



**ADDI-DATA GmbH
Dieselstraße 3
D-77833 OTTERSWEIER
+49 (0)7223 / 9493-0**

Software description

ADDIDRIVER

Watchdog

Edition: 07.01 – 03/2007

1	INTRODUCTION	3
2	WATCHDOG	4
1)	b_ADDIDATA_GetNumberOfWatchdogs (...)	4
2)	b_ADDIDATA_GetWatchdogInformation (...)	5
3)	b_ADDIDATA_GetWatchdogInformationEx (...)	7
4)	b_ADDIDATA_InitWatchdog (...)	9
5)	b_ADDIDATA_EnableDisableWatchdogInterrupt (...)	10
6)	b_ADDIDATA_StartWatchdog (...)	11
7)	b_ADDIDATA_StartAllWatchdogs (...)	12
8)	b_ADDIDATA_TriggerWatchdog (...)	13
9)	b_ADDIDATA_TriggerAllWatchdogs (...)	14
10)	b_ADDIDATA_StopWatchdog (...)	15
11)	b_ADDIDATA_StopAllWatchdogs (...)	16
12)	b_ADDIDATA_ReleaseWatchdog (...)	17
13)	b_ADDIDATA_ReadWatchdogStatus (...)	18
14)	b_ADDIDATA_EnableDisableWatchdogHardwareGate (...)	19
15)	b_ADDIDATA_GetWatchdogHardwareGateStatus (...)	20
16)	b_ADDIDATA_EnableDisableWatchdogHardwareTrigger (...)	21
17)	b_ADDIDATA_GetWatchdogHardwareTriggerStatus (...)	22
18)	b_ADDIDATA_GetWarningDelayInformation (...)	23
19)	b_ADDIDATA_InitWarningDelay (...)	24
20)	b_ADDIDATA_EnableDisableWatchdogWarningRelay (...)	25
21)	b_ADDIDATA_EnableDisableWatchdogResetRelay (...)	26
22)	b_ADDIDATA_SetWatchdogResetRelayMode (...)	27
23)	b_ADDIDATA_EnableDisableWatchdogOutput (...)	28
24)	b_ADDIDATA_GetWatchdogHardwareOutputStatus (...)	29
25)	b_ADDIDATA_TestWatchdogAsynchronousFIFOFull	30

Tables

Table 2-2. Watchdog timer units

Table 2-3. Resolution 6

1 INTRODUCTION

i

IMPORTANT!

Note the following conventions in the text:

Function: "b_ADDIDATA_GetNumberOfAnalogInputs"

Variable *dw_DriverHandle*

Table 0-1: Type Declaration for Windows 98/NT/2000/XP

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

2 WATCHDOG

1) b_ADDIDATA_GetNumberOfWatchdogs (...)

Syntax:

```
<Return value> = b_ADDIDATA_GetNumberOfWatchdogs
                                     (DWORD   dw_DriverHandle,
                                     BYTE     pb_WatchdogNumber,
                                     BYTE     pb_WatchdogType)
```

Parameters:

- Input:

DWORD *dw_DriverHandle* Handle of the ADDI-DATA driver

- Output:

BYTE *pb_WatchdogNumber* Number of watchdogs

BYTE *pb_WatchdogType* Returns the type of each watchdog. See table 2-1
 pb_WatchdogType [0]: Type of the first watchdog
 pb_WatchdogType [1]: Type of the 2nd watchdog
 ...
 pb_WatchdogType [*pb_WatchdogNumber - 1]:
 Type of the last watchdog

Task:

Returns the number of watchdogs and the type of each watchdog.

Table 0-1: Watchdog type

Constant	Value	Description
ADDIDATA_DIGITAL_OUTPUT_WATCHDOG	1	Watchdog for the digital outputs After running-down of the watchdog, the outputs are reset.
ADDIDATA_ANALOG_OUTPUT_WATCHDOG	2	Watchdog for the analogue outputs After running-down of the watchdog, the outputs are reset.
ADDIDATA_ANA_DIG_OUTPUT_WATCHDOG	3	Watchdog for the digital/analogue outputs. After running-down of the watchdog, the digital/analogue outputs are reset.
ADDIDATA_SYSTEM_WATCHDOG	4	System watchdog. This watchdog can be used for the system control.

Calling convention:

ANSI C:

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
BYTE      b_WatchdogNumber;
BYTE      b_WatchdogType [12];
```

```
b_ReturnValue = b_ADDIDATA_GetNumberOfWatchdog
                                     (dw_DriverHandle,
                                     &b_WatchdogNumber,
                                     b_WatchdogType);
```

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_GetWatchdogInformation (...)

Syntax:

<Return value> = b_ADDIDATA_GetWatchdogInformation
 (DOWRD dw_DriverHandle,
 BYTE b_WatchdogNumber,
 PBYTE pb_WatchdogTimeUnit,
 PBYTE pb_WatchdogTimeStep,
 PBYTE pb_Resolution,
 PDWORD pdw_HardwareGateAvailable,
 PDWORD pdw_HardwareTriggerAvailable,
 PDWORD pdw_WarningRelayAvailable,
 PDWORD pdw_ResetRelayAvailable,
 PDWORD pdw_WarningDelayAvailable)

Parameters:

- Input:

DWORD *dw_DriverHandle* Handle of the ADDI-DATA driver
 BYTE *b_WatchdogNumber* Watchdog number.
 The first watchdog begins from 0.

- Output:

PBYTE *pb_TimerTimeUnit* Time units available. See table 2-2.
 PWORD *pw_WatchdogTimeStep* Possible time steps for the watchdog
 PBYTE *pb_Resolution* Selection of the resolution for the watchdog.
 See table 2-3.

PDWORD *pdw_HardwareGateAvailable*
 0: Hardware gate not available.
 1: Hardware gate available.

PDWORD *pdw_HardwareTriggerAvailable*
 0: Hardware trigger not available.
 1: Hardware trigger available.

PDWORD *pdw_WarningRelayAvailable*
 0: Warning relay not available
 1: Warning relay available

PDWORD *pdw_ResetRelayAvailable*
 0: Reset relay not available
 1: Reset relay available

PDWORD *pdw_WarningDelayAvailable*
 0: Waiting time between the warning relay
 and the activation of the reset relay is not available
 1: Waiting time between the warning relay
 and the activation of the reset relay is available.

Task:

Returns the time units (*pb_WatchdogTimeUnit*), the time steps (*pw_WatchdogTimeStep*) and the resolution (*pb_Resolution*) which can be used for the selected timer (*b_TimerNumber*).

Table 2-2. Watchdog 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;
BYTE      b_WatchdogTimeUnit;
BYTE      b_WatchdogTimeStep;
BYTE      b_Resolution;
DWORD     dw_HardwareGateAvailable;
DWORD     dw_HardwareTriggerAvailable;
DWORD     dw_WarningRelayAvailable;
DWORD     dw_ResetRelayAvailable;
DWORD     dw_WarningDelayAvailable;

```

```

b_ReturnValue = b_ADDIDATA_GetWatchdogInformation
                (dw_DriverHandle,
                 0,
                 &b_WatchdogTimeUnit,
                 &b_WatchdogTimeStep,
                 &b_Resolution,
                 &dw_HardwareGateAvailable,
                 &dw_HardwareTriggerAvailable,
                 &dw_WarningRelayAvailable,
                 &dw_ResetRelayAvailable,
                 &dw_WarningDelayAvailable);

```

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_GetWatchdogInformationEx (...)

Syntax:

```
<Return value> = b_ADDIDATA_GetWatchdogInformationEx
                    (DWORD    dw_DriverHandle,
                     BYTE      b_WatchdogNumber,
                     pstr_GetWatchdogInformation ps_WatchdogInformation,
                     DWORD     dw_StructSize)
```

Parameters:

- Input:

DWORD *dw_DriverHandle* Handle of the ADDI-DATA driver
 BYTE *b_WatchdogNumber* Watchdog number.
 The first watchdog begins from 0.
 DWORD *dw_StructSize* Size of the structure.

- Output:

pstr_GetWatchdogInformation ps_WatchdogInformation Watchdog information.

Returned structure:

```
typedef struct
```

```
{
  BYTE  b_WatchdogTimeUnit; Available time unit
  BYTE  b_Resolution;       Watchdog resolution. See table 2-3
  BYTE  b_InterruptAvailable; FALSE: Interrupt not available
                               TRUE: Interrupt available
  BYTE  b_WatchdogType;      Type of watchdog. See table 2-1
  WORD  w_WatchdogTimeStep;  Time step of the watchdog
  BYTE  b_Reserved1;         reserved for alignment
  DWORD dw_HardwareGateAvailable;
                               FALSE: Hardware gate not available
                               TRUE: Hardware gate available
  DWORD dw_HardwareGateHighAvailable;
                               FALSE: Hardware gate high not available
                               TRUE: Hardware gate high available
  DWORD dw_HardwareGateLowAvailable;
                               FALSE: Hardware gate low not available
                               TRUE: Hardware gate low available
  DWORD dw_HardwareTriggerAvailable;
                               FALSE: Hardware trigger not available
                               TRUE: Hardware trigger available
  DWORD dw_HardwareTriggerHighAvailable;
                               FALSE: Hardware trigger high not available
                               TRUE: Hardware trigger high available
  DWORD dw_HardwareTriggerLowAvailable;
                               FALSE: Hardware trigger low not available
                               TRUE: Hardware trigger low available
  DWORD dw_WarningRelayAvailable;
                               FALSE: Warning relay not available
                               TRUE: Warning relay available
  DWORD dw_ResetRelayAvailable;
                               FALSE: Reset relay not available
                               TRUE: Reset relay available
```

```

BYTE    b_ResetRelayModeConfigurable;
        0: Reset mode cannot be configured
        1: Reset mode can be configured

BYTE    b_Reserved2;
BYTE    b_Reserved3;
DWORD   dw_WarningDelayAvailable;
        FALSE: Warning for reset delay not available
        TRUE: Warning for reset delay available

DWORD   dw_HardwareOutputAvailable;
        FALSE: Output not available
        TRUE: Output available

DWORD   dw_HardwareOutputHighAvailable;
        FALSE: Output high not available
        TRUE: Output high available

DWORD   dw_HardwareOutputLowAvailable;
        FALSE: Output Low not available
        TRUE: Output Low available

WORD    w_Reserved4;

}str_GetWatchdogInformation,*pstr_GetWatchdogInformation;

```

Task:

Returns the time units (*pb_WatchdogTimeUnit*), the time steps (*pw_WatchdogTimeStep*) and the resolution (*pb_Resolution*) which can be used for the selected watchdog (*b_WatchdogNumber*).

Calling convention:ANSI C:

```

BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;
str_GetWatchdogInformation s_WatchdogInformation;

b_ReturnValue = b_ADDIDATA_GetWatchdogInformation
                (dw_DriverHandle,
                 0,
                 &s_WatchdogInformation,
                 sizeof (str_GetWatchdogInformation));

```

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_InitWatchdog (...)

Syntax:

```
<Return value> = b_ADDIDATA_InitWatchdog
                                (DWORD    dw_DriverHandle,
                                BYTE      b_WatchdogNumber,
                                BYTE      b_WatchdogTimeUnit,
                                DWORD     dw_DelayValue)
```

Parameters:

- Input:

DWORD	<i>dw_DriverHandle</i>	Handle of the ADDI-DATA driver
BYTE	<i>b_WatchdogNumber</i>	Watchdog number. The first watchdog begins from 0.
BYTE	<i>b_DelayTimeUnit</i>	Selection of the time unit 0: ns 1: μ s 2: ms 3: s 4: min
DWORD	<i>dw_DelayValue</i>	Start value or watchdog time (depends from the used ADDI-DATA board) see function "b_ADDIDATA_GetWatchdogInformation"

- Ausgabe:

No output signal has occurred.

Task:

Initialises the watchdog.

Calling convention:

ANSI C:

```
BYTE      b_ReturnValue;
DWORD     dw_DriverHandle;

b_ReturnValue = b_ADDIDATA_InitWatchdog
                                (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.

5) b_ADDIDATA_EnableDisableWatchdogInterrupt (...)

Syntax:

```
<Return value> = b_ADDIDATA_EnableDisableWatchdogInterrupt
                                     (DWORD    dw_DriverHandle,
                                     BYTE      b_WatchdogNumber,
                                     BYTE      b_InterruptFlag)
```

Parameters:

- Input:

DWORD	<i>dw_DriverHandle</i>	Handle of the ADDI-DATA driver
BYTE	<i>b_WatchdogNumber</i>	Watchdog number. The first watchdog begins from 0.
BYTE	<i>b_InterruptFlag</i>	ADDIDATA_ENABLE or ADDIDATA_DISABLE

- Ausgabe:

No output signal has occurred.

Task:

Activates or deactivates the IRQ for the watchdog process

Calling convention:

ANSI C:

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

```
b_ReturnValue = b_ADDIDATA_EnableDisableWatchdogInterrupt
                (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.

i

IMPORTANT!

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

6) b_ADDIDATA_StartWatchdog (...)

Syntax:

<Return value> = b_ADDIDATA_StartWatchdog (DWORD dw_DriverHandle,
BYTE b_WatchdogNumber)

Parameters:**- Input:**

DWORD *dw_DriverHandle* Handle of the ADDI-DATA driver
BYTE *b_WatchdogNumber* Watchdog number.
The first watchdog begins from 0.

- Output:

No output signal has occurred.

Task:

Starts the watchdog.

Calling convention:ANSI C:

BYTE b_ReturnValue;
DWORD dw_DriverHandle;

b_ReturnValue = b_ADDIDATA_StartWatchdog (dw_DriverHandle, 0);

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_StartAllWatchdogs (...)

Syntax:

<Return value> = b_ADDIDATA_StartAllWatchdogs (DWORD dw_DriverHandle)

Parameters:**- Input:**

DWORD *dw_DriverHandle* Handle of the ADDI-DATA driver

- Output:

No output signal has occurred.

Task:

Start all watchdogs of all boards.

Calling convention:ANSI C:

BYTE b_ReturnValue;
DWORD dw_DriverHandle;

b_ReturnValue = b_ADDIDATA_StartAllWatchdogs (dw_DriverHandle);

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_TriggerWatchdog (...)

Syntax:

<Return value> = b_ADDIDATA_TriggerWatchdog (DWORD dw_DriverHandle,
BYTE b_WatchdogNumber)

Parameters:**- Input:**

DWORD *dw_DriverHandle* Handle of the ADDI-DATA driver
BYTE *b_WatchdogNumber* Watchdog number.
The first watchdog begins from 0.

- Output:

A trigger has occurred on the watchdog

Task:

Trigger the watchdog.

Calling convention:ANSI C:

BYTE b_ReturnValue;
DWORD dw_DriverHandle;

b_ReturnValue = b_ADDIDATA_TriggerWatchdog (dw_DriverHandle, 0);

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_TriggerAllWatchdogs (...)

Syntax:

<Return value> = b_ADDIDATA_TriggerAllWatchdogs
(DWORD dw_DriverHandle)

Parameters:

- Input:

DWORD *dw_DriverHandle* Handle of the ADDI-DATA driver

- Output:

A trigger has occurred on the watchdog.

Task:

Triggert all watchdogs on all boards.

Calling convention:

ANSI C:

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

```
b_ReturnValue = b_ADDIDATA_TriggerAllWatchdogs      (dw_DriverHandle);
```

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_StopWatchdog (...)**Syntax:**

<Return value> = b_ADDIDATA_StopWatchdog (DWORD dw_DriverHandle,
 BYTE b_WatchdogNumber)

Parameters:**- Input:**

DWORD *dw_DriverHandle* Handle of the ADDI-DATA driver
 BYTE *b_WatchdogNumber* Watchdog number.
 The first watchdog begins from 0.

- Output:

No output signal has occurred.

Task:

Stops the watchdog.

Calling convention:ANSI C:

BYTE b_ReturnValue;
 DWORD dw_DriverHandle;

b_ReturnValue = b_ADDIDATA_StopWatchdog (dw_DriverHandle, 0);

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_StopAllWatchdogs (...)

Syntax:

<Return value> = b_ADDIDATA_StopAllWatchdogs
(DWORD dw_DriverHandle)

Parameters:

- Input:

DWORD *dw_DriverHandle* Handle of the ADDI-DATA driver

- Output:

No output signal has occurred.

Task:

Stops all watchdogs on all boards.

Calling convention:

ANSI C:

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

```
b_ReturnValue = b_ADDIDATA_StopAllWatchdogs (dw_DriverHandle);
```

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_ReleaseWatchdog (...)**Syntax:**

<Return value> = b_ADDIDATA_ReleaseWatchdog (DWORD dw_DriverHandle,
BYTE b_WatchdogNumber)

Parameters:**- Input:**

DWORD *dw_DriverHandle* Handle of the ADDI-DATA driver
BYTE *b_WatchdogNumber* Watchdog number.
The first watchdog begins from 0.

- Output:

No output signal has occurred.

Task:

Releases the watchdog for a new initialisation.

Calling convention:ANSI C:

BYTE b_ReturnValue;
DWORD dw_DriverHandle;

b_ReturnValue = b_ADDIDATA_ReleaseWatchdog (dw_DriverHandle, 0);

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_ReadWatchdogStatus (...)

Syntax:

```
<Return Value> = b_ADDIDATA_ReadWatchdogStatus
                                (DWORD dw_DriverHandle,
                                 BYTE b_WatchdogNumber,
                                 PBYTE pb_WatchdogStatus,
                                 PBYTE pb_SoftwareTriggerStatus,
                                 PBYTE pb_HardwareTriggerStatus)
```

Parameters:
- Input:

DWORD *dw_DriverHandle* Handle of the ADDI-DATA driver
 BYTE *b_WatchdogNumber* Watchdog number.
 The first watchdog begins from 0.

- Output:

PBYTE *pb_WatchdogStatus* 0: Watchdog has run down or did not start
 1: Watchdog is running
 PBYTE *pb_SoftwareTriggerStatus*
 0: Software trigger did not occur
 1: Software trigger occurred
 When the status of the software trigger is read, it is automatically reset. By the next calling of the parameter, 0 is returned if no trigger occurred during this period.
 PBYTE *pb_HardwareTriggerStatus*
 0: Hardware trigger did not occur
 1: Hardware trigger occurred

Task:

Returns the status of the watchdog, the software trigger and the hardware trigger.

Calling convention:
ANSI C:

```
BYTE b_ReturnValue;
DWORD dw_DriverHandle;
BYTE b_WatchdogStatus;
BYTE b_SoftwareTriggerStatus;
BYTE b_HardwareTriggerStatus;
```

```
b_ReturnValue = b_ADDIDATA_ReadWatchdogStatus
                (dw_DriverHandle, 0, &b_WatchdogStatus,
                 &b_SoftwareTriggerStatus, &b_HardwareTriggerStatus);
```

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_EnableDisableWatchdogHardwareGate (...)

Syntax:

```
<Return Value> = b_ADDIDATA_EnableDisableWatchdogHardwareGate  
                (DWORD dw_DriverHandle,  
                 BYTE    b_WatchdogNumber,  
                 BYTE    b_HardwareGateFlag,  
                 BYTE    b_HardwareGateLevel)
```

Parameters:

- Input:

DWORD	<i>dw_DriverHandle</i>	Handle of the ADDI-DATA driver
BYTE	<i>b_WatchdogNumber</i>	Watchdog number. The first watchdog begins from 0.
BYTE	<i>b_ExternGateFlag</i>	ADDIDATA_ENABLE: enables the hardware gate. ADDIDATA_DISABLE: Hardware gate disabled by starting the watchdog
BYTE	<i>b_HardwareGateLevel</i>	ADDIDATA_LOW: If the hardware gate is enabled, it is active at "0" ADDIDATA_HIGH: If the hardware gate is enabled, it is active at "1"

- Output:

No output signal has occurred.

Task:

Releases or blocks the action of the hardware gate.

Calling convention:

ANSI C:

BYTE	b_ReturnValue;
DWORD	dw_DriverHandle;

```
b_ReturnValue = b_ADDIDATA_EnableDisableWatchdogHardwareGate
(dw_DriverHandle,
0,
ADDIDATA_ENABLE,
ADDIDATA_HIGH);
```

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_GetWatchdogHardwareGateStatus (...)**Syntax:**

```
<Return Value> = b_ADDIDATA_GateWatchdogHardwareGateStatus
                    (DWORD dw_DriverHandle,
                     BYTE    b_WatchdogNumber,
                     PBYTE   pb_HardwareGateStatus)
```

Parameters:**- Input:**

DWORD *dw_DriverHandle* Handle of the ADDI-DATA driver
 BYTE *b_WatchdogNumber* Watchdog number.
 The first watchdog begins from 0.

- Output:

PBYTE *pb_HardwareGateStatus*
 0: Hardware gate is not active (low status)
 1: Hardware gate is active (high status)

Task:

Returns the status of the watchdog hardware gate (active or not).

Calling convention:ANSI C:

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

```
b_ReturnValue = b_ADDIDATA_GateWatchdogHardwareGateStatus
                (dw_DriverHandle,
                 0,
                 &b_HardwareGateStatus);
```

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_EnableDisableWatchdogHardwareTrigger (...)

Syntax:

<Return Value> = b_ADDIDATA_EnableDisableWatchdogHardwareTrigger
 (DWORD dw_DriverHandle,
 BYTE b_WatchdogNumber,
 BYTE b_HardwareTriggerFlag,
 BYTE b_HardwareTriggerLevel)

Parameters:

- Input:

DWORD *dw_DriverHandle* Handle of the ADDI-DATA driver
 BYTE *b_WatchdogNumber* Watchdog number.
 The first watchdog begins from 0.
 BYTE *b_HardwareTriggerFlag*
 ADDIDATA_ENABLE: Enables the hardware trigger.
 ADDIDATA_DISABLE: Hardware trigger disabled by triggering the watchdog
 BYTE *b_HardwareTriggerLevel*
 ADDIDATA_LOW: If the hardware trigger is used, it triggers from "1" to "0"
 ADDIDATA_HIGH: If the hardware trigger is used, it triggers from "0" to "1"
 ADDIDATA_HIGH_LOW: If the hardware trigger is used, it triggers from "0" to "1" and from "1" to "0"

- Output:

No output signal has occurred.

Task:

Releases or blocks the action of the hardware trigger.

Calling convention:

ANSI C:

BYTE b_ReturnValue;
 DWORD dw_DriverHandle;

b_ReturnValue = b_ADDIDATA_EnableDisableWatchdogHardwareTrigger
 (dw_DriverHandle,
 0,
 ADDIDATA_ENABLE,
 ADDIDATA_HIGH);

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_GetWatchdogHardwareTriggerStatus (...)

Syntax:

```
<Return Value> = b_ADDIDATA_GateWatchdogHardwareTriggerStatus
                                     (DWORD   dw_DriverHandle,
                                     BYTE     b_WatchdogNumber,
                                     PBYTE    pb_HardwareTriggerStatus)
```

Parameters:

- Input:

DWORD *dw_DriverHandle* Handle of the ADDI-DATA driver
 BYTE *b_WatchdogNumber* Watchdog number.
 The first watchdog begins from 0.

- Output:

PBYTE *pb_HardwareTriggerStatus*
 0: Hardware trigger is not active (low status)
 1: Hardware trigger is active (high status)

Task:

Returns the status of the watchdog hardware trigger (active or not).

Calling convention:

ANSI C:

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

```
b_ReturnValue = b_ADDIDATA_GateWatchdogHardwareTriggerStatus
                                     (dw_DriverHandle,
                                     0,
                                     &b_HardwareTriggerStatus);
```

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_GetWarningDelayInformation (...)**Syntax:**

<Return value> = b_ADDIDATA_GetWarningDelayInformation
 (DWORD dw_DriverHandle,
 BYTE b_WatchdogNumber,
 PBYTE pb_WarningDelayTimeUnit,
 PBYTE pb_WarningDelayTimeStep,
 PBYTE pb_WarningDelayResolution)

Parameters:**- Input:**

DWORD *dw_DriverHandle* Handle of the ADDI-DATA driver
 BYTE *b_WatchdogNumber* Watchdog number.
 The first watchdog begins from 0.

- Output:

PBYTE *pb_WarningDelayTimeUnit*
 Possible time units for the waiting time between the warning relay and the activation of the reset relay. See table 2-2.

PBYTE *pb_WarningDelayTimeStep*
 Timer steps for the waiting time between the warning relay and the activation of the reset relay.

PBYTE *pb_WarningDelayResolution*
 Resolution of the waiting time between the warning relay and the activation of the reset relay.

Task:

Returns the time units (*pb_WarningDelayTimeUnit*) and the time steps (*pb_WarningDelayTimeStep*) which can be set for the waiting time between the warning relay and the activation of the reset relay for the selected watchdog (*b_WatchdogNumber*)

Calling convention:ANSI C:

BYTE b_ReturnValue;
 DWORD dw_DriverHandle;
 BYTE b_WarningDelayTimeUnit;
 BYTE b_WarningDelayTimeStep;
 BYTE b_WarningDelayResolution;

b_ReturnValue = b_ADDIDATA_GetWarningDelayInformation
 (dw_DriverHandle,
 0,
 &b_WarningDelayTimeUnit,
 &b_WarningDelayTimeStep,
 &b_WarningDelayResolution);

Return value:

1: No error
 0: Error by calling up the function. Use the function
 "i_ADDIDATA_GetLastError", to find the error source.

19) b_ADDIDATA_InitWarningDelay (...)

Syntax:

<Return value> = b_ADDIDATA_InitWarningDelay

```
(DWORD    dw_DriverHandle,
BYTE      b_WatchdogNumber,
BYTE      b_DelayTimeUnit,
DWORD     dw_DelayValue)
```

Parameters:

- Input:

DWORD	<i>dw_DriverHandle</i>	Handle of the ADDI-DATA driver
BYTE	<i>b_WatchdogNumber</i>	Watchdog number. The first watchdog begins from 0.
BYTE	<i>b_DelayTimeUnit</i>	Time unit 0: ns 1: μ s 2: ms 3: s
DWORD	<i>dw_DelayValue</i>	Start value or watchdog time between the warning relay and the activation of the reset delay (depends from the used ADDI-DATA board) see function "b_ADDIDATA_GetWatchdogInformation"

- Ausgabe:

No output signal has occurred.

Task:

Initialises the waiting timer between the warning relay and the activation of the reset relay.

Calling convention:

ANSI C:

BYTE	b_ReturnValue;
DWORD	dw_DriverHandle;

```
b_ReturnValue = b_ADDIDATA_InitWarningDelay
                (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.

20) b_ADDIDATA_EnableDisableWatchdogWarningRelay (...)

Syntax:

```
<Return Value> = b_ADDIDATA_EnableDisableWatchdogWarningRelay  
                (DWORD dw_DriverHandle,  
                 BYTE    b_WatchdogNumber,  
                 BYTE    b_WarningFlag)
```

Parameters:

- Input:

DWORD	<i>dw_DriverHandle</i>	Handle of the ADDI-DATA driver
BYTE	<i>b_WatchdogNumber</i>	Watchdog number. The first watchdog begins from 0.
BYTE	<i>b_WarningFlag</i>	ADDIDATA_ENABLE: Enables the switching on of the warning relay after running down of the watchdog ADDIDATA_DISABLE: The warning relay is not activated after running down of the watchdog

- Output:

No output signal has occurred.

Task:

Releases or blocks the activation of the warning relay after running down of the watchdog.

Calling convention:

ANSI C:

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

```
b_ReturnValue = b_ADDIDATA_EnableDisableWatchdogWarningRelay
(dw_DriverHandle,
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.

21) b_ADDIDATA_EnableDisableWatchdogResetRelay (...)

Syntax:

```
<Return Value> = b_ADDIDATA_EnableDisableWatchdogResetRelay
                                     (DWORD dw_DriverHandle,
                                     BYTE    b_WatchdogNumber,
                                     BYTE    b_ResetFlag)
```

Parameters:

- Input:

DWORD	<i>dw_DriverHandle</i>	Handle of the ADDI-DATA driver
BYTE	<i>b_WatchdogNumber</i>	Watchdog number. The first watchdog begins from 0.
BYTE	<i>b_WarningFlag</i>	ADDIDATA_ENABLE: Enables the switching on of the reset relay after running down of the watchdog ADDIDATA_DISABLE: The reset relay is not activated after running down of the watchdog

- Output:

No output signal has occurred.

Task:

Releases or blocks the activation of the reset relay after running down of the watchdog.

Calling convention:

ANSI C:

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

```
b_ReturnValue = b_ADDIDATA_EnableDisableWatchdogResetRelay
                                     (dw_DriverHandle,
                                     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_SetWatchdogResetRelayMode (...)**Syntax:**

```
<Return Value> = b_ADDIDATA_SetWatchdogResetRelayMode
                                (DWORD    dw_DriverHandle,
                                 BYTE      b_WatchdogNumber,
                                 BYTE      b_ResetRelayMode)
```

Parameters:**- Input:**

DWORD	<i>dw_DriverHandle</i>	Handle of the ADDI-DATA driver
BYTE	<i>b_WatchdogNumber</i>	Watchdog number. The first watchdog begins from 0.
BYTE	<i>b_ResetRelayMode</i>	ADDIDATA_ENABLE: Reset relay is set when watchdog has run down. ADDIDATA_DISABLE: Reset relay is reset when watchdog has run down.

- Output:

No output signal has occurred.

Task:

Sets the mode of the reset relay during and after the running down of the watchdog.

Calling convention:ANSI C:

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

```
b_ReturnValue = b_ADDIDATA_EnableDisableWatchdogResetRelay
                                (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.

23) b_ADDIDATA_EnableDisableWatchdogOutput (...)**Syntax:**

```
<Return value> = b_ADDIDATA_EnableDisableWatchdogOutput
                    (DWORD dw_DriverHandle,
                     BYTE   b_WatchdogNumber,
                     BYTE   b_OutputFlag,
                     BYTE   b_OutputLevel)
```

Parameters:**- Input:**

DWORD	dw_DriverHandle	Handle of the ADDI-DATA driver
BYTE	b_WatchdogNumber	Number of the watchdog The first watchdog begins from 0.
BYTE	b_OutputFlag	ADDIDATA_ENABLE or ADDIDATA_DISABLE
BYTE	b_OutputInverted	ADDIDATA_DISABLE: the watchdog output is not inverted. ADDIDATA_ENABLE: The watchdog output is inverted.

- Output:

No output signal has occurred.

Task:

Activates or deactivates the watchdog output.

Calling convention:ANSI C :

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

```
b_ReturnValue = b_ADDIDATA_EnableDisableWatchdogOutput
                (dw_DriverHandle,
                 0,
                 ADDIDATA_ENABLE,
                 ADDIDATA_DISABLE);
```

Return value:

1: No error

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

"i_ADDIDATA_GetLastError", to find the error source.

24) b_ADDIDATA_GetWatchdogHardwareOutputStatus (...)**Syntax:**

```
<Return Value> = b_ADDIDATA_GateWatchdogHardwareOutputStatus
                    (DWORD dw_DriverHandle,
                     BYTE b_WatchdogNumber,
                     PBYTE pb_HardwareOutputStatus)
```

Parameters:**- Input:**

DWORD dw_DriverHandle Handle of the ADDI-DATA driver
 BYTE b_WatchdogNumber Number of the watchdog
 The first watchdog begins from 0.

- Output:

PBYTE pb_HardwareOutputStatus
 0: Hardware output is not active (low status)
 1: Hardware output is active (high status)

Task:

Returns the status of the watchdog hardware output (active or not).

Calling convention:ANSI C:

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

```
b_ReturnValue = b_ADDIDATA_GateWatchdogHardwareOutputStatus
                (dw_DriverHandle,
                 0,
                 &b_HardwareOutputStatus);
```

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_TestWatchdogAsynchronousFIFOFull

Syntax:

```
<Return value> = b_ADDIDATA_TestWatchdogAsynchronousFIFOFull
                                     (DWORD    dw_DriverHandle,
                                     PBYTE     pb_Full)
```

Parameters:

- Input:

DWORD *dw_DriverHandle* Driver handle

- Output:

PBYTE *pb_Full* 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_TestWatchdogAsynchronousFIFOFull
                                     (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.