



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

**Software description**

**ADDIDRIVER**

**Digital input channels**

Edition: 07.02 – 04/2006

<b>1</b>	<b>INTRODUCTION .....</b>	<b>1</b>
<b>2</b>	<b>DIGITAL INPUT CHANNELS .....</b>	<b>2</b>
1)	b_ADDIDATA_GetNumberOfDigitalInputs (...)	2
2)	b_ADDIDATA_GetDigitalInputInformation (...)	3
3)	b_ADDIDATA_GetDigitalInputInformationEx (...)	4
4)	b_ADDIDATA_Read1DigitalInput (...)	6
5)	b_ADDIDATA_Read1DigitalInputStatus (...)	7
6)	b_ADDIDATA_Read1DigitalInputValue (...)	8
7)	b_ADDIDATA_Convert1DigitalInputValueInAnalogValue (...)	9
8)	b_ADDIDATA_Read2DigitalInputs (...)	10
9)	b_ADDIDATA_Read2DigitalInputStatus (...)	11
10)	b_ADDIDATA_ReadMoreDigitalInputValue (...)	12
11)	b_ADDIDATA_Read4DigitalInputs (...)	13
12)	b_ADDIDATA_Read4DigitalInputStatus (...)	14
13)	b_ADDIDATA_Read8DigitalInputs (...)	15
14)	b_ADDIDATA_Read8DigitalInputStatus (...)	16
15)	b_ADDIDATA_Read16DigitalInputs (...)	17
16)	b_ADDIDATA_Read16DigitalInputStatus (...)	18
17)	b_ADDIDATA_Read32DigitalInputs (...)	19
18)	b_ADDIDATA_Init1DigitalInputLevel (...)	20
19)	b_ADDIDATA_InitDigitalInputInterrupt (...)	21
20)	b_ADDIDATA_ReleaseDigitalInputInterrupt (...)	26
21)	b_ADDIDATA_EnableDisableDigitalInputInterrupt (...)	27
22)	b_ADDIDATA_Init1DigitalInputLevel (...)	28
23)	b_ADDIDATA_SaveDigitalInputModuleLevel (...)	29
24)	b_ADDIDATA_Read1DigitalInputStatus (...)	30
25)	b_ADDIDATA_ReadMoreDigitalInputStatus (...)	31
26)	b_ADDIDATA_Read1DigitalInputValue (...)	32
27)	b_ADDIDATA_ReadMoreDigitalInputValue (...)	33
28)	b_ADDIDATA_Convert1DigitalInputValueInAnalogValue (...)	34
29)	b_ADDIDATA_GetDigitalInputModuleFilterInformation (...)	35
30)	b_ADDIDATA_InitDigitalInputModuleFilter (...)	37
31)	b_ADDIDATA_EnableDisableDigitalInputModuleFilter (...)	38
32)	b_ADDIDATA_SetDigitalInputModuleLevelSelection (...)	39
33)	b_ADDIDATA_TestDigitalInputAsynchronousFIFOFull (...)	40

## Tables

Table 1-1: Type Declaration for Windows 98/NT/2000/XP .....	1
Table 2-1: Principle of the AND logic (for 2 channels) .....	24
Table 2-2: Filter timer units    Table 2-3: Resolution .....	35

# 1 INTRODUCTION

**i**

## IMPORTANT!

Please note the following conventions in the text:

Function: "b\_ADDIDATA\_GetNumberOfAnalogInputs"  
Variable *dw\_DriverHandle*

Table 1-1: Type Declaration for Windows 98/NT/2000/XP

	Borland C	Microsoft C	Borland Pascal	Microsoft Visual Basic Windows
<b>VOID</b>	void	void	pointer	any
<b>BYTE</b>	unsigned char	unsigned char	byte	integer
<b>INT</b>	int	int	integer	integer
<b>WORD</b>	unsigned short int	unsigned short int	long	long
<b>DWORD</b>	long	long	longint	long
<b>PBYTE</b>	unsigned char *	unsigned char *	var byte	integer
<b>PINT</b>	int *	int *	var integer	integer
<b>PWORD</b>	unsigned short int *	unsigned short int *	var long	long
<b>PCHAR</b>	char *	char *	var string	string
<b>PDWORD</b>	long *	long *	var longint	long
<b>DOUBLE</b>	double	double	double	double

## 2 DIGITAL INPUT CHANNELS

### 1) b\_ADDIDATA\_GetNumberOfDigitalInputs (...)

#### Syntax:

```
<Return value> = b_ADDIDATA_GetNumberOfDigitalInputs
                                     (DWORD   dw_DriverHandle,
                                     PWORD    pw_NumberOfChannels)
```

#### Parameters:

##### - Input:

DWORD *dw\_DriverHandle*      Driver handle

##### - Output:

PWORD *pw\_NumberOfChannels*      Number of digital input channels

#### Task:

Returns the number of digital input channels.

#### Calling convention:

##### ANSI C:

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
WORD      w_NumberOfChannels;
```

```
b_ReturnValue = b_ADDIDATA_GetNumberOfDigitalInput
                (dw_DriverHandle,
                &w_NumberOfChannels);
```

#### Return value:

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

**2) b\_ADDIDATA\_GetDigitalInputInformation (...)****Syntax:**

```
<Return value> = b_ADDIDATA_GetDigitalInputInformation
                                     (DWORD   dw_DriverHandle,
                                     WORD     w_DigitalInputNumber,
                                     PBYTE    pb_DigitalInputType,
                                     PDWORD   pdw_DigitalInputInterrupt)
```

**Parameters:****- Input:**

DWORD	dw_DriverHandle	Handle of the ADDI-DATA driver
WORD	w_DigitalInputNumber	Number of the selected digital input

**- Output:**

PBYTE	pb_DigitalInputType	Type of the selected input
	0x00	Input type not defined
	1 – 15	5V type
	16 – 31	12 type
	32 – 47	24 type
	48 – 255	Any other type

PDWORD	pdw_DigitalInputInterrupt	Available interrupt for the input
--------	---------------------------	-----------------------------------

**Task:**

Returns information about the digital input channels.

**Calling convention:**ANSI C:

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
BYTE      b_DigitalInputType;
DWORD     dw_DigitalInputInterrupt;
```

```
b_ReturnValue = b_ADDIDATA_GetDigitalInputInformation
                                     (dw_DriverHandle,
                                     0,
                                     &b_DigitalInputType,
                                     &dw_DigitalInputInterrupt);
```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

**3) b\_ADDIDATA\_GetDigitalInputInformationEx (...)****Syntax:**

```
<Return value> = b_ADDIDATA_GetDigitalInputInformationEx
                                     (DWORD   dw_DriverHandle,
                                     WORD     w_DigitalInputNumber,
                                     pstr_GetDigitalInputInformation
                                     ps_DigitalInputInformation,
                                     DWORD   dw_StructSize)
```

**Parameters:****- Input:**

DWORD dw_DriverHandle	Handle of the ADDI-DATA driver
WORD w_DigitalInputNumber	Number of the selected digital input
DWORD dw_StructSize	Size of the Digital Input Information Structure. (In byte)

**- Output:**

pstr_GetDigitalInputInformation	ps_DigitalInputInformation	The digital input information are returned in this structure.
---------------------------------	----------------------------	---

The structure contains:

BYTE	b_DigitalInputType	Type of the selected input
		0x00 Input type not defined
		1 – 15 5 V type
		16 – 31 12 type
		32 – 47 24 type
		48 – 255 Any other type
DWORD	dw_DigitalInputInterrupt	Available interrupt for the input
BYTE	b_InputStatusAvailable	0 : Input status information not available
		1 : Input status information available
BYTE	b_InputStatusResolution	Return the resolution of the input status.
BYTE	b_InputValueAvailable	0 : Input value not available
		1 : Input value available
BYTE	b_InputValueResolution	Return the resolution of the input value
BYTE	b_InputLevelConfigurable	0: Input level not configurable
		1: Input level configurable
BYTE	b_InputLevelNbr	Return the number of Input Level.

**Task:**

Returns information about the digital input channels.

**Calling convention:**

ANSI C :

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
str_GetDigitalInputInformation s_DigitalInputInformation;
```

```
b_ReturnValue = b_ADDIDATA_GetDigitalInputInformationEx
                                     (dw_DriverHandle,
                                     0,
```

```
        & s_DigitalInputInformation,  
  
        sizeof(str_GetDigitalInputInformation));
```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

#### 4) b\_ADDIDATA\_Read1DigitalInput (...)

##### Syntax:

```
<Return value> = b_ADDIDATA_Read1DigitalInput
                                     (DWORD dw_DriverHandle,
                                     WORD  w_Channel,
                                     PBYTE pb_ChannelStatus)
```

##### Parameters:

###### - Input:

DWORD <i>dw_DriverHandle</i>	Driver handle
WORD <i>w_Channel</i>	Number of the input to be read

###### - Output:

PBYTE <i>pb_ChannelStatus</i>	Status of the digital input
	0 → low
	1 → high

##### Task:

Indicates the status of a digital input. The variable *b\_Channel* passes the input to be read. A value is returned through the variable *pb\_ChannelStatus*: 0 (low) or 1 (high).

##### Calling convention:

###### ANSI C :

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
BYTE      b_Value;
```

```
b_ReturnValue = b_ADDIDATA_Read1DigitalInput (dw_DriverHandle,
                                              0,
                                              &b_Value);
```

##### Return value:

1: No error  
 0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.



## 5) b\_ADDIDATA\_Read1DigitalInputStatus (...)

### Syntax:

```
<Return value> = b_ADDIDATA_Read1DigitalInputStatus
                                     (DWORD dw_DriverHandle,
                                     WORD  w_Channel,
                                     PWORD pw_ChannelStatus)
```

### Parameters:

#### - Input:

DWORD	dw_DriverHandle	Handle of the ADDI-DATA driver
WORD	w_Channel	Number of the input to be read

#### - Output:

PWORD	pw_ChannelStatus	Status of the digital input
-------	------------------	-----------------------------

The number of bit is given by the function  
b\_ADDIDATA\_GetDigitalInputInformation.  
Ex : 2 bit :  
00 : Voltage < Compare value A  
01 : Voltage > Compare value A and  
      Voltage < Compare value B  
10 : Voltage > Compare value B

### Task:

Indicates the status of a digital input. The variable *b\_Channel* passes the input to be read (1 to 32). A value is returned through the variable *pw\_ChannelStatus*.

### Calling convention:

#### ANSI C :

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
WORD      w_Value;
```

```
b_ReturnValue = b_ADDIDATA_Read1DigitalInputStatus (dw_DriverHandle,
                                                    0,
                                                    &w_Value);
```

### Return value:

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

## 6) b\_ADDIDATA\_Read1DigitalInputValue (...)

### Syntax:

```
<Return value> = b_ADDIDATA_Read1DigitalInputValue
                                     (DWORD dw_DriverHandle,
                                     WORD  w_Channel,
                                     PWORD pw_ChannelStatus
                                     PDWORD pdw_ChannelValue)
```

### Parameters:

#### - Input:

DWORD	dw_DriverHandle	Handle of the ADDI-DATA driver
WORD	w_Channel	Number of the input to be read

#### - Output:

PWORD	pw_ChannelStatus	Status of the digital input The number of bit is given by the function b_ADDIDATA_GetDigitalInputInformation. Ex : 2 bit : 00 : Voltage < Compare value A 01 : Voltage > Compare value A and Voltage < Compare value B 10 : Voltage > Compare value B
PDWORD	pdw_ChannelValue	Value of the digital input : Use the convert function to have the value in Volt.

### Task:

Indicates the status of a digital input. The variable *b\_Channel* passes the input to be read (1 to 32). A value is returned through the variable *pw\_ChannelStatus*. The value of the input is returned by the variable *pdw\_ChannelValue*. To become the real voltage value, use the function *b\_ADDIDATA\_Convert1DigitalInputValueInAnalogValue*.

### Calling convention:

#### ANSI C:

BYTE	b_ReturnValue;
DWORD	dw_DriverHandle;
WORD	w_Value;
DWORD	dw_Value;

```
b_ReturnValue = b_ADDIDATA_Read1DigitalInputValue (dw_DriverHandle,
                                                    0,
                                                    &w_Value,
                                                    &dw_Value);
```

### Return value:

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

**7) b\_ADDIDATA\_Convert1DigitalInputValueInAnalogValue (...)****Syntax:**

<Return value> = b\_ADDIDATA\_Convert1DigitalInputValueInAnalogValue  
 (DWORD dw\_DriverHandle,  
 WORD w\_Channel,  
 DWORD dw\_DigitalValue,  
 DOUBLE \*pd\_AnalogValue)

**Parameters:****- Input:**

DWORD dw_DriverHandle	Handle of the ADDI-DATA driver
WORD w_Channel	Number of the input to be read
DWORD dw_DigitalValue	Digital value of the input returned by the b_ADDIDATA_Read1DigitalInputValue or b_ADDIDATA_ReadMoreDigitalInputValue.

**- Output:**

DOUBLE *pd_AnalogValue	Value in Volt from the input signal.
------------------------	--------------------------------------

**Task:**

Convert the digital input value in a voltage value..

**Calling convention:**ANSI C:

```

BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
DWORD     dw_Value;
DOUBLE    d_VoltageValue;
```

```

b_ReturnValue = b_ADDIDATA_Convert1DigitalInputValueInAnalogValue
(dw_DriverHandle,
0,
dw_Value,
&d_VoltageValue);
```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

### 8) b\_ADDIDATA\_Read2DigitalInputs (...)

### Syntax:

```
<Return value> = b_ADDIDATA_Read2DigitalInputs
                                     (DWORD    dw_DriverHandle,
                                     BYTE      b_Port,
                                     PBYTE     pb_PortValue)
```

### Parameters:

## - Input

DWORD	<i>dw_DriverHandle</i>	Driver handle
BYTE	<i>b_Port</i>	Number of the 2-bit input port to be read

**- Output:**

PBYTE	<i>pb_PortValue</i>	Status of the digital input ports (0 to 3)
-------	---------------------	---

### Task:

Indicates the status of a 2-bit port. The variable *b\_Port* passes the input to be read. The value is returned through the variable *pb\_PortValue*.

The input range of the channels set to "1" is within the high voltage range (See chapter 4 "Limit values" of the technical manual of the board concerned).  
The input range of the channels set to "1" is within the low voltage range (See chapter 4 "Limit values" of the technical manual of the board concerned).

### Calling convention:

ANSI C :

```

BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
BYTE      b_PortValue;

```

```
b_ReturnValue = b_ADDIDATA_Read2DigitalInputs(dw_DriverHandle,
0,
&b_PortValue);
```

**Return value:**

1: No error  
0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError",  
to find the error source.

**9) b\_ADDIDATA\_Read2DigitalInputStatus (...)****Syntax:**

```
<Return value> = b_ADDIDATA_Read2DigitalInputStatus
                                     (DWORD dw_DriverHandle,
                                     BYTE   b_Port,
                                     PDWORD pdw_PortStatus)
```

**Parameters:****- Input:**

DWORD	dw_DriverHandle	Handle of the ADDI-DATA driver
BYTE	b_Port	Number of the 2-Input port to be read

**- Output:**

PDWORD	pdw_PortStatus	Status of the digital input port. The number of bit is given by the function b_ADDIDATA_GetDigitalInputInformation. Ex : 2 bit : 00 : Voltage < Compare value A 01 : Voltage > Compare value A and Voltage < Compare value B 10 : Voltage > Compare value B
--------	----------------	--

**Task:**

Indicates the status of a 2-Input port. The variable *b\_Port* passes the input to be read. The value is returned through the variable *pdw\_PortStatus*.

**Calling convention:**ANSI C :

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
DWORD     dw_PortStatus;
```

```
b_ReturnValue = b_ADDIDATA_Read2DigitalInputStatus (dw_DriverHandle,
                                                    0,
                                                    &dw_PortStatus);
```

**Return value:**

1: No error  
0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

**10) b\_ADDIDATA\_ReadMoreDigitalInputValue (...)****Syntax:**

<Return value> = b\_ADDIDATA\_ReadMoreDigitalInputValue  
 (DWORD dw\_DriverHandle,  
 WORD w\_NbrOfChannel,  
 PWORD pw\_ChannelArray,  
 PWORD pw\_ChannelStatusArray,  
 PDWORD pdw\_ChannelValueArray)

**Parameters:****- Input:**

DWORD	dw_DriverHandle	Handle of the ADDI-DATA driver
WORD	w_NbrOfChannel	Define the Nbr of Channel in the channel array.
PWORD	pw_ChannelArray	Array composed of the number of the input to be read.

**- Output:**

PWORD	pw_ChannelStatusArray	Status of the digital input
		The number of bit is given by the function b_ADDIDATA_GetDigitalInputInformation.
		Ex : 2 bit :
		00 : Voltage < Compare value A
		01 : Voltage > Compare value A and
		Voltage < Compare value B
		10 : Voltage > Compare value B
PDWORD	pdw_ChannelValueArray	Value of the digital input :
		Use the convert function to have the value in Volt.

**Task:**

Read the analog value of several digital input. To become the real voltage value, use the function *b\_ADDIDATA\_Convert1DigitalInputValueInAnalogValue*.

**Calling convention:**ANSI C:

```

BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
WORD      w_Channel[10];
WORD      w_Value[10];
DWORD     dw_Value[10];

```

```

b_ReturnValue = b_ADDIDATA_ReadMoreDigitalInputValue
(dw_DriverHandle,

```

```

5,
&w_Channel,
&w_Value,
&dw_Value);

```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

### 11) b\_ADDIDATA\_Read4DigitalInputs (...)

### Syntax:

```
<Return value> = b_ADDIDATA_Read4DigitalInputs
                                     (DWORD    dw_DriverHandle,
                                     BYTE      b_Port,
                                     PBYTE     pb_PortValue)
```

### Parameters:

## - Input

DWORD	<i>dw_DriverHandle</i>	Driver handle
BYTE	<i>b_Port</i>	Number of the 4-bit input port to be read

**- Output:**

PBYTE	<i>pb_PortValue</i>	Status of the digital input port (0 to 15)
-------	---------------------	--

### Task:

Indicates the status of a 4-bit port. The variable *b\_Port* passes the input to be read. The value is returned through the variable *pb\_PortValue*.

The input range of the channels set to "1" is within the high voltage range (see chapter 4 "Limit values" of the technical manual of the board concerned).  
The input range of the channels set to "1" is within the low voltage range (See chapter 4 "Limit values" of the technical manual of the board concerned).

### Calling Convention:

ANSI C :

```

BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
BYTE      b_PortValue;

```

```
b_ReturnValue = b_ADDIDATA_Read4DigitalInputs(dw_DriverHandle,
1,
&b_PortValue);
```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError",  
to find the error source.

**12) b\_ADDIDATA\_Read4DigitalInputStatus (...)****Syntax:**

```
<Return value> = b_ADDIDATA_Read4DigitalInputStatus
                                     (DWORD dw_DriverHandle,
                                     BYTE   b_Port,
                                     PDWORD pdw_PortStatus)
```

**Parameters:****- Input:**

DWORD	dw_DriverHandle	Handle of the ADDI-DATA driver
BYTE	b_Port	Number of the 4-Input input port to be read

**- Output:**

PDWORD	pdw_PortStatus	Status of the digital input port. The number of bit is given by the function b_ADDIDATA_GetDigitalInputInformationEx. Ex : 2 bit : 00 : Voltage < Compare value A 01 : Voltage > Compare value A and Voltage < Compare value B 10 : Voltage > Compare value B
--------	----------------	--

**Task:**

Indicates the status of a 4-Input port. The variable *b\_Port* passes the input to be read. The value is returned through the variable *pdw\_PortStatus*.

**Calling convention:**ANSI C :

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
DWORD     dw_PortStatus;
```

```
b_ReturnValue = b_ADDIDATA_Read4DigitalInputStatus (dw_DriverHandle,
                                                    0,
                                                    &dw_PortStatus);
```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.



**13) b\_ADDIDATA\_Read8DigitalInputs (...)****Syntax:**

```
<Return value> = b_ADDIDATA_Read8DigitalInputs
                                     (DWORD    dw_DriverHandle,
                                     BYTE      b_Port,
                                     PBYTE     pb_PortValue)
```

**Parameters:****- Input**

DWORD	<i>dw_DriverHandle</i>	Driver handle
BYTE	<i>b_Port</i>	Number of the 8-bit input port to be read

**- Output:**

PBYTE	<i>pb_PortValue</i>	Status of the digital input port (0 to 255)
-------	---------------------	---

**Task:**

Indicates the status of a 8-bit port. The variable *b\_Port* passes the input to be read. The value is returned through the variable *pb\_PortValue*.

The input range of the channels set to "1" is within the high voltage range (See chapter 4 "Limit values" of the technical manual of the board concerned).  
The input range of the channels set to "1" is within the low voltage range (See chapter 4 "Limit values" of the technical manual of the board concerned).

**Calling convention:**ANSI C:

```
BYTE    b_ReturnValue;
DWORD   dw_DriverHandle;
BYTE    b_PortValue;
```

```
b_ReturnValue = b_ADDIDATA_Read8DigitalInputs    (dw_DriverHandle,
                                                    0,
                                                    &b_PortValue);
```

**Return value:**

1: No error  
0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

#### 14) b\_ADDIDATA\_Read8DigitalInputStatus (...)

### Syntax:

```
<Return value> = b_ADDIDATA_Read8DigitalInputStatus
                    (DWORD dw_DriverHandle,
                     BYTE   b_Port,
                     PDWORD pdw_PortStatus)
```

### Parameters:

**- Input:**

DWORD	dw_DriverHandle	Handle of the ADDI-DATA driver
BYTE	b_Port	Number of the 8-Input input port to be read

**- Output:**

PDWORD pdw_PortStatus	Status of the digital input port. The number of bit is given by the function b_ADDIDATA_GetDigitalInputInformationEx. Ex : 2 bit : 00 : Voltage < Compare value A 01 : Voltage > Compare value A and Voltage < Compare value B 10 : Voltage > Compare value B
-----------------------	--

### Task:

Indicates the status of a 8-Input port. The variable *b\_Port* passes the input to be read. The value is returned through the variable *pdw\_PortStatus*.

### Calling convention:

ANSI C :

BYTE	b_ReturnValue;
DWORD	dw_DriverHandle;
DWORD	dw_PortStatus;

```
b_ReturnValue = b_ADDIDATA_Read8DigitalInputStatus (dw_DriverHandle,  
0,  
&dw_PortStatus);
```

### Return value:

1: No error  
0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError",  
to find the error source.

**15) b\_ADDIDATA\_Read16DigitalInputs (...)****Syntax:**

```
<Return value> = b_ADDIDATA_Read16DigitalInputs
                                (DWORD    dw_DriverHandle,
                                BYTE      b_Port,
                                PWORD     pw_PortValue)
```

**Parameters:****- Input**

DWORD	<i>dw_DriverHandle</i>	Driver handle
BYTE	<i>b_Port</i>	Number of the 16-bit input port to be read

**- Output:**

PWORD	<i>pw_PortValue</i>	Status of the digital input port (0 to 65535)
-------	---------------------	---

**Task:**

Indicates the status of a 16-bit port.

The input range of the channels set to "1" is within the high voltage range  
(See chapter 4 "Limit values" of the technical manual of the board concerned).  
The input range of the channels set to "1" is within the low voltage range  
(See chapter 4 "Limit values" of the technical manual of the board concerned).

**Calling convention:**ANSI C:

```
BYTE    b_ReturnValue;
DWORD   dw_DriverHandle;
WORD    w_PortValue;
```

```
b_ReturnValue = b_ADDIDATA_Read16DigitalInputs    (dw_DriverHandle,
                                                    0,
                                                    &w_PortValue);
```

**Return value:**

1: No error  
0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError",  
to find the error source.

## 16) b\_ADDIDATA\_Read16DigitalInputStatus (...)

### Syntax:

```
<Return value> = b_ADDIDATA_Read16DigitalInputStatus
                                   (DWORD dw_DriverHandle,
                                   BYTE   b_Port,
                                   PDWORD pdw_PortStatus)
```

### Parameters:

#### - Input:

DWORD	dw_DriverHandle	Handle of the ADDI-DATA driver
BYTE	b_Port	Number of the 16-Input input port to be read

#### - Output:

PDWORD	pdw_PortStatus	Status of the digital input port. The number of bit is given by the function b_ADDIDATA_GetDigitalInputInformationEx. Ex : 2 bit : 00 : Voltage < Compare value A 01 : Voltage > Compare value A and Voltage < Compare value B 10 : Voltage > Compare value B
--------	----------------	--

### Task:

Indicates the status of a 16-Input port. The variable *b\_Port* passes the input to be read. The value is returned through the variable *pdw\_PortStatus*.

### Calling convention:

#### ANSI C:

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
DWORD     dw_PortStatus;
```

```
b_ReturnValue = b_ADDIDATA_Read16DigitalInputStatus(dw_DriverHandle,
                                                    0,
                                                    &dw_PortStatus);
```

### Return value:

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

**17) b\_ADDIDATA\_Read32DigitalInputs (...)****Syntax:**

```
<Return value> = b_ADDIDATA_Read32DigitalInputs
                                     (DWORD    dw_DriverHandle,
                                     BYTE      b_Port,
                                     PDWORD    pdw_PortValue)
```

**Parameters:****- Input**

DWORD	<i>dw_DriverHandle</i>	Driver handle
BYTE	<i>b_Port</i>	Number of the 32-bit input port to be read

**- Output:**

PDWORD	<i>pdw_PortValue</i>	Status of the digital input port (0 to $2^{32}-1$ )
--------	----------------------	---

**Task:**

Indicates the status of a 32-bit port.

The input range of the channels set to "1" is within the high voltage range  
(See chapter 4 "Limit values" of the technical manual of the board concerned).

The input range of the channels set to "1" is within the low voltage range  
(See chapter 4 "Limit values" of the technical manual of the board concerned).

**Calling convention:**ANSI C:

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
DWORD     dw_PortValue;
```

```
b_ReturnValue = b_ADDIDATA_Read32DigitalInputs (dw_DriverHandle,
                                                0,
                                                &dw_PortValue);
```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError",  
to find the error source.

**18) b\_ADDIDATA\_Init1DigitalInputLevel (...)****Syntax:**

```
<Return value> = b_ADDIDATA_Init1DigitalInputLevel
                                (DWORD   dw_DriverHandle,
                                WORD     w_DigitalInput,
                                DOUBLE   *pd_VoltageValueArray);
```

**Parameters:****- Input:**

DWORD dw\_DriverHandle      Handle of the ADDI-DATA driver  
 WORD w\_DigitalInput      Number of the digital input to configure.  
 DOUBLE \*pd\_VoltageValueArray      Array witch contains the voltage value for the level.

pd\_VoltageValueArray [0] : Determines level A  
 pd\_VoltageValueArray [1] : Determines level B  
 pd\_VoltageValueArray [2] : Determines level C, usw.

**- Output:**

No output signal has occurred.

**Task:**

Initialise the input level voltage of the digital input.

**Calling convention:**ANSI C :

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
DOUBLE    d_VoltageValue[2];
```

```
d_VoltageValue[0] = 10.0;
```

```
d_VoltageValue[1] = 15.0;
```

```
b_ReturnValue = b_ADDIDATA_Init1DigitalInputLevel
                (dw_DriverHandle,
                0, d_VoltageValue);
```

**Return value:**

1: No error

0: Error by calling up of the function. Use the function

"i\_ADDIDATA\_GetLastError", to find the error source.

**19) b\_ADDIDATA\_InitDigitalInputInterrupt (...)****Syntax:**

```
<Return value> = b_ADDIDATA_InitDigitalInputInterrupt
                                (DWORD    dw_DriverHandle,
                                WORD      w_FirstDigitalInput,
                                WORD      w_LastDigitalInput,
                                BYTE      b_InterruptLogic,
                                PDWORD    pdw_DigitalInputArrayMode1,
                                PDWORD    pdw_DigitalInputArrayMode2);
```

**Parameters:****- Input:**

DWORD	<i>dw_DriverHandle</i>	Driver handle
WORD	<i>w_FirstDigitalInput</i>	Number of the first digital input to be initialised.
WORD	<i>w_LastDigitalInput</i>	Number of the last digital input to be initialised.
BYTE	<i>b_InterruptLogic</i>	Interrupt logic for the inputs (0= no, 1= OR, 2= AND).
PDWORD	<i>pdw_DigitalInputArrayMode1</i>	Mask of Mode 1 (see table below).
PDWORD	<i>pdw_DigitalInputArrayMode2</i>	Mask of Mode 2 (see table below).

**- Output:**

No output signal has occurred.

**Task:**

Initialises the digital input interrupt.

**i****IMPORTANT!**

The user has to set only the bits of the inputs which are contained in the range between "w\_FirstDigitalInput" and "w\_LastDigitalInput".

If the user sets other bits out of the defined range, an error is returned.

The OR logic (for edge only) is defined as follows:

If one of the selected edges occurs on the inputs, the interrupt is generated.

The AND logic (status) is defined as follows:

Each time a logic is true (when the rising or falling edge of the last input which verifies the logic occurs), the interrupt is generated.

The AND logic and the OR logic have the same action when both modes (mode1 and mode2) are set to 1 for all inputs (rising or falling edge for OR logic; low and high status for AND logic).

The AND logic and the OR logic have the same action when only one input is considered.

Table 2-1: Overview: Table Index 0

Table Index	0
Mode_1 table values	<i>MSB</i> 0000 0000 0000 0000 0000 0000 0000 0000 <i>LSB</i>
Mode_2 table values	<i>MSB</i> 0000 0000 0000 0000 0000 0000 0000 0000 <i>LSB</i>

Table 2-2: Overview: Table Index 1

Table Index	1
Mode_1 table values	<i>MSB</i> 0000 0000 0000 0000 0000 0000 0000 0000 <i>LSB</i>
Mode_2 table values	<i>MSB</i> 0000 0000 0000 0000 0000 0000 0000 0000 <i>LSB</i>

The Mode\_X table values are binary. Each bit defines the interrupt mask of one input.

One channel is defined with 2 bit, the one bit is from Mode\_1 and the other bit is from Mode\_2. The combination of both bits is shown in the table below:

Table 2-3: Definition of interrupt logic

OR logic	Disable	Rising edge	Falling edge	Rising/falling edge
AND logic	Disable	High level	Low level	High/low level
Mode_1	0	1	0	1
Mode_2	0	0	1	1

### Example:

If you want to use the OR logic with 2 boards of each 32 inputs and you want to have a rising edge on input Index 2, then input Index 36 must react with a falling edge and all other inputs must be disabled. The inputs Index begin with 0.

Table 2-4: Example: Table Index 0

Table Index	0
Mode_1 table values	<i>MSB</i> 0000 0000 0000 0000 0000 0000 0000 0100 <i>LSB</i>
Mode_2 table values	<i>MSB</i> 0000 0000 0000 0000 0000 0000 0000 0000 <i>LSB</i>

In the tables the first bit at the left (*in italics*) stands for channel 31 and the first bit at the right (*in italics*) stands for channel 0.

Table 2-5: Example: Table Index 1

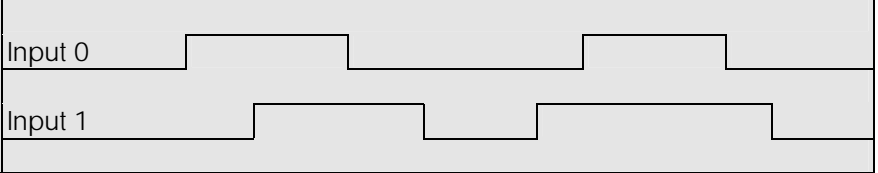
Table Index	1
Mode_1 table values	<i>MSB</i> 0000 0000 0000 0000 0000 0000 0000 0000 <i>LSB</i>
Mode_2 table values	<i>MSB</i> 0000 0000 0000 0000 0000 0000 0001 0000 <i>LSB</i>



The OR logic reacts to rising or falling edges.

An interrupt is generated if an edge modification occurs on an interruptible input and fulfils the interrupt conditions set by Mode\_1 and Mode\_2.

**Table 2-6: Principle of the OR logic (for 2 channels)**

INPUT 1 MODE2, MODE1	INPUT 0 MODE2, MODE1								
		Input 0							
		Input 1							
0,0	0,0	No interrupt							
1,0	0,0	10*   10							
0,1	0,0	10   10							
1,1	0,0	10   10   10   10							
0,0	1,0	01**   01							
1,0	1,0	01   10   01   10							
0,1	1,0	10   01   10   01							
1,1	1,0	10   01   10   10   01   10							
0,0	0,1	01   01							
1,0	0,1	01   10   01   10							
0,1	0,1	01   10   10   01							
1,1	0,1	01   10   10   10   01   10							
0,0	1,1	01   01   01   01							
1,0	1,1	01   01   10   01   01   10							
0,1	1,1	01   10   01   10   01   01							
1,1	1,1	01   10   01   10   10   01   01   10							

\* 10: **Interrupt source:** the 2<sup>nd</sup> input (Channel 1) has generated an interrupt.

\*\* 01: **Interrupt source:** the 1<sup>st</sup> input (Channel 0) has generated an interrupt.

## AND Logic

The AND logic reacts to a change in the level of the selected inputs.

An interrupt is generated each time when the following conditions are fulfilled:

- the interruptible inputs fulfil the conditions set in Mode 1 and Mode 2.
- the IRQ condition must be fulfilled during a determined time interval (differs according to the board specifications: see in the corresponding board manual).

Possible bounce pulses are thus eliminated.

- after an interrupt, a level modification is to occur on the interruptible inputs so that the interrupt logic is released.

**Table 2-1: Principle of the AND logic (for 2 channels)**

INPUT 1 MODE2, MODE1	INPUT 0 MODE2, MODE1	Input 0
		Input 1
0,0	0,0	Kein Interrupt
1,0	0,0	
0,1	0,0	
1,1	0,0	
0,0	1,0	
1,0	1,0	
0,1	1,0	
1,1	1,0	
0,0	0,1	
1,0	0,1	
0,1	0,1	
1,1	0,1	
0,0	1,1	
1,0	1,1	
0,1	1,1	
1,1	1,1	

**Calling convention:**ANSI C :

```

BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
DWORD     dw_DigitalInputArrayMode1[2];
           dw_DigitalInputArrayMode2[2];
           dw_DigitalInputArrayMode 1 [0] = 4;
           (virtual bit 31 0000 0000 0000 0000 0000 0000 000 0100 virtual bit 0

           dw_DigitalInputArrayMode1 [1] = 0
           virtual bit 63 0000 0000 0000 0000 0000 0000 0000 0000 virtual bit 32

           dw_DigitalInputArrayMode2 [0] = 0
           virtual bit 31 0000 0000 0000 0000 0000 0000 0000 0000 virtual bit 0

           dw_DigitalInputArrayMode2 [1] = 16

           virtual bit 63 0000 0000 0000 0000 0000 0000 0001 0000 virtual bit 32

```

```

b_ReturnValue = b_ADDIDATA_InitDigitalInputInterrupt
                (dw_DriverHandle,
                 0,
                 63,
                 ADDIDATA_OR,
                 dw_DigitalInputArrayMode1,
                 dw_DigitalInputArrayMode2);

```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

**i****IMPORTANT!**

The **interrupt mask** for the functionality is detailed in the "**Interrupt**" function description. (Tables 2-1 and 2-2).

**20) b\_ADDIDATA\_ReleaseDigitalInputInterrupt (...)****Syntax:**

```
<Return value> = b_ADDIDATA_ReleaseDigitalInputInterrupt
                                (DWORD   dw_DriverHandle,
                                WORD      w_FirstDigitalInput,
                                WORD      w_LastDigitalInput);
```

**Parameters:****- Input:**

DWORD <i>dw_DriverHandle</i>	Driver handle
WORD <i>w_FirstDigitalInput</i>	Number of the first digital input to be released.
WORD <i>w_LastDigitalInput</i>	Number of the last digital input to be released.

**- Output:**

No output signal has occurred.

**Task:**

Releases the digital input interrupt.

**Calling convention:**ANSI C:

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
```

```
b_ReturnValue = b_ADDIDATA_ReleaseDigitalInputInterrupt
                (dw_DriverHandle,
                 0,
                 15);
```

**Return value:**

1: No error  
 0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError",  
 to find the error source.

**21) b\_ADDIDATA\_EnableDisableDigitalInputInterrupt (...)****Syntax:**

```
<Return value> = b_ADDIDATA_EnableDisableDigitalInputInterrupt
                                (DWORD dw_DriverHandle,
                                WORD w_FirstDigitalOutput,
                                WORD w_LastDigitalOutput,
                                BYTE b_InterruptFlag);
```

**Parameters:****- Input:**

DWORD	<i>dw_DriverHandle</i>	Driver handle
WORD	<i>w_FirstDigitalInput</i>	Number of the first digital input to be enabled or disabled
WORD	<i>w_LastDigitalInput</i>	Number of the last digital input to be enabled or disabled
BYTE	<i>b_InterruptFlag</i>	Interrupt flag (0 = disable, 1 = enable).

**- Output:**

No output signal has occurred.

**Task:**

Enables or disables the digital input interrupt.

**Calling convention:**ANSI C:

```
BYTE    b_ReturnValue;
DWORD   dw_DriverHandle;
```

```
b_ReturnValue = b_ADDIDATA_EnableDisableDigitalInputInterrupt
                (dw_DriverHandle,
                0,
                64,
                ADDIDATA_ENABLE);
```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

**22) b\_ADDIDATA\_Init1DigitalInputLevel (...)****Syntax:**

```
<Return value> = b_ADDIDATA_Init1DigitalInputLevel
                    (DWORD    dw_DriverHandle,
                     WORD     w_DigitalInput,
                     DOUBLE   *pd_VoltageValueArray)
```

**Parameters:****- Input:**

DWORD	<i>dw_DriverHandle</i>	Driver handle
WORD	<i>w_DigitalInput</i>	Number of the digital input to be configured.
DOUBLE	<i>*pd_VoltageValueArray</i>	Array to determine the voltage value of the levels.
		pd_VoltageValueArray [0] : Determines level A
		pd_VoltageValueArray [1] : Determines level B
		pd_VoltageValueArray [2] : Determines level C
		...

**- Output:**

No output signal has occurred.

**Task:**

Initialises the input voltage level of the digital input.

**Calling convention:**ANSI C :

```
BYTE    b_ReturnValue;
DWORD   dw_DriverHandle;
DOUBLE  d_VoltageValue[2] = { 10.0, 15.0};
```

```
b_ReturnValue = b_ADDIDATA_Init1DigitalInputLevel
                    (dw_DriverHandle,
                     0,
                     d_VoltageValue);
```

**Return value:**

1: No error

0: Error by calling up of the function. Use the function  
"i\_ADDIDATA\_GetLastError", to find the error source.

### 23) b\_ADDIDATA\_SaveDigitalInputModuleLevel (...)

**Syntax:**

<Return value> = b\_ADDIDATA\_SaveDigitalInputModuleLevel  
(DWORD dw\_DriverHandle,  
DWORD dw\_DigitalModuleNumber)

**Parameters:****- Input:**

DWORD *dw\_DriverHandle* Handle of the ADDI-DATA driver  
DWORD *dw\_DigitalModuleNumber* Number of the digital input module

**- Output:**

No output signal has occurred.

**Task:**

Saves the selected level configuration to the hardware.

**Calling convention:**ANSI C :

BYTE b\_ReturnValue;  
DWORD dw\_DriverHandle;

b\_ReturnValue = b\_ADDIDATA\_SaveDigitalInputModuleLevel  
(dw\_DriverHandle,  
0);

**Return value:**

1: No error

0: Error by callup of the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

**24) b\_ADDIDATA\_Read1DigitalInputStatus (...)****Syntax:**

```
<Return value> = b_ADDIDATA_Read1DigitalInputStatus
                    (DWORD dw_DriverHandle,
                     WORD w_Channel,
                     PWORD pw_ChannelStatus)
```

**Parameters:****- Input:**

DWORD <i>dw_DriverHandle</i>	Driver handle
WORD <i>w_Channel</i>	Number of the input to be read

**- Output:**

PWORD <i>pw_ChannelStatus</i>	Status of the digital input The number of bits is given by the function <code>b_ADDIDATA_GetDigitalInputInformationEx</code> .
-------------------------------	--

**Task:**

Indicates the status of a digital input. The variable *w\_Channel* passes the input to be read. A value is returned through the variable *pw\_ChannelStatus*. The value is coded on 2 bits.

Ex : 2 bit :

00 : Voltage <= Compare value A

01 : Voltage > Compare value A and Voltage < Compare value B

11 : Voltage >= Compare value B

**Calling convention:**ANSI C :

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
WORD      w_ChannelStatus;
```

```
b_ReturnValue = b_ADDIDATA_Read1DigitalInputStatus
                (dw_DriverHandle,
                 0,
                 &w_ChannelStatus);
```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "`i_ADDIDATA_GetLastError`", to find the error source.



**25) b\_ADDIDATA\_ReadMoreDigitalInputStatus (...)****Syntax:**

```
<Return value> = b_ADDIDATA_ReadMoreDigitalInputStatus
                                (DWORD   dw_DriverHandle,
                                WORD     w_NbrOfChannel,
                                PWORD    pw_ChannelArray,
                                PWORD    pw_ChannelArrayStatus)
```

**Parameters:****- Input:**

DWORD	<i>dw_DriverHandle</i>	Driver handle
WORD	<i>w_NbrOfChannel</i>	Defines the number of channels in the channel array.
PWORD	<i>pw_ChannelArray</i>	Array composed of the number of inputs to be read.

**- Output:**

PWORD	<i>pw_ChannelArrayStatus</i>	Status of the selected digital input channels. <i>pw_ChannelArrayStatus</i> [0]: Returns the status of the first selected channel <i>pw_ChannelArrayStatus</i> [1]: Returns the status of the second selected channel <i>pw_ChannelArrayStatus</i> [ <i>w_NbrOfChannel</i> - 1]: Returns the status of the last selected channel The number of bits is given by the function <i>b_ADDIDATA_GetDigitalInputInformationEx</i> . 0 → low 1 → high
-------	------------------------------	--

**Task:**

Reads the status of several digital inputs.

**Calling convention:**ANSI C:

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
WORD      w_ChannelArray[3];
WORD      w_ChannelArrayStatus [3];
```

```
w_ChannelArray [0] = 0; // Select channel 0
w_ChannelArray [1] = 3; // Select channel 3
w_ChannelArray [2] = 5; // Select channel 5
```

```
b_ReturnValue = b_ADDIDATA_ReadMoreDigitalInputStatus
                (dw_DriverHandle,
                 3,
                 w_ChannelArray,
                 w_ChannelArrayStatus);
```

**Return value:**

1: No error  
 0: Error by calling up the function. Use the function "*i\_ADDIDATA\_GetLastError*", to find the error source.

## 26) b\_ADDIDATA\_Read1DigitalInputValue (...)

### Syntax:

<Return value> = b\_ADDIDATA\_Read1DigitalInputValue  
                                   (DWORD     dw\_DriverHandle,  
                                   WORD       w\_Channel,  
                                   PWORD      pw\_ChannelStatus,  
                                   PDWORD     pdw\_ChannelValue)

### Parameters:

#### - Input:

DWORD *dw\_DriverHandle*           Driver handle  
 WORD   *w\_Channel*                Number of the input to be read

#### - Output:

PWORD *pw\_ChannelStatus*           Status of the digital input  
                                       The number of bits is given by the function  
                                       b\_ADDIDATA\_GetDigitalInputInformationEx.  
 PDWORD *pdw\_ChannelValue*        Value of the digital input: Use the conversion  
                                       function to obtain the value in Volt.

### Task:

Indicates the status of a digital inputs. The variable *w\_Channel* passes the input to be read. A value is returned through the variable *pw\_ChannelStatus*. The value of the inputs is returned by the variable *pdw\_ChannelValue*. To obtain the real voltage value, please use the function *b\_ADDIDATA\_Convert1DigitalInputValueInAnalogValue*.

### Calling convention:

#### ANSI C:

BYTE           b\_ReturnValue;  
 DWORD         dw\_DriverHandle;  
 WORD          w\_Value;  
 DWORD         dw\_Value;

b\_ReturnValue = b\_ADDIDATA\_Read1DigitalInputValue  
                                   (dw\_DriverHandle,  
                                   0,  
                                   &w\_Value,  
                                   &dw\_Value);

### Return value:

1: No error  
 0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError",  
 to find the error source.

**27) b\_ADDIDATA\_ReadMoreDigitalInputValue (...)****Syntax:**

```
<Return value> = b_ADDIDATA_ReadMoreDigitalInputValue
                    (DWORD    dw_DriverHandle,
                     WORD     w_NbrOfChannel,
                     PWORD    pw_ChannelArray,
                     PWORD    pw_ChannelStatusArray,
                     PDWORD   pdw_ChannelValueArray)
```

**Parameters:****- Input:**

DWORD *dw\_DriverHandle*      Driver handle

WORD *w\_NbrOfChannel*      Defines the number of channels in the channel array.

PWORD *pw\_ChannelArray*      Array containing the number of the inputs to be read.

**- Output:**

PWORD *pw\_ChannelStatusArray*      Status of the digital input.  
The number of bit is given by the function *b\_ADDIDATA\_GetDigitalInputInformationEx*.

PDWORD *pdw\_ChannelValueArray*      Value of the digital input: Use the conversion function to obtain the value in Volt.

**Task:**

Reads the analog value of several digital inputs. To obtain the real voltage value, use the function *b\_ADDIDATA\_Convert1DigitalInputValueInAnalogValue*.

**Calling convention:**ANSI C:

```
BYTE    b_ReturnValue;
DWORD   dw_DriverHandle;
WORD    w_Channel[10];
WORD    w_Value[10];
DWORD   dw_Value[10];
```

```
b_ReturnValue = b_ADDIDATA_ReadMoreDigitalInputValue
                (dw_DriverHandle,
                 5,
                 w_Channel,
                 w_Value,
                 dw_Value);
```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "*i\_ADDIDATA\_GetLastError*", to find the error source.

**28) b\_ADDIDATA\_Convert1DigitalInputValueInAnalogValue (...)****Syntax:**

```
<Return value> = b_ADDIDATA_Convert1DigitalInputValueInAnalogValue
                    (DWORD    dw_DriverHandle,
                     WORD     w_Channel,
                     DWORD    dw_DigitalValue,
                     DOUBLE   *pd_AnalogValue)
```

**Parameters:****- Input:**

DWORD <i>dw_DriverHandle</i>	Driver handle
WORD <i>w_Channel</i>	Number of the input to be read
DWORD <i>dw_DigitalValue</i>	Digital value of the input returned by the b_ADDIDATA_Read1DigitalInputValue or b_ADDIDATA_ReadMoreDigitalInputValue.

**- Output:**

DOUBLE <i>*pd_AnalogValue</i>	Value of the input signal in V.
-------------------------------	---------------------------------

**Task:**

Converts the digital input value in a voltage value.

**Calling convention:**ANSI C:

```
BYTE    b_ReturnValue;
DWORD   dw_DriverHandle;
DWORD   dw_Value;
DOUBLE  d_VoltageValue;
```

```
b_ReturnValue = b_ADDIDATA_Convert1DigitalInputValueInAnalogValue
                (dw_DriverHandle,
                 0,
                 dw_Value,
                 &d_VoltageValue);
```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

**29) b\_ADDIDATA\_GetDigitalInputModuleFilterInformation (...)****Syntax:**

```
<Return value> = b_ADDIDATA_GetDigitalInputModuleFilterInformation
                    (DWORD    dw_DriverHandle,
                     DWORD    dw_ModuleNumber,
                     DWORD    dw_StructSize
                     pstr_DigitalInputModuleFilterInf
                     ps_DigitalInputModuleFilterInf)
```

**Parameters:****- Input:**

DWORD *dw\_DriverHandle*                      Driver handle

DWORD *dw\_ModuleNumber*                    Digital input module number.  
The first module begins from 0.

DWORD *dw\_StructSize*                        Size of the digital input module filter  
Information structure. (In bytes)

**- Output:**

pstr\_DigitalInputModuleFilterInf ps\_DigitalInputModuleFilterInf  
The digital input module filter information are  
returned in this structure.

The structure contains :

BYTE *b\_FilterTimeUnit*                      Time units available. See Table 2-2.

WORD *w\_FilterTimeStep*                      Possible time steps for the filter

BYTE *b\_Resolution*                          Selection of the resolution for the filter.  
See Table 2-2.

**Task:**

Returns the time units (*b\_FilterTimeUnit*), the time steps (*w\_FilterTimeStep*) and the resolution (*b\_Resolution*) which can be used for the selected digital input module.

**Table 2-2: Filter timer units**

Value	Description
1	ns
2	µs
3	ns und µs
4	ms
5	ns und ms
6	µs und ms
7	ns und µs und ms
8	s
9	ns und s
10	µs und s
11	ns und ms und s
12	ms und s
13	ns und ms und s
14	µs und ms und s
15	ns und µs und ms und s

**Table 2-3: Resolution**

Value	Resolution
8	8-bit
12	12-bit
16	16-bit
24	24-bit
32	32-bit

**Calling convention:**ANSI C :

BYTE           b\_ReturnValue;

DWORD         dw\_DriverHandle;

str\_DigitalInputModuleFilterInf s\_DigitalInputModuleFilterInf

```
b_ReturnValue = b_ADDIDATA_GetDigitalInputModuleFilterInformation
                (dw_DriverHandle,
                 0,
                 &b_FilterTimeUnit,
                 sizeof (s_DigitalInputModuleFilterInf),
                 &s_DigitalInputModuleFilterInf);
```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

### 30) b\_ADDIDATA\_InitDigitalInputModuleFilter (...)

### Syntax:

```
<Return value> = b_ADDIDATA_InitDigitalInputModuleFilter
                                (DWORD    dw_DriverHandle,
                                DWORD    dw_ModuleNumber,
                                BYTE      b_DelayTimeUnit,
                                DWORD    dw_DelayValue)
```

### Parameters:

**- Input:**

DWORD	<i>dw_DriverHandle</i>	Driver handle
DWORD	<i>dw_ModuleNumber</i>	Digital input module number. The first module begins from 0.
BYTE	<i>b_DelayTimeUnit</i>	Selection of the time unit 0: ns 1: μs 2: ms 3: s
DWORD	<i>dw_DelayValue</i>	Filter time

**- Output:**

No output signal has occurred.

### Task:

Initialises the digital input filter.

### Calling convention:

ANSI C :

BYTE	b_ReturnValue;
DWORD	dw_DriverHandle;

```
b_ReturnValue = b_ADDIDATA_InitDigitalInputModuleFilter
(dw_DriverHandle,
0,
0,
1000);
```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.

### 31) b\_ADDIDATA\_EnableDisableDigitalInputModuleFilter (...)

**Syntax:**

```
<Return value> = b_ADDIDATA_EnableDisableDigitalInputModuleFilter
                                (DWORD   dw_DriverHandle,
                                DWORD   dw_ModuleNumber,
                                BYTE     b_FilterFlag)
```

**Parameters:**
**- Input:**

DWORD	<i>dw_DriverHandle</i>	Driver handle
DWORD	<i>dw_ModuleNumber</i>	Module number.
		The first begins from 0.
BYTE	<i>b_FilterFlag</i>	ADDIDATA_ENABLE or ADDIDATA_DISABLE

**- Ausgabe:**

No output signal has occurred.

**Task:**

Activates or deactivates the filter for the selected digital input module.

**Calling convention:**
ANSI C :

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
```

```
b_ReturnValue = b_ADDIDATA_EnableDisableDigitalInputModuleFilter
                (dw_DriverHandle,
                 0,
                 ADDIDATA_ENABLE);
```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError", to find the error source.



### 32) b\_ADDIDATA\_SetDigitalInputModuleLevelSelection (...)

**Syntax:**

<Return value> = b\_ADDIDATA\_SetDigitalInputModuleLevelSelection  
(DWORD dw\_DriverHandle,  
DWORD dw\_ModuleNumber,  
BYTE b\_LevelFlag)

**Parameters:****- Input:**

DWORD	<i>dw_DriverHandle</i>	Driver handle
DWORD	<i>dw_ModuleNumber</i>	Module number. The first module begins from 0.
BYTE	<i>b_LevelFlag</i>	ADDIDATA_NOT_INVERTED or ADDIDATA_INVERTED

**- Output:**

No output signal has occurred.

**Task:**

Inverts or not the level of the digital inputs of the selected module.

**Calling convention:**ANSI C:

BYTE	b_ReturnValue;
DWORD	dw_DriverHandle;

```
b_ReturnValue = b_ADDIDATA_SetDigitalInputModuleLevelSelection  
                (dw_DriverHandle,  
                 0,  
                 ADDIDATA_INVERTED);
```

**Return value:**

1: No error

0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError",  
to find the error source.

### 33) b\_ADDIDATA\_TestDigitalInputAsynchronousFIFOFull (...)

### Syntax:

<Return value> = b\_ADDIDATA\_TestDigitalInputAsynchronousFIFOFull  
(DWORD dw\_DriverHandle,  
PBYTE pb\_Full)

### Parameters:

**- Input:**

DWORD	<i>dw_DriverHandle</i>	Driver handle
-------	------------------------	---------------

**- Output:**

PBYTE	<i>pb_Full</i>	0: Asynchronous interrupt FIFO memory not full
		1: Asynchronous interrupt FIFO memory is full

### Task:

Tests if the asynchronous interrupt FIFO memory is full or not.  
The asynchronous interrupt FIFO memory is the FIFO memory in which the asynchronous events generated by the asynchronous interrupt are stored.

### Calling convention:

ANSI C :

BYTE	b_ReturnValue;
DWORD	dw_DriverHandle;
BYTE	b_FIFOFull

```
b_ReturnValue = b_ADDIDATA_TestDigitalInputAsynchronousFIFOFull
(dw_DriverHandle,
&b_FIFOFull);
```

### Return value:

1: No error  
0: Error by calling up the function. Use the function "i\_ADDIDATA\_GetLastError",  
to find the error source.