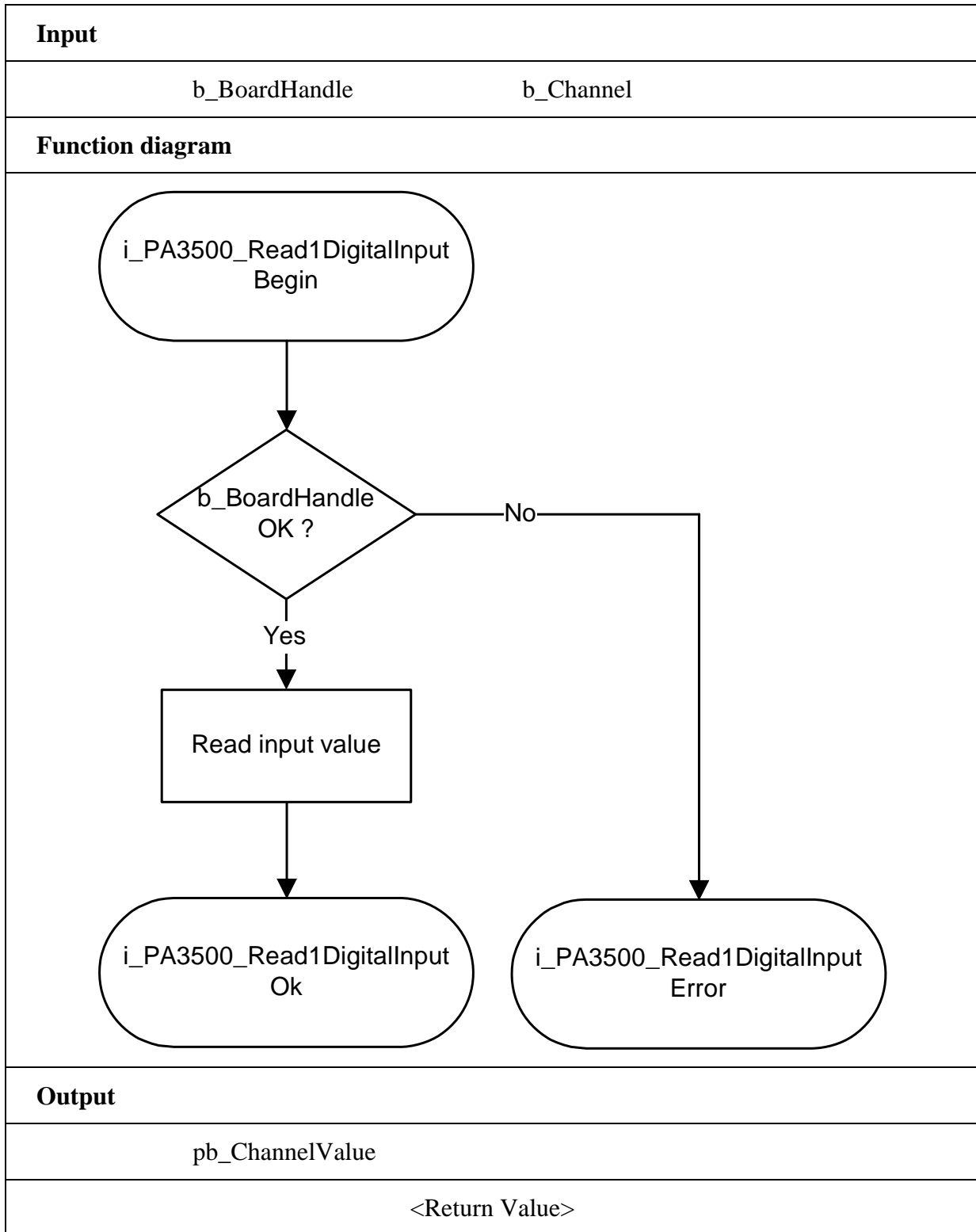
ANSI C:

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

```
i_ReturnValue = i_PA3500_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 2.



2) i_PA3500_Read2DigitalInput (...)

Syntax:

```
<Return value> = i_PA3500_Read2DigitalInput
                    (BYTE  b_BoardHandle,
                     PBYTE pb_PortValue)
```

Parameters:

- Input:

BYTE b_BoardHandle Handle of board **PA 3500**

- Output:

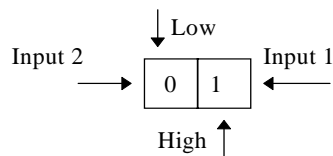
PBYTE pb_PortValue Status of the digital input Port (0 to 3)

Task:

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

Example:

pb_PortValue = 1 Hex



A voltage is present on input 1
There is no voltage on input 2

Calling convention:

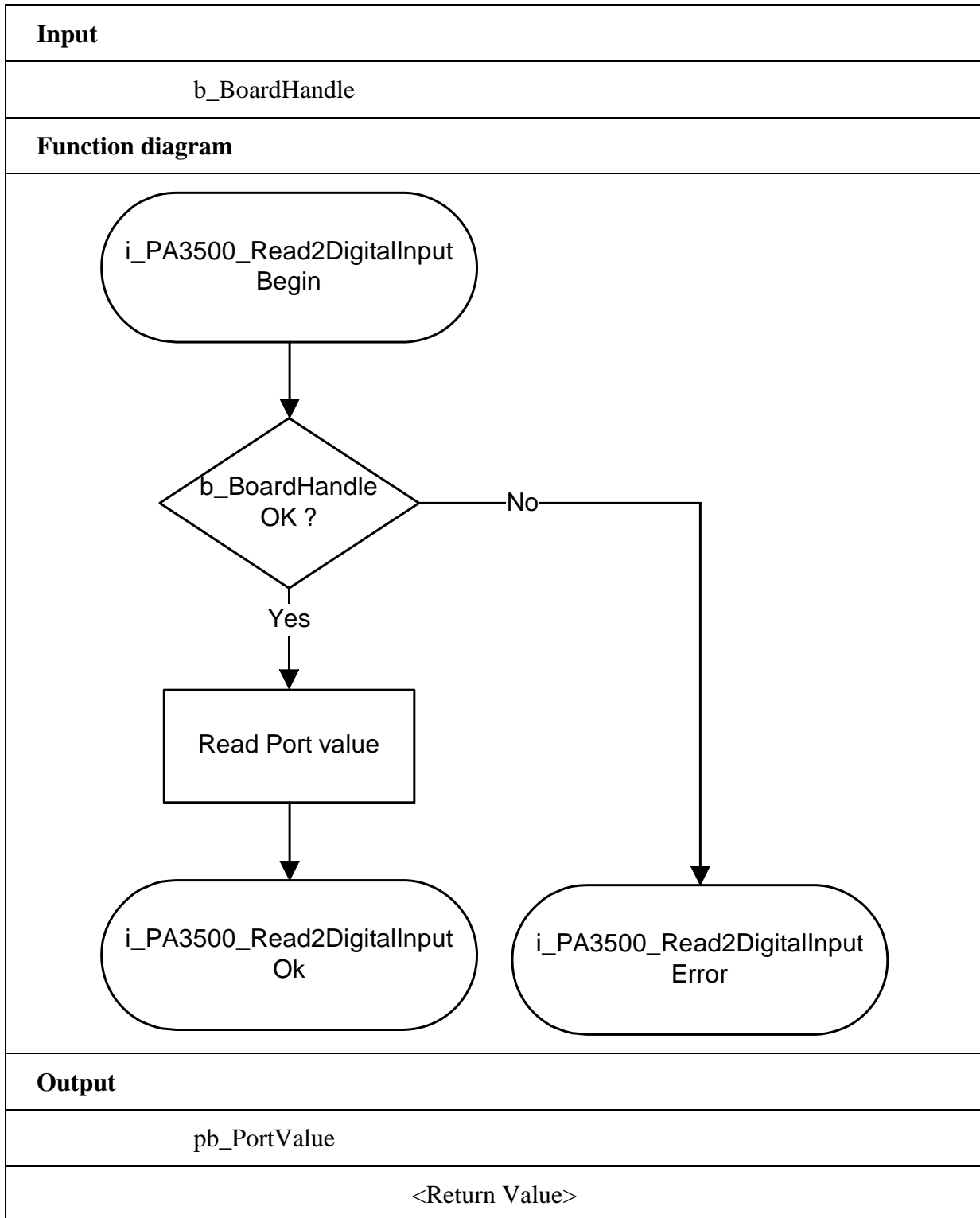
ANSI C:

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

```
i_ReturnValue = i_PA3500_Read2DigitalInput (b_BoardHandle,
                                             &pb_PortValue);
```

Return value:

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



3) _PA3500_EnableDigitalInputInterrupt (...)**Syntax:**

```
<Return value> = i_PA3500_EnableDigitalInputInterrupt
                    (BYTE b_BoardHandle,
                     BYTE b_ChannelValue)
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of board PA 3500
BYTE	b_ChannelValue	1: Interrupt of dig. input 1 is enabled 2: Interrupt of dig. input 2 is enabled 3: Interrupt of dig. input 1 & 2 are enabled

- Output:

No output signal has occurred.

Task:

Enables the interrupt handling of the digital input(s) selected by b_ChannelValue.

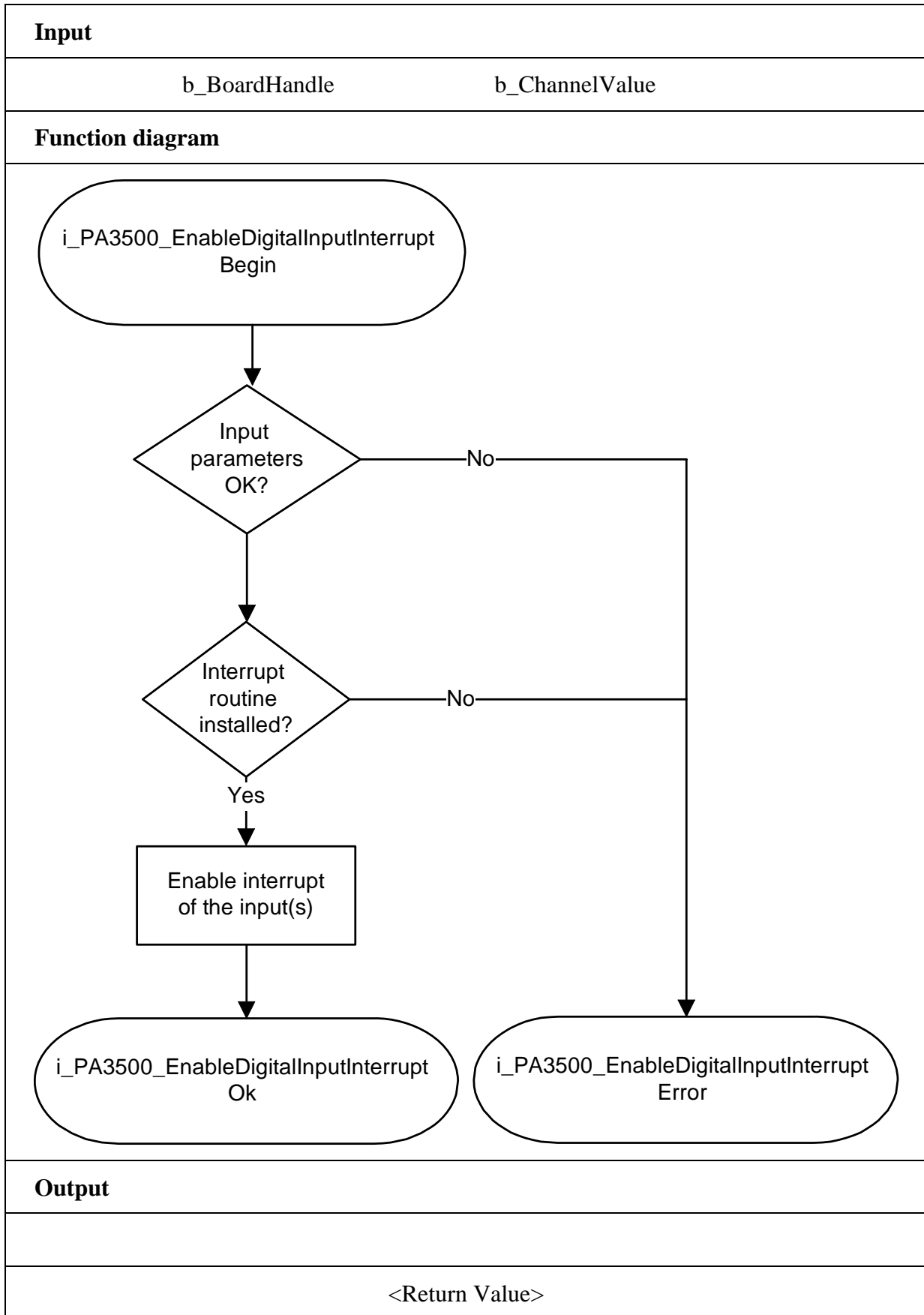
Calling convention:ANSI C:

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

```
i_ReturnValue = i_PA3500_EnableDigitalInputInterrupt (b_BoardHandle,
                                                       3);
```

Return value:

0: No error.
-1: The handle parameter of the board is wrong.
-2: The user interrupt routine is not installed.
-3: The Parameter *b_ChannelValue* is wrong.



4) i_PA3500_DisableDigitalInputInterrupt (...)

Syntax:

```
<Return value> = i_PA3500_DisableDigitalInputInterrupt  
                (BYTE b_BoardHandle)
```

Parameters:**- Input:**

BYTE b_BoardHandle Handle of board **PA 3500**

- Output:

No output signal has occurred.

Task:

Disables the interrupt handling of the digital input(s).

Calling convention:ANSI C:

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

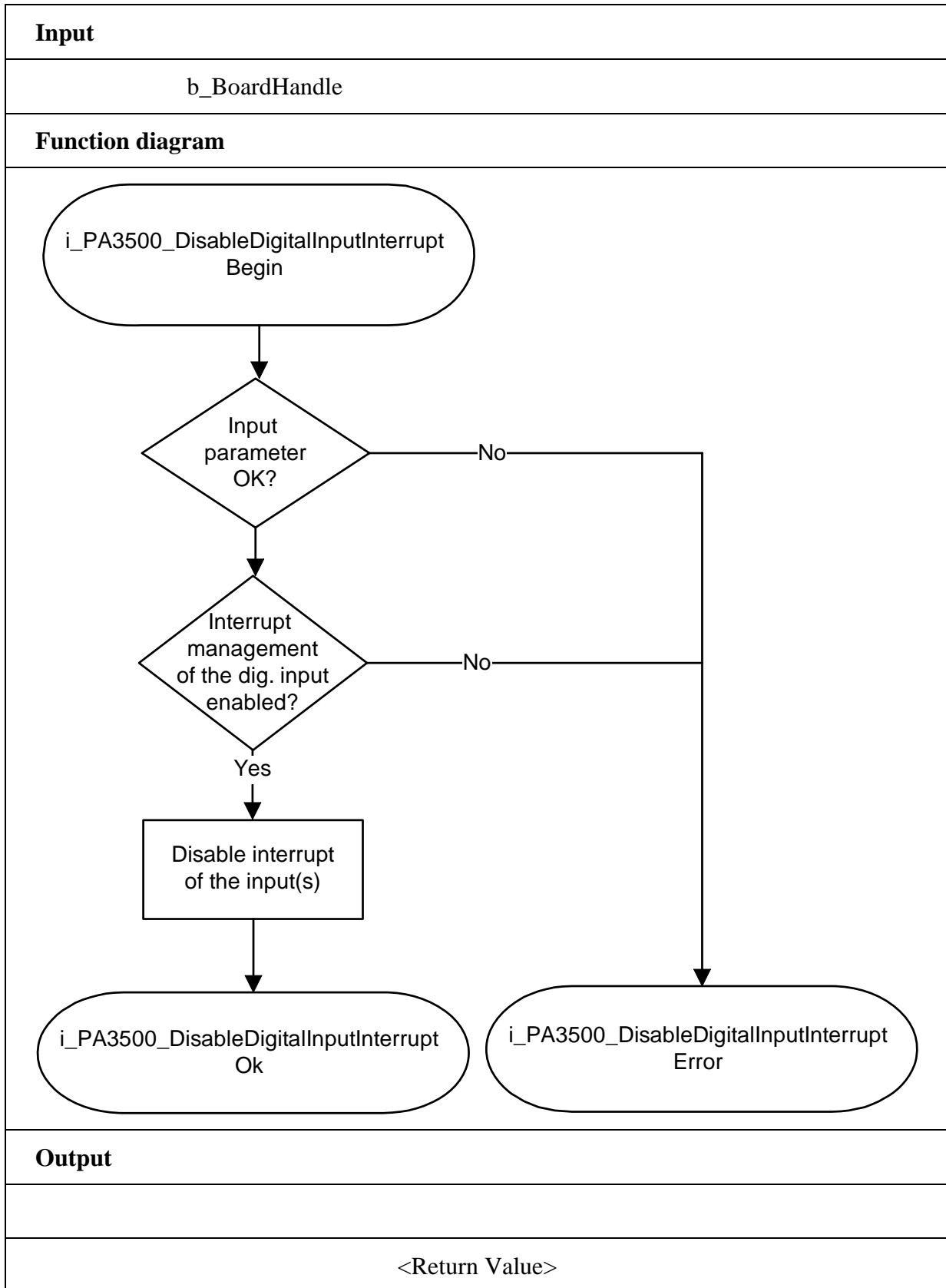
```
i_ReturnValue = i_PA3500_DisableDigitalInputInterrupt (b_BoardHandle);
```

Return value:

0: No error.

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

-2: The digital input interrupt handling has not been enabled. See function
"i_PA3500_EnableDigitalInputInterrupt"



5) i_PA3500_ReadExternTriggerInput (...)**Syntax:**

```
<Return value> = i_PA3500_ReadExternTriggerInput
                    (BYTE b_BoardHandle,
                     PBYTE pb_TriggerValue)
```

Parameters:**- Input:**

BYTE	b_BoardHandle	Handle of board PA 3500
PBYTE	pb_TriggerValue	Status of the input
		0 → no trigger
		1 → trigger

- Output:

No output signal has occurred.

Task:

Indicates the status of the trigger input.

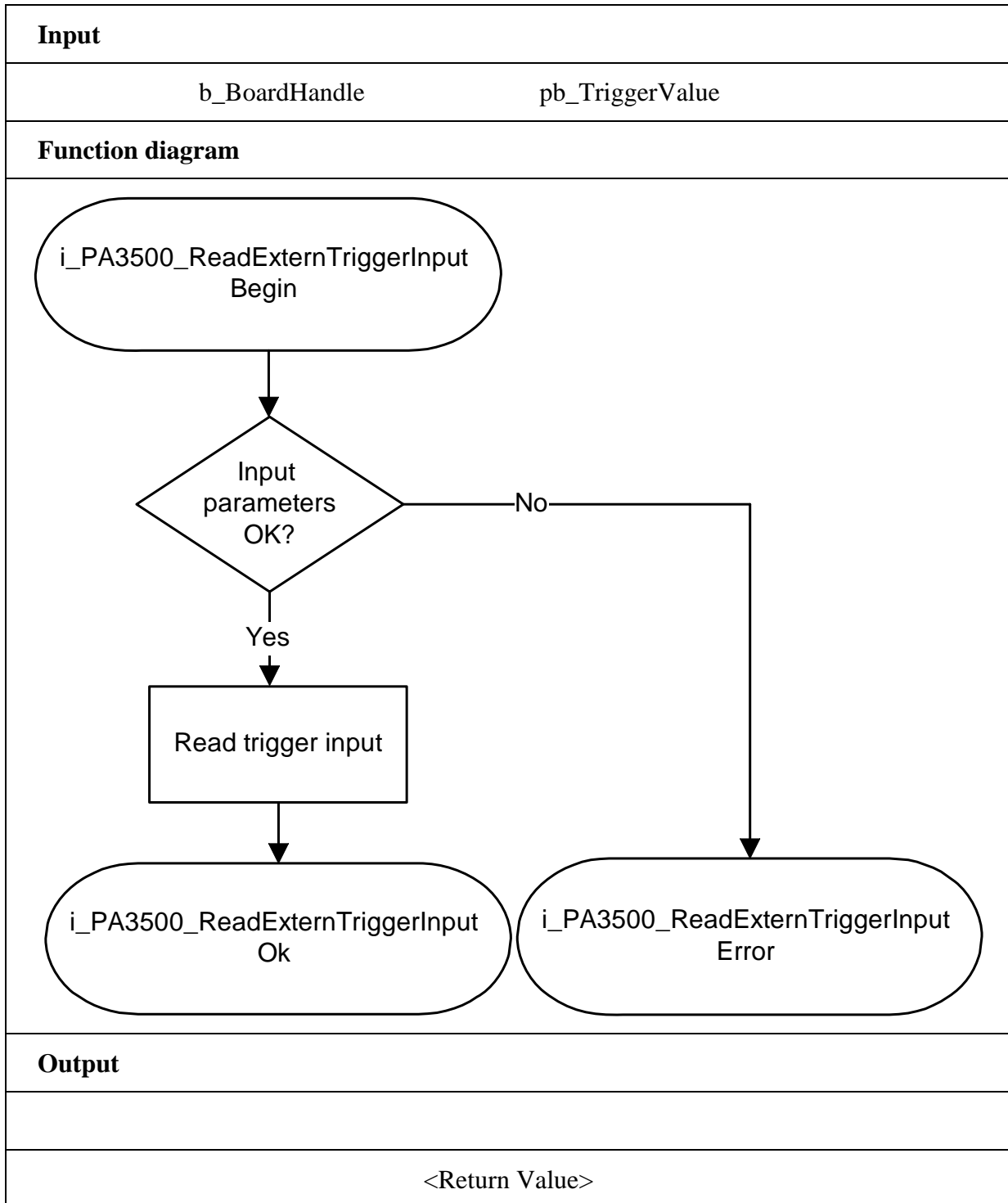
Calling convention:ANSI C:

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

```
i_ReturnValue = i_PA3500_ReadExternTriggerInput (b_BoardHandle,
                                                  &b_TriggerValue);
```

Return value:

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



3.7 Functions in KERNEL Mode

1) i_PA3500_KRNL_Write1AnalogValue (...)

i

IMPORTANT!

This function is only available in PA3500_SIMPLE_MODE.

Syntax:

```
<Return value> = i_PA3500_KRNL_Write1AnalogValue
                                (UINT  ui_Address,
                                BYTE   b_ChannelNbr,
                                BYTE   b_Polarity,
                                UINT   ui_ValueToWrite)
```

Parameters:

- Input:

UINT	ui_Address	Address of the board PA 3500
BYTE	b_ChannelNbr	Number of the selected analog output (0 to 7).
BYTE	b_Polarity	Polarity selection for the analog output channel: - PA3500_UNIPOLAR: Unipolar (0-10V) - PA3500_BIPOLAR: Bipolar ($\pm 10V$)
UINT	ui_ValueToWrite	Analog output value to be written.

- Output:

No output signal has occurred.

Task:

Writes one analog value directly on analog output channel.

If the watchdog is activated and in alarm status, the function returns an error code and does not write the value on the output.

Calling convention:

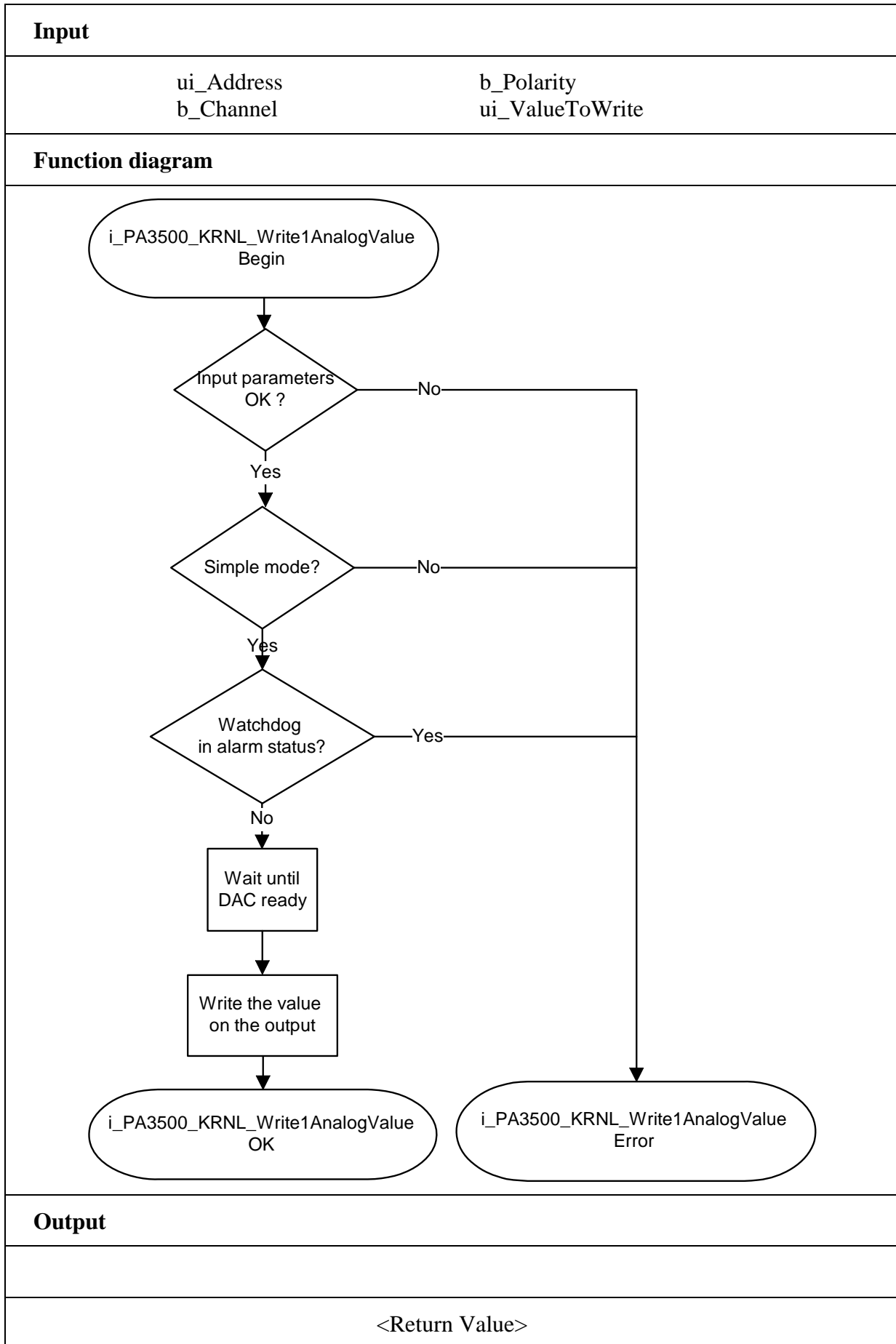
ANSI C:

```
int      i_ReturnValue;
```

```
i_ReturnValue = i_PA3500_KRNL_Write1AnalogValue
                                (0x390,
                                PA3500_CHANNEL0,
                                PA3500_UNIPOLAR,
                                4095);
```

Return value:

0: No error.
-1: The polarity selected is wrong.
-2: The output value is wrong.
-3: The watchdog is in alarm status.
-4: Board operating mode error.



2) i_PA3500_KRNL_StoreAnalogOutputValue (...)

i

IMPORTANT!

This function is only available in PA3500_TRIGGER_MODE or PA3500_SYNCHRONISATION_MODE.

Syntax:

```
<Return value> = i_PA3500_KRNL_StoreAnalogOutputValue
                    (UINT  ui_Address,
                     BYTE  b_EraseLastStorage,
                     BYTE  b_ChannelNbr,
                     BYTE  b_Polarity,
                     UINT  ui_ValueToWrite)
```

Parameters:

- Input:

UINT	ui_Address	Address of the board PA 3500
BYTE	b_EraseLastStorage	Chooses the function course (see table 3-5). - PA3500_ENABLE: Overwrites the last storage and quits the function with a warning code. - PA3500_DISABLE: Quits the function with an error code.
BYTE	b_ChannelNbr	Number of the selected analog output (0 to 7).
BYTE	b_Polarity	Polarity selection for the analog output channel: - PA3500_UNIPOLAR: Unipolar (0-10V) - PA3500_BIPOLAR: Bipolar ($\pm 10V$)
UINT	ui_ValueToWrite	Analog output value to write (0 to 4095). See table 3-4.

- Output:

No output signal has occurred.

Task:

Stores the analog value and the polarity selection of an output in the output buffer. The output value buffer writes on the output when an external trigger occurs or when the external trigger is simulated with the function "i_PA3500_SimulateExternalTrigger".

Calling convention:

ANSI C :

```
int    i_ReturnValue;
```

```
i_ReturnValue = i_PA3500_KRNL_StoreAnalogOutputValue
                (0x390,
                 PA3500_ENABLE,
                 PA3500_CHANNEL0,
                 PA3500_UNIPOLAR,
                 4095);
```