Watchdog Timer

UNO-2170, UNO-2171, UNO-205XE

User Manual
Copyright
The documentation and the software included with this product are copyrighted 2006 by Advantech Co., Ltd. All rights are reserved. Advantech Co., Ltd. reserves the right to make improvements in the products described in this manual at any time without notice. No part of this manual may be reproduced, copied, translated or transmitted in any form or by any means without the prior written permission of Advantech Co., Ltd. Information provided in this manual is intended to be accurate and reliable. However, Advantech Co., Ltd. assumes no responsibility for its use, nor for any infringements of the rights of third parties, which may result from its use.

Acknowledgements
Intel and Pentium are trademarks of Intel Corporation.
Microsoft Windows and MS-DOS are registered trademarks of Microsoft Corp.
All other product names or trademarks are properties of their respective owners.
Declaration of Conformity

CE

This product has passed the CE test for environmental specifications when shielded cables are used for external wiring. We recommend the use of shielded cables. This kind of cable is available from Advantech. Please contact your local supplier for ordering information.

FCC Class A

Note: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Technical Support and Assistance

Step 1. Visit the Advantech web site at www.advantech.com/support where you can find the latest information about the product. Contact your distributor, sales representative, or Advantech's customer service center for technical support if you need additional assistance. Please have the following information ready before you call:
- Product name and serial number
- Description of your peripheral attachments
- Description of your software (OS, version, application software, etc.)
- A complete description of the problem
- The exact wording of any error messages
Contents

Chapter 1 Installation .......................................................... 2

Chapter 2 Configuration................................................... 8
  2.1 User Interface .................................................................. 8
    2.1.1 General Tag .......................................................... 9
    2.1.2 Setting Tag ........................................................... 11
    2.1.3 Event Log Database Operations ............................... 12
    2.1.4 Abort Tag .............................................................. 16

Chapter 3 Function API’s.................................................. 18
  3.1 Error Codes Returned by Functions ............................... 18
  3.2 Data Structures ........................................................... 18
    3.2.1 Watchdog Driver Interfaces .................................... 19

Chapter 4 Example Call Flows.......................................... 28
  4.1 ElapsedTme ................................................................. 28
  4.2 RebootMachine .......................................................... 29
  4.3 Setlog ......................................................................... 30
  4.4 Setmode ....................................................................... 31
  4.5 TimerSpan .................................................................. 32
  4.6 WatchApplication....................................................... 33
  4.7 WatchSystem.............................................................. 35
Installation

This chapter introduces how to install UNO-2170, UNO-2171 and UNO-205XE Watchdog timer driver under Windows 2K/XP platform.
Chapter 1 Installation

<Step1> Insert the UNO-2000 Driver and Utility CD in the CD-ROM, and execute AdvWDT.exe from following path: “\UNO-2170, UNO-2171 and UNO-205XE\Driver\Watchdog Timer\Win2K_XP\”. Click the Next button to install watchdog timer.
<Step2> Enter your name and company, and press Next.

<Step3> For UNO-2170, UNO-2171 and UNO-205XE, please select “Advantech [W83627HF] WDT” and then press the Next button for next step.
<Step 4> Select “Typical”, the watchdog timer driver will be installed with the most common options. Recommended for most users.

<Step 5> Press Next to start the installation.
<Step 6> Press the Finish button to finish the watchdog timer installation. If UNO-2170, UNO-2171 and UNO-205XE watchdog timer is installed successfully, the relevant applications and samples will be placed in following path: \Program Files\ADVANTECH\Watchdog\n
<Step 7> Press OK to reboot the system.
Configuration

This chapter introduces how to use and configure the watchdog timer function. You can execute the Watchdog Timer Configuration in “Control Panel”.

Chapter 2  Configuration

2.1  User Interface

There are three tags in Advantech Watchdog Service Configuration, including General, Setting and About.

**General Tag:** Display general information on the watchdog service

**Setting Tag:** Display setting information on the watchdog service

**About Tag:** Display the copyright information of the watchdog service
2.1.1 General Tag
There are five items mentioned in General tag,

**Service Name:** Display the name of the Advantech watchdog service in the Service Control Manager(SCM) database

**Watchdog Type:** Display the type of the watchdog chipset type. If you are using UNO-2170, UNO-2171 or UNO-205XE, it will display “Winbond W83627HF

**Running Status:** display the watchdog current status: Enabled or Disabled

**Elapsed Time:** The elapsed time from when the watchdog is enabled. If the watchdog is disabled, elapsed time will be 00 hour 00 minute 00 second.

**Start Watchdog Serviced on Boot:** If the this check box is selected and the settings is applied by the “Apply” button on the main dialog then the Advantech watchdog service will be started after the system boots, otherwise the use should start the Advantech service manually.
If the Advantech watchdog service is running, then the caption of this button will be “Stop Service”, if the user left clicks the button then the Advantech watchdog service will be stopped, if this operation succeeds then the caption of this button will be changed to “Start Service”, otherwise the caption will not change a piece of warring message will popup to notify the user that the service can not be stopped now.
2.1.2 Setting Tag

The setting tag includes the following items.

(1) **“Timer Span” Combo Box**: The user can select one timer span for the watchdog and apply the change when the watchdog is disabled.

(2) **“Watch Mode” Group**: There are two watch modes: system watch mode and application watch mode. If you are in Application mode, the system will warn you to “strobe” (reset) the watchdog timer, otherwise the system will reboot in a specified time period.

**Definitions:**

**System mode**: watchdog timer is running in Windows background. If the hardware is hung up, the system will reboot automatically.

**Application mode**: watchdog timer will be enabling when you call the APIs within your application. Further information about the watchdog timer APIs, please refer to Chapter 3.

(3) **“Log Event”**: If this check box is selected and press “Apply” button to apply the setting. Then the “Enabled”, “Disable”, “Reboot” operation of the watchdog will be logged into the system event base, otherwise the three operations will not be logged into the system event base.
(4) **“Enable/Disable” Button:** Enable or disable the watchdog. If the watchdog is enabled user cannot change the watch mode and the timer span, so these related controls become grayed. These controls resume to theirs normal status when the watchdog become disabled

(6) **“Strobe” Button:** Strobe the watchdog. This button become available only when the watchdog runs in application-watch mode and the watchdog is enabled.

(7) **“Reboot” Button:** Reboot the machine by no strobe the watchdog hardware. This button is not available when the watchdog is disabled. If the watchdog is enabled and the user left clicks this “Reboot” button then all the three buttons: “Enable/Disable”, “Strobe” and “Reboot” becomes grayed, no operations can cancel the rebooting machine operation but stop the Advantech watchdog service

2.1.3 Event Log Database Operations

<Step1> Select “Time Span”, “System” and then click “Log Event”.

![Advantech Watchdog Service Configuration](image)
<Step2> Press “Apply”.

<Step3> Press “Enable”.
<Step4> Open Control Panel and then click “Administrative Tools”.

<Step5> Click “Event Viewer”.

UNO Watchdog Timer User Manual
<Step6> Click the item, “AdsWatchdog” you can view the event message.
2.1.4 Abort Tag
This tag displays the copyright information of the watchdog timer service.
Function API’s
Chapter 3  Function API’s

3.1 Error Codes Returned by Functions

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADS_WATCHDOG_ERROR_SUCCESS</td>
<td>This operation is success</td>
</tr>
<tr>
<td>ADS_WATCHDOG_ERROR_INITFAILED</td>
<td>Initialize watchdog failed</td>
</tr>
<tr>
<td>ADS_WATCHDOG_ERROR_DEINITFAILED</td>
<td>De-initialize the watchdog failed</td>
</tr>
<tr>
<td>ADS_WATCHDOG_ERROR_INVALID_HANDLE</td>
<td>invalid device handle</td>
</tr>
<tr>
<td>ADS_WATCHDOG_ERROR_INVALID_PARAMETER</td>
<td>Invalid input parameter</td>
</tr>
<tr>
<td>ADS_WATCHDOG_ERROR_WDT_RUNNING</td>
<td>The watchdog is running now and can not do this kind of operation</td>
</tr>
<tr>
<td>ADS_WATCHDOG_ERROR_WDT_NOTRUNNING</td>
<td>The watchdog is not running now, can not do this kind of operation</td>
</tr>
</tbody>
</table>

3.2 Data Structures

```c
enum WatchMode
{
    WATCH_MODE_SYSTEM = 0, WATCH_MODE_APPLICATION = 1;
};
```

Description:

(1) **WATCH_MODE_SYSTEM**: Watch the whole system, the feed dog thread is supplied in the SYS driver.

(2) **WATCH_MODE_APPLICATION**: Watch the specified application, the user should supply the user thread to feed the dog

```c
#define ADS_WATCHDOG_CHIPSET_UNKNOWN 0
#define ADS_WATCHDOG_CHIPSET_SOM443 1
#define ADS_WATCHDOG_CHIPSET_W83977AF 2
#define ADS_WATCHDOG_CHIPSET_W83627HF 3
```
enum WatchdogType
{
    WATCHDOG_TYPE_UNKNOWN = ADS_WATCHDOG_CHIPSET_UNKNOWN,
    WATCHDOG_TYPE_W83977AF = ADS_WATCHDOG_CHIPSET_W83977AF,
    WATCHDOG_TYPE_W83627HF = ADS_WATCHDOG_CHIPSET_W83627HF,
    WATCHDOG_TYPE_SOM443 = ADS_WATCHDOG_CHIPSET_SOM443
};

(1) WATCHDOG_TYPE_W83977AF: Winbond SuperIO W83977AF watchdog Chip
(2) WATCHDOG_TYPE_W83627HF: Winbond SuperIO W83627HF watchdog Chip
(3) WATCHDOG_TYPE_SOM443: The 443 standard watchdog Chip

3.2.1 Watchdog Driver Interfaces

LONG WDT_Init ( LONG * o_hHandle );
Description: Initialize the watchdog
Input Parameters
(1) o_hHandle: Handle of the watchdog driver

Return Values
(1) ADS_WATCHDOG_ERROR_SUCCESS: operation succeed
(2) ADS_WATCHDOG_ERROR_INITFAILED: Can not initialize the watchdog
LONG WDT_DeInit ( LONG * io_hHandle );
Description: De-initialize the watchdog
Input Parameters:
(1) io_hHandle: Handle of the watchdog

Return Values:
(1) ADS_WATCHDOG_ERROR_SUCCESS: operation succeed
(2) ADS_WATCHDOG_ERROR_DEINITFAILED: Can not de-initialize the watchdog
(3) ADS_WATCHDOG_ERROR_INVALID_HANDLE: Invalid device handle

LONG WDT_Enable ( LONG i_hHandle );
Description: Enable the watchdog
Input Parameters
(1) i_hHandle: Handle of the watchdog driver

Return Values:
(1) ADS_WATCHDOG_ERROR_SUCCESS: operation succeed
(2) ADS_WATCHDOG_ERROR_INVALID_HANDLE: Invalid device handle

LONG WDT_Disable ( LONG i_hHandle );
Description: Disable the watchdog
Input Parameters:
(1) i_hHandle: Handle of the watchdog driver

Return Values:
(1) ADS_WATCHDOG_ERROR_SUCCESS: operation succeed
(2) ADS_WATCHDOG_ERROR_INVALID_HANDLE: Invalid device handle
LONG WDT_SetMode( LONG i_hHandle, WatchMode i_watchMode );
Description: Set the watch mode of the watchdog
Input Parameters:
(1) i_hHandle: Handle of the watchdog driver
(2) i_watchMode: The mode of the watchdog

Return Values:
(1) ADS_WATCHDOG_ERROR_SUCCESS: operation succeed
(2) ADS_WATCHDOG_ERROR_INVALID_HANDLE: Invalid device handle
(3) ADS_WATCHDOG_ERROR_WDT_RUNNING: The watchdog is running now and can not change mode

LONG WDT_GetMode ( LONG i_hHandle, WatchMode * o_pWatchMode );
Description: Get the current running mode of the watchdog
Input Parameters:
(1) i_hHandle: Handle of the watchdog driver
(2) o_pWatchMode: The current watch mode of the watchdog

Return Values:
(1) ADS_WATCHDOG_ERROR_SUCCESS: operation succeed
(2) ADS_WATCHDOG_ERROR_INVALID_HANDLE: Invalid device handle

LONG WDT_SetTimerSpan ( LONG i_hHandle, DWORD i_dwIndex );
Description: Set the timer span of the watchdog driver
Input Parameters:
(1) i_hHandle: Handle of the watchdog driver
(2) i_dwIndex: The timer span index
Return Values:

1. **ADS_WATCHDOG_ERROR_SUCCESS**: operation succeed
2. **ADS_WATCHDOG_ERROR_INVALID_HANDLE**: Invalid device handle
3. **ADS_WATCHDOG_ERROR_WDT_RUNNING**: The watchdog is running now and can not set the timer span of the watchdog

LONG WDT_GetTimerSpan ( LONG i_hHandle, DWORD * o_pIndex, DWORD * o_pValue );
Description: Get the timer span of the watchdog

Input Parameters:

1. **i_hHandle**: Handle of the watchdog driver
2. **o_pIndex**: The timer span index
3. **o_pValue**: Current time span value of watchdog timer

Return Values:

1. **ADS_WATCHDOG_ERROR_SUCCESS**: operation succeed
2. **ADS_WATCHDOG_ERROR_INVALID_HANDLE**: Invalid device handle
3. **ADS_WATCHDOG_ERROR_WDT_NOTRUNNING**: Watchdog not running now

LONG WDT_Reboot ( LONG i_hHandle ); Description: Reboot the machine by the watchdog

Input Parameters:

1. **i_hHandle**: Handle of the watchdog driver

Return Values:

1. **ADS_WATCHDOG_ERROR_SUCCESS**: operation succeed
2. **ADS_WATCHDOG_ERROR_INVALID_HANDLE**: Invalid device handle
3. **ADS_WATCHDOG_ERROR_WDT_NOTRUNNING**: Watchdog not running now and can not reboot the machine
LONG WDT_Strobe ( LONG i_hHandle );
Description: Strobe the watchdog
Input Parameters:
(1) i_hHandle: Handle of the watchdog drier

Return Values:
(1) ADS_WATCHDOG_ERROR_SUCCESS: operation succeed
(2) ADS_WATCHDOG_ERROR_INVALID_HANDLE: Invalid device handle

LONG WDT_SetType ( LONG i_hHandle, WatchdogType i_watchdogType );
Description: Set the watchdog type
Input Parameters:
(1) i_hHandle: Handle of the watchdog driver
(2) i_watchdogType: The type of the watchdog

Return Values:
(1) ADS_WATCHDOG_ERROR_SUCCESS: operation succeed
(2) ADS_WATCHDOG_ERROR_INVALID_HANDLE: Invalid device handle

LONG WDT_GetType ( LONG i_hHandle, WatchdogType * o_pWatchdogType );
Description: Get the watchdog type
Input Parameters:
(1) i_hHandle: Handle of the watchdog driver
(2) o_pWatchdogType: The type of the watchdog

Return Values:
(1) ADS_WATCHDOG_ERROR_SUCCESS: operation succeed
(2) ADS_WATCHDOG_ERROR_INVALID_HANDLE: Invalid device handle
LONG WDT_IsEnabled (LONG i_hHandle, BOOL * o_bEnabled);
Description: Get the watchdog’s running status: Enabled or Disabled
Input Parameters:
(1) i_hHandle: Handle of the watchdog driver
(2) o_bEnabled: The watchdog current running status, TRUE for enabled and FALSE for disabled

Return Values:
(1) ADS_WATCHDOG_ERROR_SUCCESS: operation succeed
(2) ADS_WATCHDOG_ERROR_INVALID_HANDLE: Invalid device handle

LONG WDT_LogEvent (LONG i_hHandle, BOOL i_bLog);
Description: Set the watchdog operations: Enable, Disable and Reboot to be logged into the system event base or not
Input Parameters:
(1) i_hHandle: Handle of the watchdog driver
(2) i_bLog: TRUE for log the three operations into system base, FALSE for not logging.

Return Values:
(1) ADS_WATCHDOG_ERROR_SUCCESS: operation succeed
(2) ADS_WATCHDOG_ERROR_INVALID_HANDLE: Invalid device handle
(3) ADS_WATCHDOG_ERROR_WDT_RUNNING: Watchdog is running now

LONG WDT_IsLogged (LONG i_hHandle, BOOL * o_bLogged);
Description: Get the watchdog event log information: The “Enable”, “Disable”, “Reboot” operation logged into system event base or not
Input Parameters:
(1) i_hHandle: Handle of the watchdog driver
(2) o_bLogged: TRUE for log the “Enabled”, “Disable”, “Reboot” operations into system base, FALSE for not logging.
Return Values:
(1) \texttt{ADS\_WATCHDOG\_ERROR\_SUCCESS}: operation succeed
(2) \texttt{ADS\_WATCHDOG\_ERROR\_INVALID\_HANDLE}: Invalid device handle

\texttt{LONG WDT\_GetStartupTime (LONG i\_hHandle,}
\texttt{LARGE\_INTEGER * o\_pSartupTime);}\
Description: Get the watchdog enabled time
Input Parameters:
(1) \texttt{i\_hHandle}: Handle of the watchdog driver
(2) \texttt{o\_pSartupTime}: The count of 100-nanosecond intervals that the watchdog is enabled.

Return Values:
(1) \texttt{ADS\_WATCHDOG\_ERROR\_SUCCESS}: operation succeed
(2) \texttt{ADS\_WATCHDOG\_ERROR\_INVALID\_HANDLE}: Invalid device handle

\texttt{LONG WDT\_GetTimerSpanDescription ( LONG i\_hHandle,}
\texttt{DWORD i\_dwIndex, LPTSTR o\_pDescription )};\
Description: Get the description of the specified timer span
Input Parameters:
(1) \texttt{i\_hHandle}: Handle of the watchdog driver
(2) \texttt{i\_dwIndex}: The timer span index
(3) \texttt{o\_pDescription}: The description of the specified timer index. The memory pointed by this pointer should be allocated and initialized before transferred into this function, as well should be de-allocated outside this function. The buffer size should be large enough to load 64 characters.
Return Values:

1. **ADS_WATCHDOG_ERROR_SUCCESS**: operation succeed
2. **ADS_WATCHDOG_ERROR_INVALID_HANDLE**: Invalid device handle
3. **ADS_WATCHDOG_ERROR_INVALID_PARAMETER**: Invalid timer span index

LONG WDT_GetErrMsg ( LONG i_hHandle, LONG i_lErrCode, LPTSTR o_pErrMsg );

Description: Get the error description of the specified error code

Input Parameters:

1. **i_hHandle**: Handle of the watchdog driver
2. **i_lErrCode**: The error code returned by a function call
3. **o_pErrMsg**: The pointer to a buffer to store the error message associated with a specified error code. The memory pointed by this pointer should be allocated and initialized before transferred into this function, as well should be de-allocated outside this function. The buffer size should be large enough to load 64 characters.

Return Values:

1. **ADS_WATCHDOG_ERROR_SUCCESS**: operation succeeds
2. **ADS_WATCHDOG_ERROR_INVALID_HANDLE**: Invalid device handle
3. **ADS_WATCHDOG_ERROR_INVALID_PARAM**: The error code is invalid
Example Call Flows
Chapter 4  Example Call Flows

4.1  ElapsedTme

Path:
C:\Program Files\ADVANTECH\Watchdog\Example\Console\Elapsed-Time\Elapsedtime.cpp

Purpose: Get the elapsed time of watchdog timer.

![Flowchart diagram showing the process of ElapsedTme]

1. Call `WDT_Init()`
2. Check if `WDT_IsEnabled()` is True or False
   - If False, exit
   - If True, call `WDT_GetStartTime()`
3. Call `WDT_Deinit()`
4.2 RebootMachine

Path:
C:\Program Files\ADVANTECH\Watchdog\Example\Console\Rebootmachine\Rebootmachine.cpp

Purpose: Describe the process of reboot machine if enable watchdog timer.

```
WDT_Init()

WDT_IsEnabled() ->
No          Yes

WDT_Enable()

WDT_Reboot()

WDT_Deinit()
```
4.3 Setlog

Path:
C:\Program Files\ADVANTECH\Watchdog\Example\Console\Setlog\Setlog.cpp

Purpose: Describe how to record the watchdog timer history in event log.
4.4 Setmode

Path:
C:\Program Files\ADVANTECH\Watchdog\Example\Console\Setmode\Setmode.cpp

Purpose: Describe how to switch the mode of watchdog timer, system mode or application mode.

```
WDT_Init()

WDT_IsEnabled()
    Yes -> WDT_Disable()
    No

WDT_GetMode()
    System Mode -> WDT_LogEvent(handle, WATCH_MODE_APPLICATION)
    Application Mode -> WDT_LogEvent(handle, WATCH_MODE_SYSTEM)

WDT_Deinit()
```
4.5 TimerSpan

Path:
C:\Program Files\ADVANTECH\Watchdog\Example\Console\SetTimer-Span\Timerspan.cpp

Purpose: Define how to set the time span of watchdog timer.
4.6 WatchApplication

Path:
C:\Program Files\ADVANTECH\Watchdog\Example\Console\WatchApplication\EnableDisable.cpp

Purpose: Enable watchdog timer function under Application Mode. You can refer to the source code of “SetMode” to change mode to Application Mode.

```
WDT_Init()

WDT_GetMode

System Mode

WDT_Init()

WDT_DeInit()

SetupUserWatchApplication()

WDT_IsEnabled()

No

WDT_Enable()

Yes

Press any key to continue

WDT_Disable()

WDT_DeInit()
```
1. SetupUserWatchApplication()
2. Create Windows Event & Timer
   Enter watchdog timer span
   0. Use Default: 2000
   1. Watchdog hardware timer span
   Create thread
   Return
2. Create thread
   WaitForMultipleObjects()
   Timer event occur
   WDT_Strobe()
   Trigger
   Exit event occur
   Cancel timer
   Return
4.7 WatchSystem

Path:
C:\Program Files\ADVANTECH\Watchdog\Example\Console\WatchSystem\Enable-Disable.cpp Purpose: Enable watchdog timer function under System Mode. You can refer to the source code of “SetMode” to change mode to System Mode.