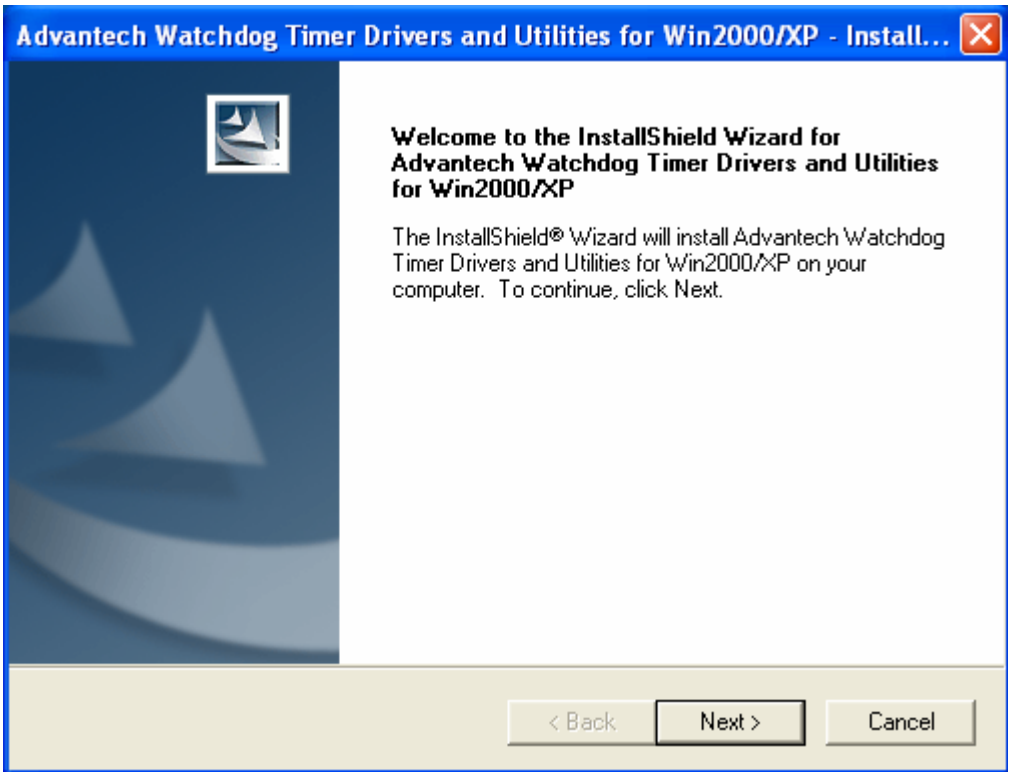
UNO-2170 Watchdog Timer User Manual

The UNO-2170 support watchdog timer function. If the UNO-2170 system crashes or hangs, the Watchdog timer will reboot the system. There is no need for an engineer to physically reboot the system. This is beneficial in unmanned installations as well as critical processes where minimum interruption is needed.

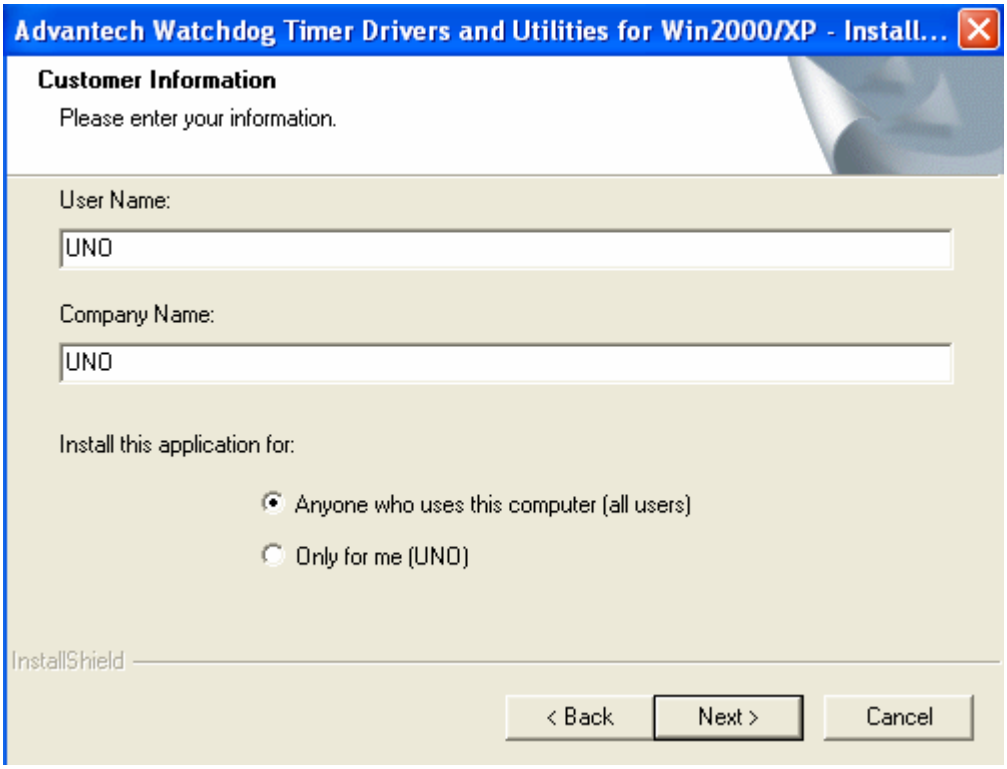
1. Watchdog Timer Installation Guide

This chapter introduces how to install UNO-2170 Watchdog timer driver under Windows 2K/XP platform.

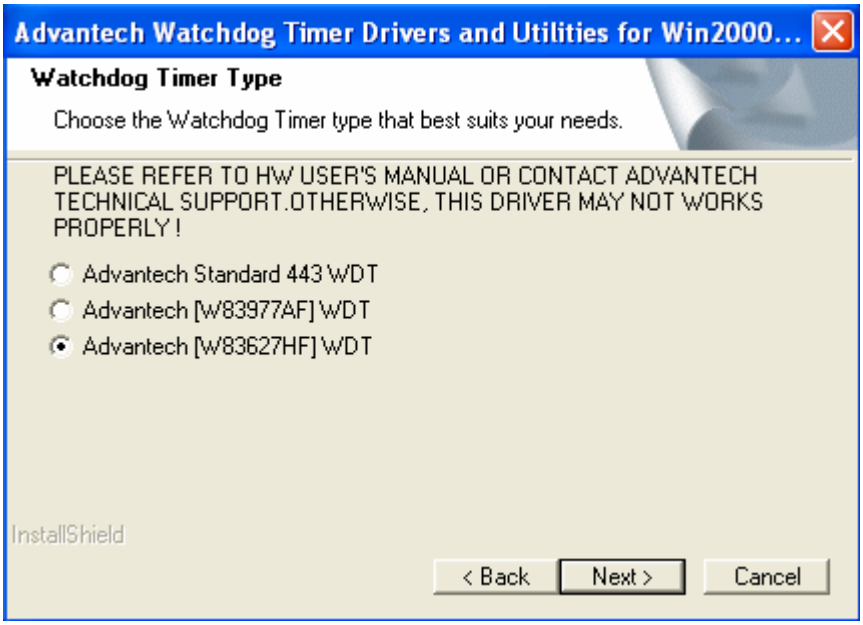
<Step1> Insert the UNO-2000 Driver and Utility CD in the CD-ROM, and execute **AdvWDT.exe** from following path: "\\UNO-2170\Driver\Watchdog Timer\Win2K.XP\". Click the Next button to install watchdog timer.



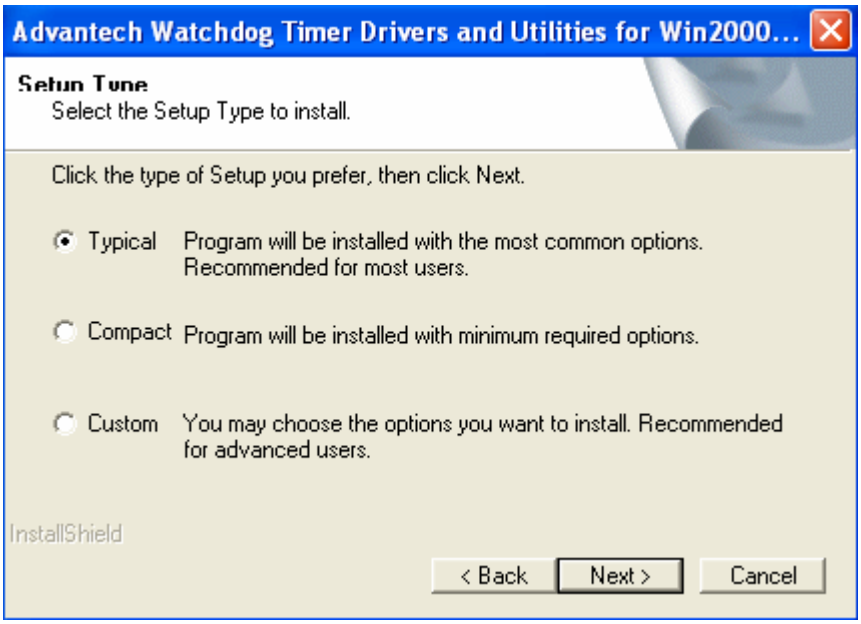
<Step2> Enter your name and company name in the blank, and press Next button for next step.



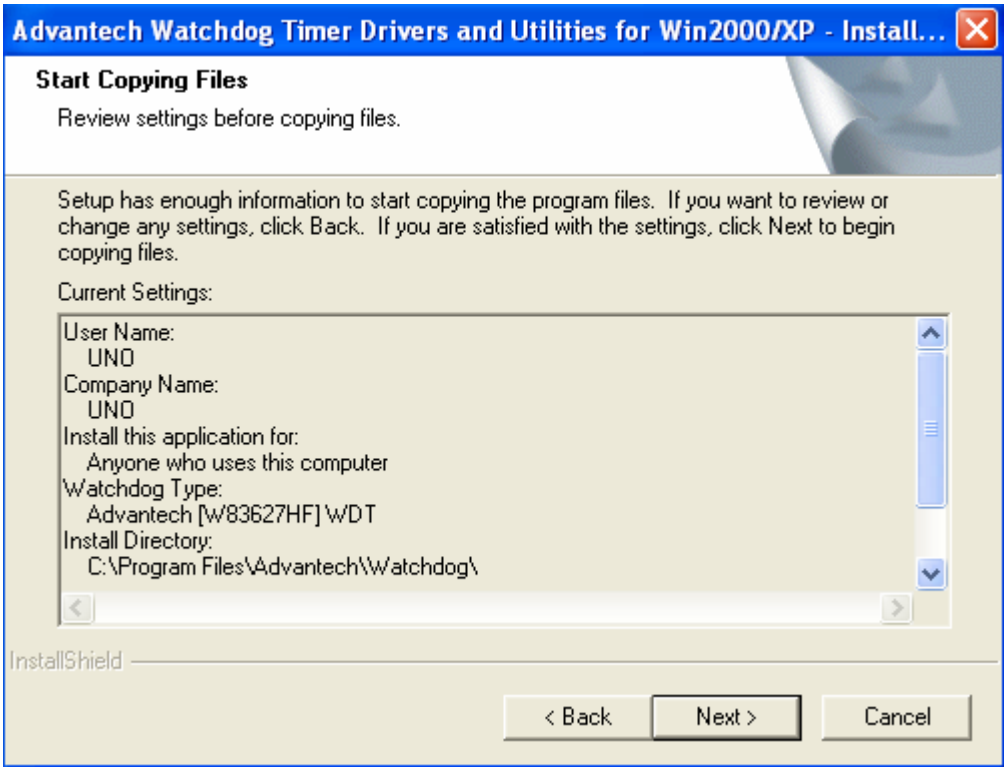
<Step3> For UNO-2170, please select “Advantech [W83627HF] WDT” and then press the Next button for next step.



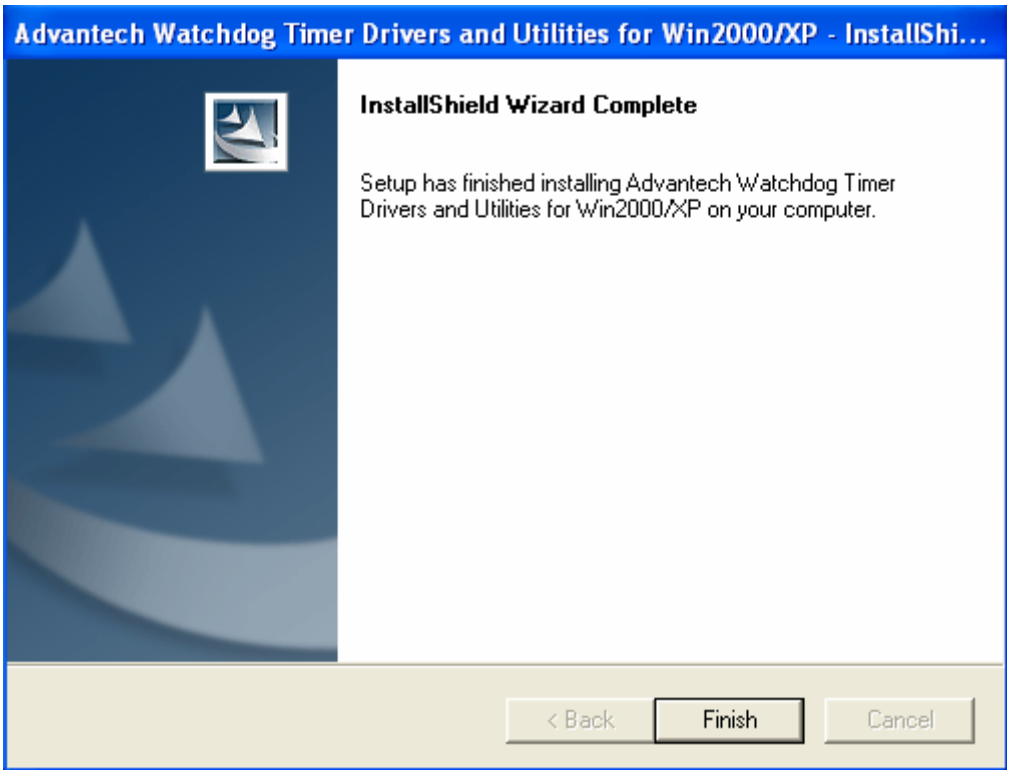
<Step4 > Select “Typical”, the watchdog timer driver will be installed with the most common options. Recommended for most users.



<Step 5> Press Next to start the installation.



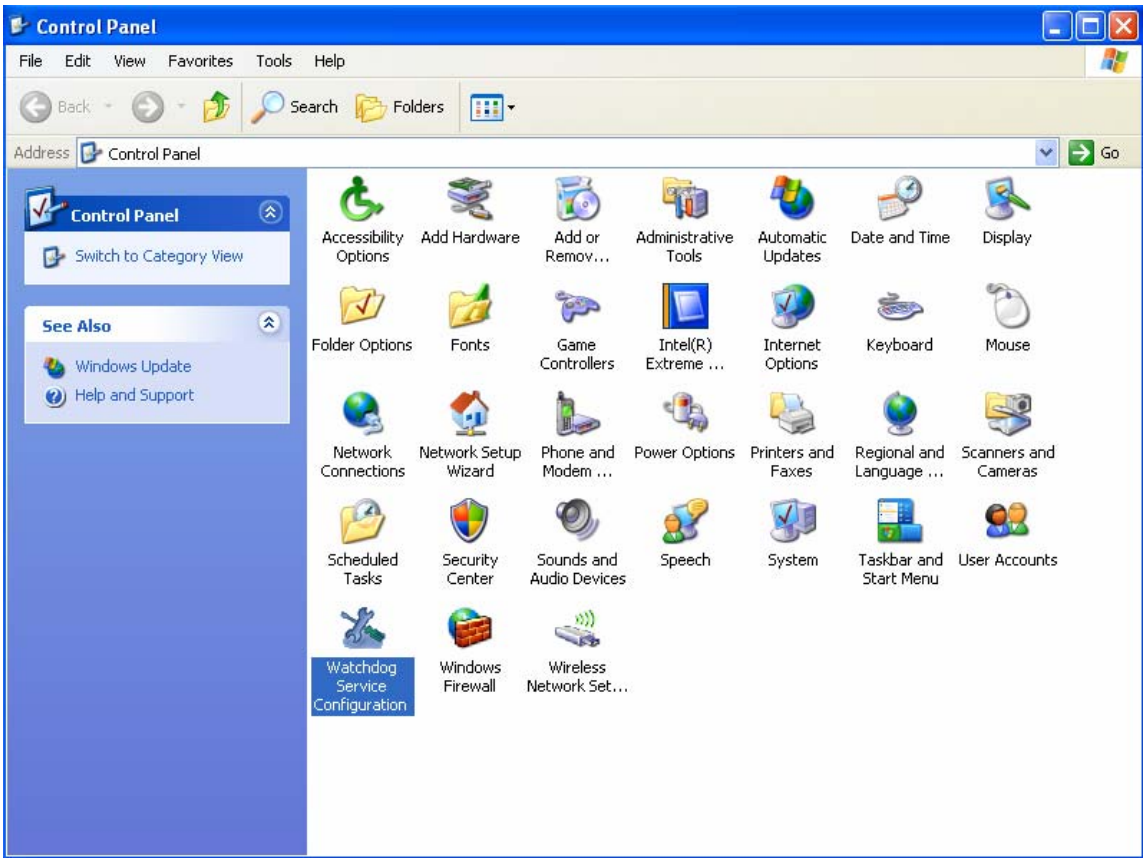
<Step 6> Press the Finish button to finish the watchdog timer installation. If UNO-2170 watchdog timer is installed successfully, the relevant applications and samples will be placed in following path: C:\\Program Files\ADVANTECH\Watchdog\



<Step 7> Press OK to reboot the system.

2. Watchdog Timer Configuration

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



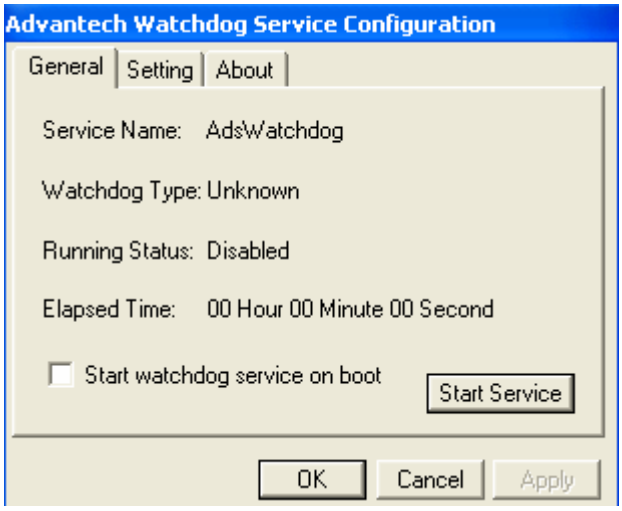
2.1 User Interface

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

General tag: display some general information on the Advantech watchdog service

Setting tag: display all the setting information on the Advantech watchdog service

About tag: display the copyright information of the Advantech watchdog service



2.1.1 General tag

There are five items mentioned in General tag,

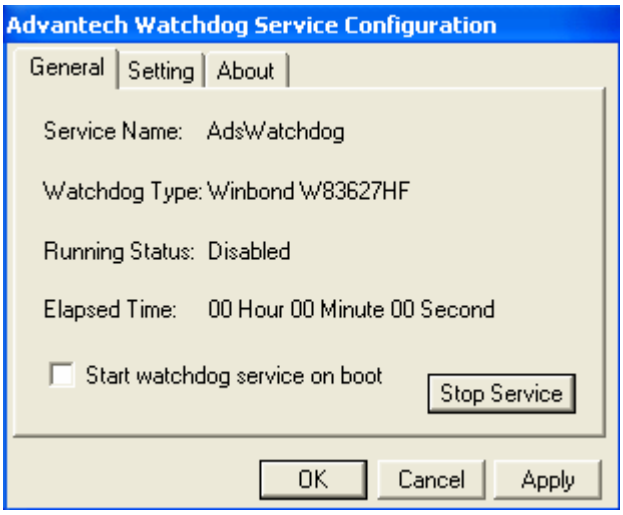
Service Name: display the name of the Advantech watchdog service in the Service Control Manager(SCM) database

Watchdog Type: display the type of the watchdog chipset type. If you are using UNO-2170, it will display “Winbond W83627HF

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

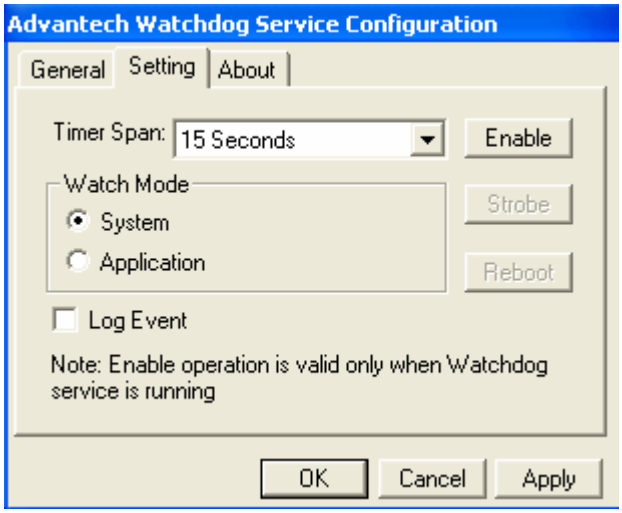
Elapsed time: The elapsed time from the time that the watchdog is enabled. If the watchdog is disabled the elapsed time will be 00 hour 00 minute 00 second.

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 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, system will warn you to "strobe" (reset) the watchdog timer, otherwise the system will reboot in a specified time period.

Definition:

System mode: watchdog timer is running in Windows background. If the hardware is hung up, it will reboot system 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 operation will be not be logged into the system event base.

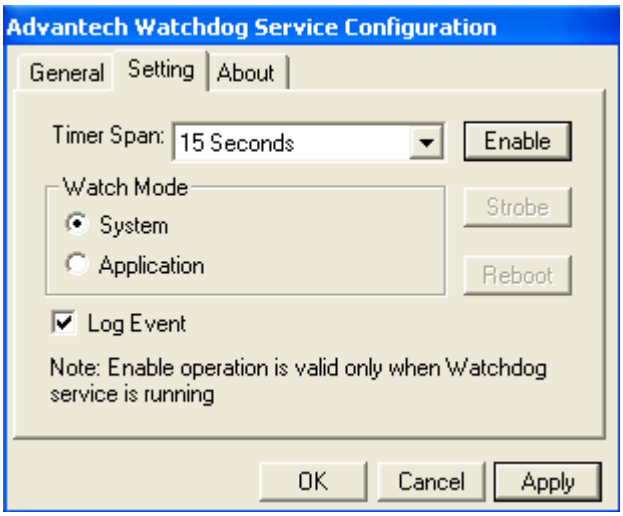
(4) "Enable/Disable" button: Enable or disable the watchdog. If the watchdog is enabled the user cannot change the watch mode and the timer span of the watchdog, so these related controls become grayed. These controls resume to theirs normal status when the watchdog become disabled

(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

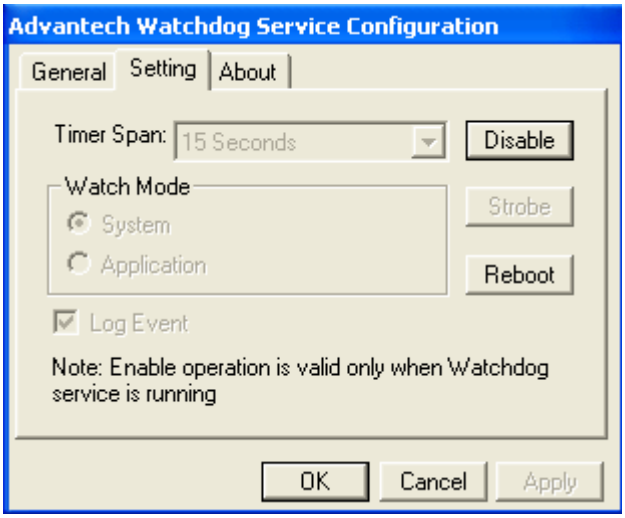
2.1.3 View the logged operations of watchdog service configuration in Event Log Database

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

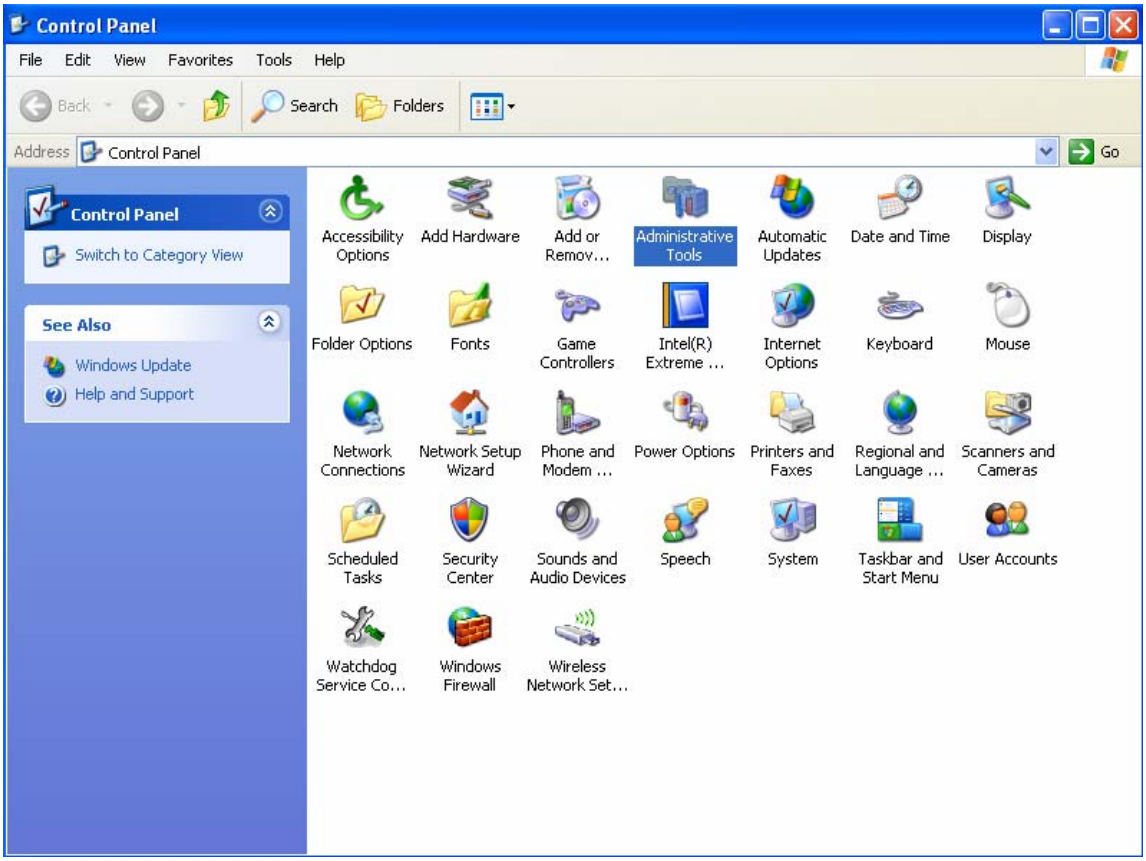


<Step2> Press “Apply”.

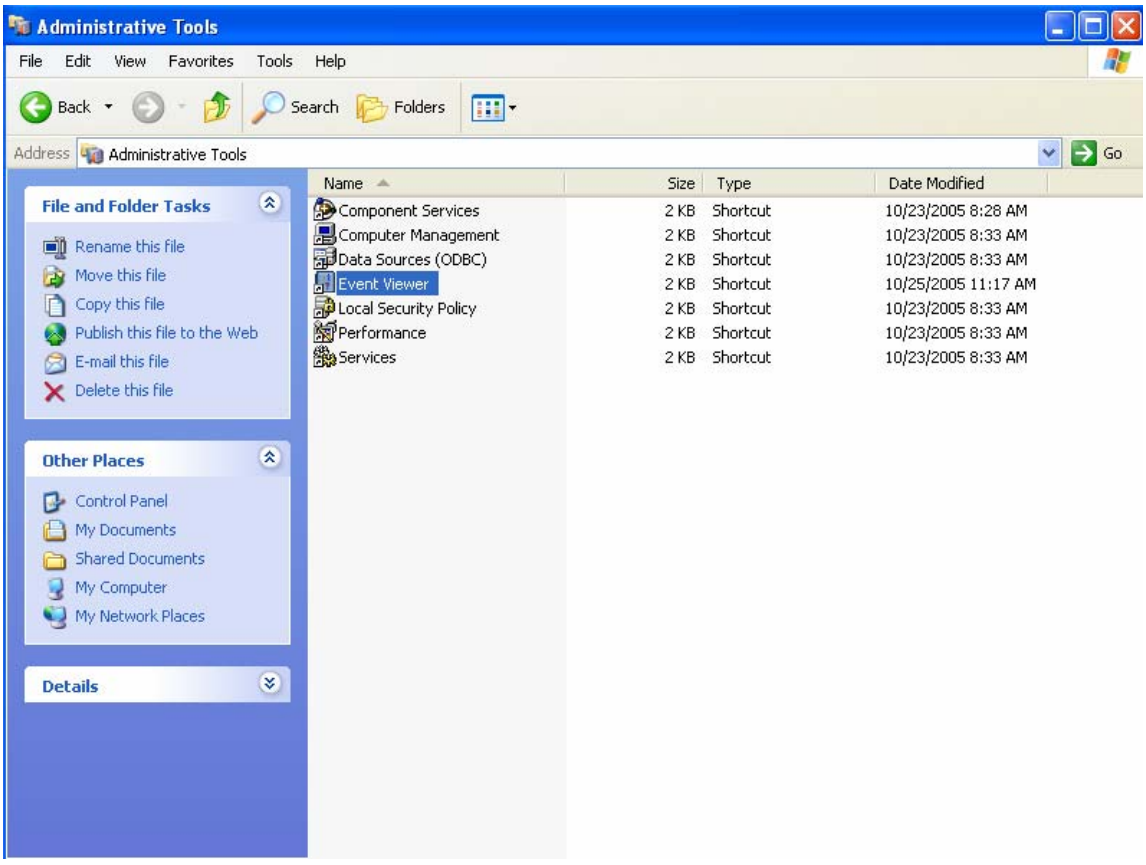
<Step3> Press “Enable”.



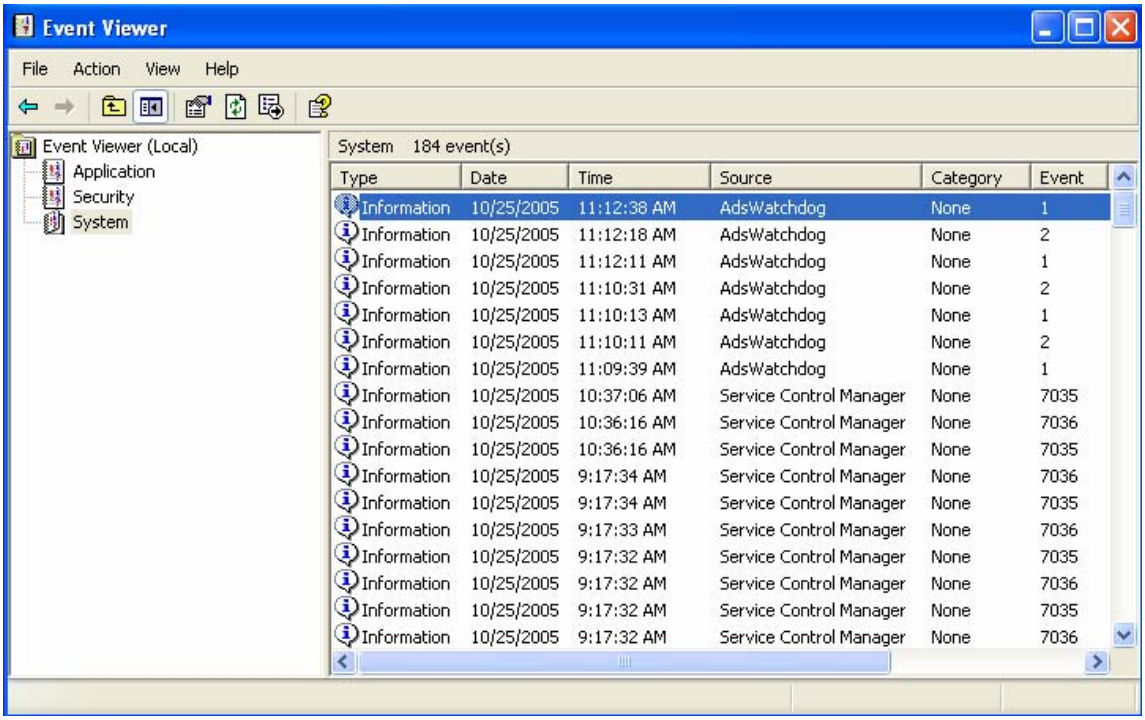
<Step4> Open Control Panel and then click “Administrative Tools”.

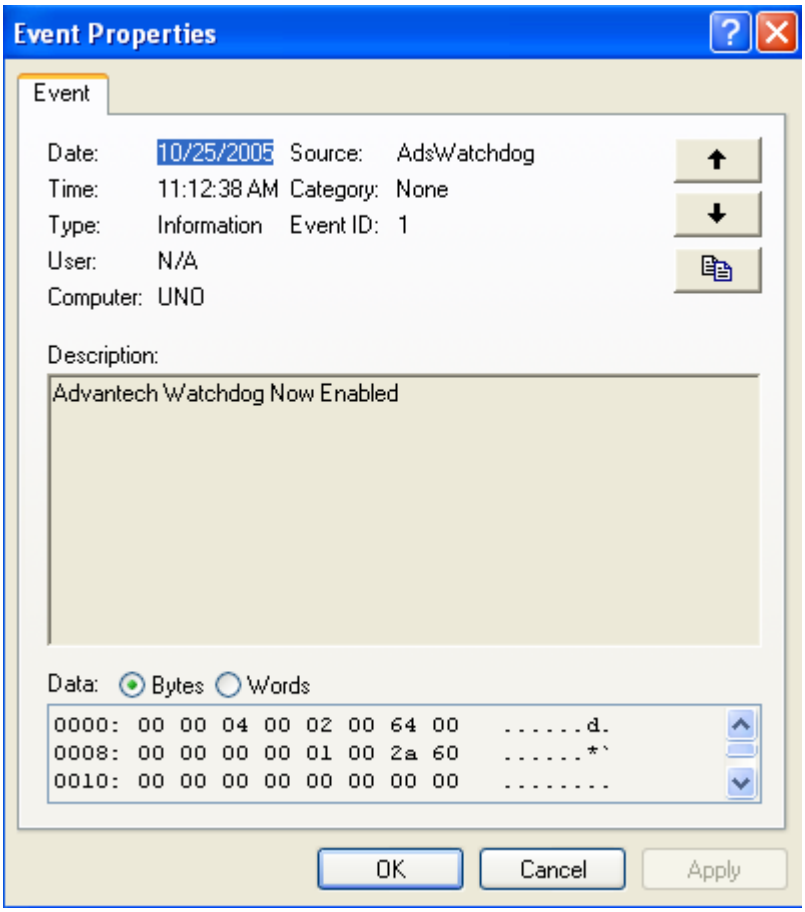


<Step5> Click “Event Viewer”.



<Step6> Click the item, “AdsWatchdog” and you can view the event message.





2.1.4 Abort tag

This tag displays the copyright information of Advantech watchdog timer service.

3. Function APIs

3.1 Error Codes Returned by Functions

Error Code	Description
ADS_WATCHDOG_ERROR_SUCCESS	This operation is success
ADS_WATCHDOG_ERROR_INITFAILED	Initialize watchdog failed
ADS_WATCHDOG_ERROR_DEINITFAILED	De-initialize the watchdog failed
ADS_WATCHDOG_ERROR_INVALID_HANDLE	Invalid device handle
ADS_WATCHDOG_ERROR_INVALID_PARAMETER	Invalid input parameter
ADS_WATCHDOG_ERROR_WDT_RUNNING	The watchdog is running now and can not do this kind of operation
ADS_WATCHDOG_ERROR_WDT_NOTRUNNING	The watchdog is not running now, can not do this kind of operation

3.2 Data Structures

```
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

```
#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: The Winbond SuperIO W83977AF watchdog Chip

(2) WATCHDOG_TYPE_W83627HF: The Winbond SuperIO W83627HF watchdog Chip

(3) WATCHDOG_TYPE_SOM443: The Advantech 443 standard watchdog Chip

3.3 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 drvier

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) ADS_WATCHDOG_ERROR_SUCCESS: operation succeed
(2) ADS_WATCHDOG_ERROR_INVALID_HANDLE: Invalid device handle

LONG WDT_GetStartupTime (LONG i_hHandle, LARGE_INTEGER * o_pSartupTime);

Description: Get the watchdog enabled time

Input Parameters:

(1) i_hHandle: Handle of the watchdog driver

(2) o_pSartupTime: The count of 100-nanosecond intervals that the watchdog is enabled.

Return Values:

(1) ADS_WATCHDOG_ERROR_SUCCESS: operation succeed

(2) ADS_WATCHDOG_ERROR_INVALID_HANDLE: Invalid device handle

LONG WDT_GetTimerSpanDescription (LONG i_hHandle, DWORD i_dwIndex, LPTSTR o_pDescription);

Description: Get the description of the specified timer span

Input Parameters:
(1) i_hHandle: Handle of the watchdog driver
(2) i_dwIndex: The timer span index
(3) 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

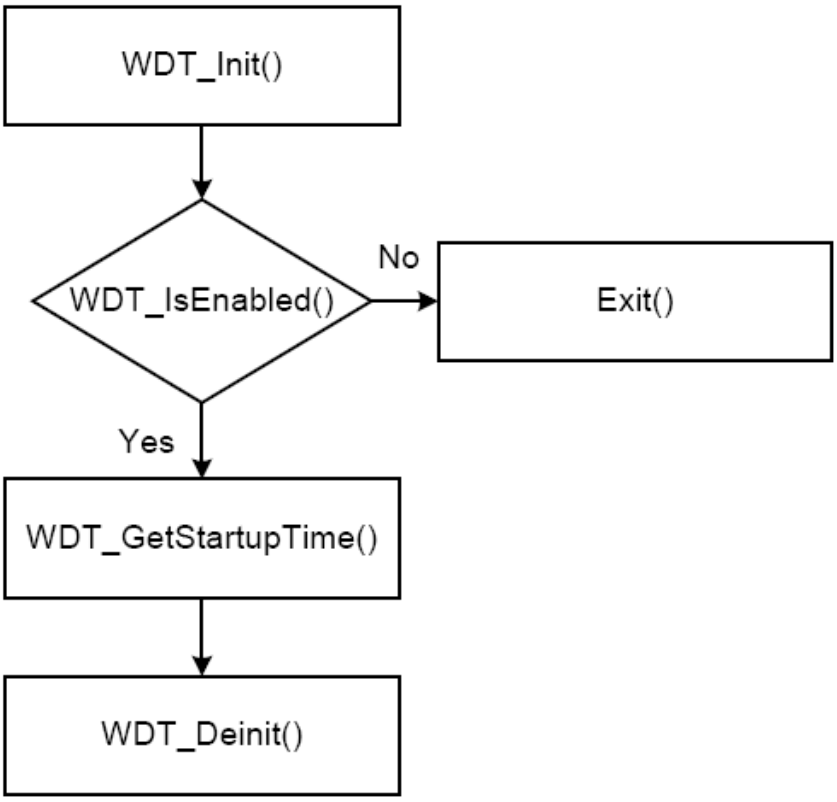
3.3 Example Call Flows

3.3.1 ElapsedTme

Path:

C:\Program Files\ADVANTECH\Watchdog\Example\Console\ElapsedTime\Elapsedtime.cpp

Purpose: Get the elapsed time of watchdog timer.

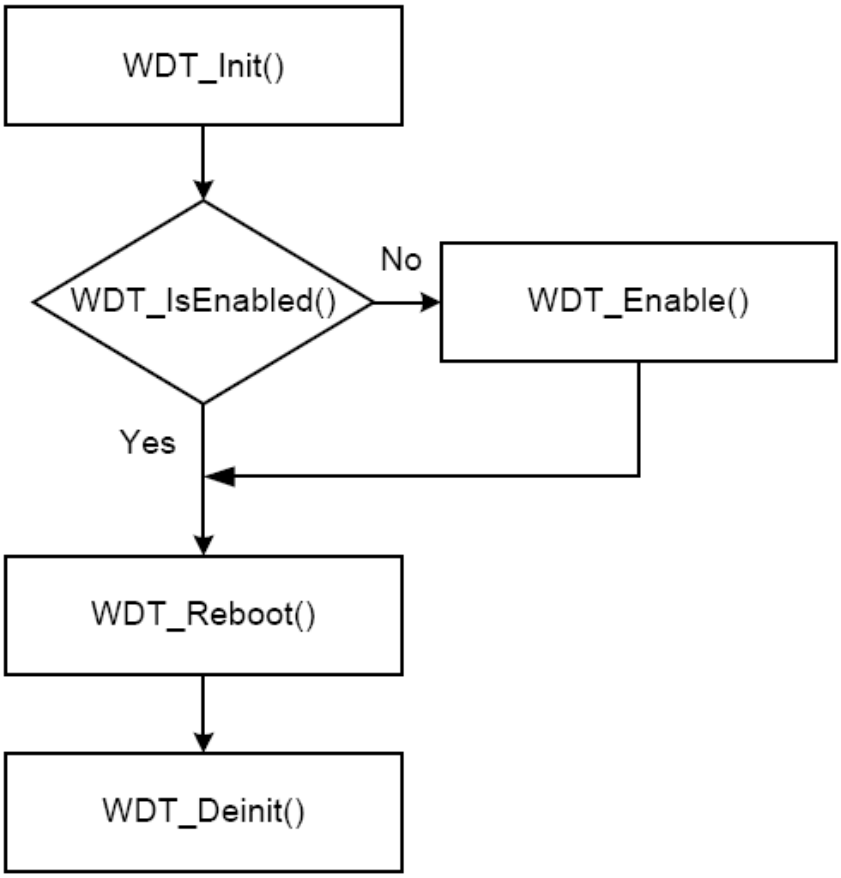


3.3.2 RebootMachine

Path:

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

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

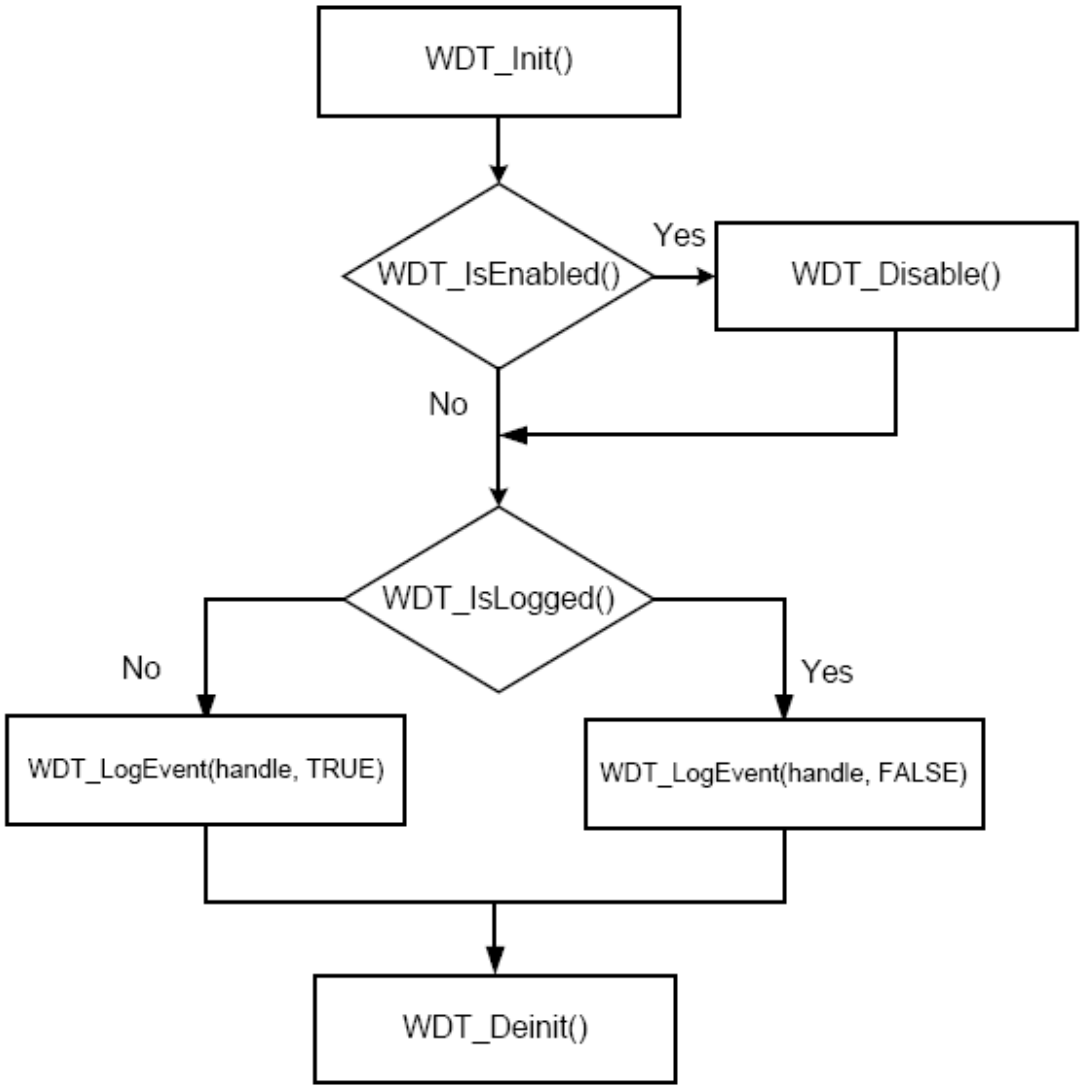


3.3.3 Setlog

Path:

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

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

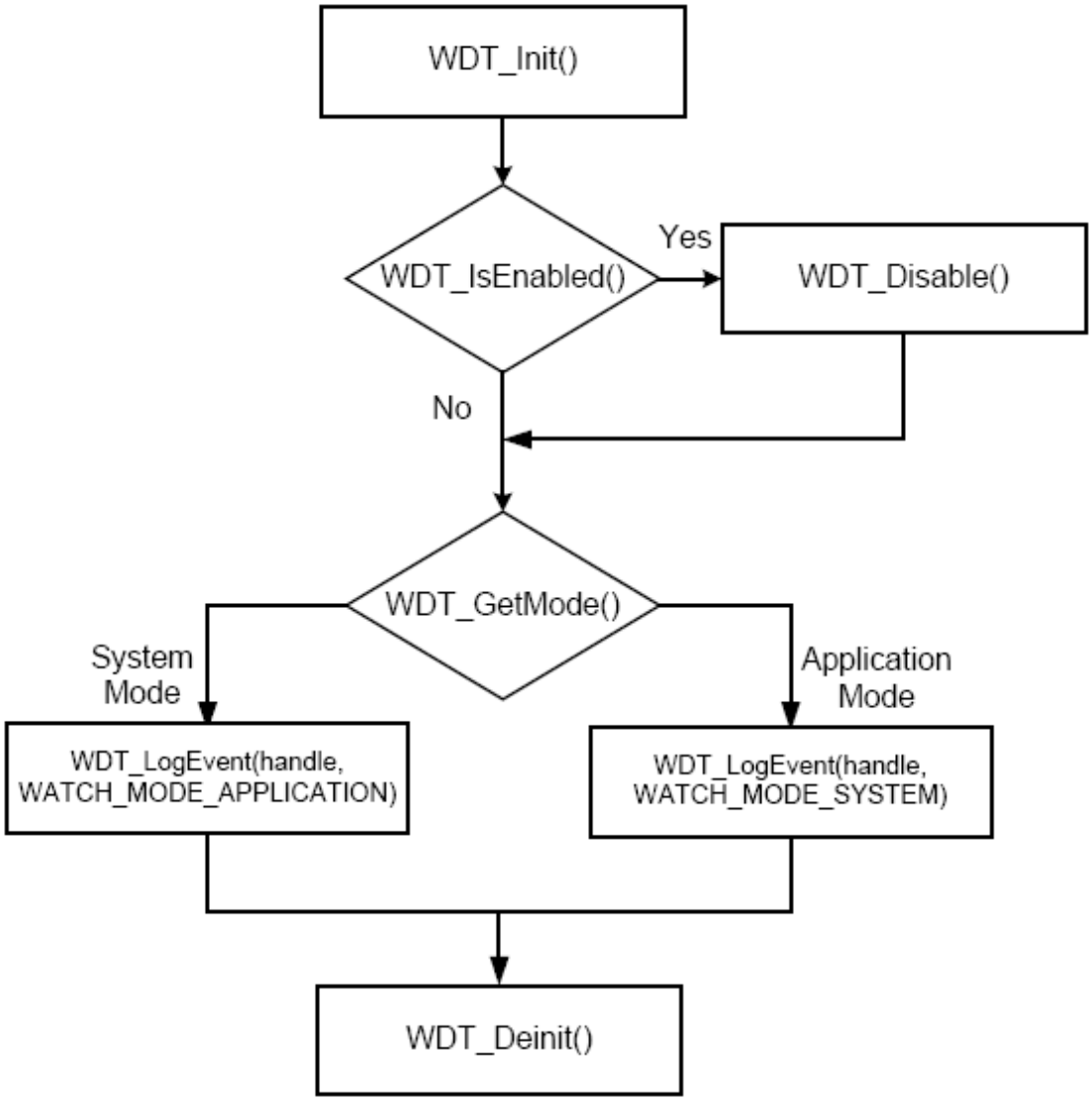


3.3.4 Setmode

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.

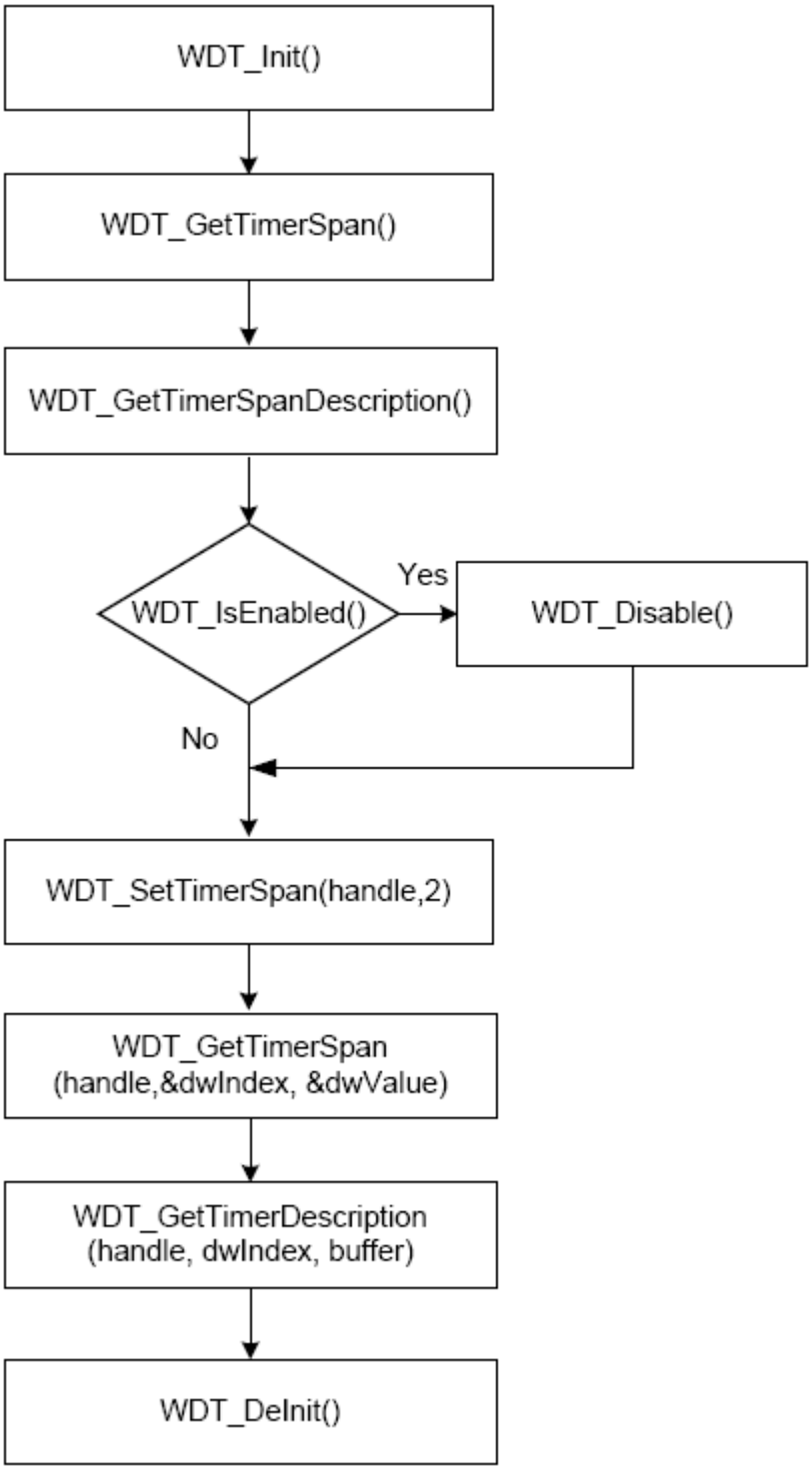


3.3.5 TimerSpan

Path:

C:\Program Files\ADVANTECH\Watchdog\Example\Console\SetTimerSpan\Timerspan.cpp

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

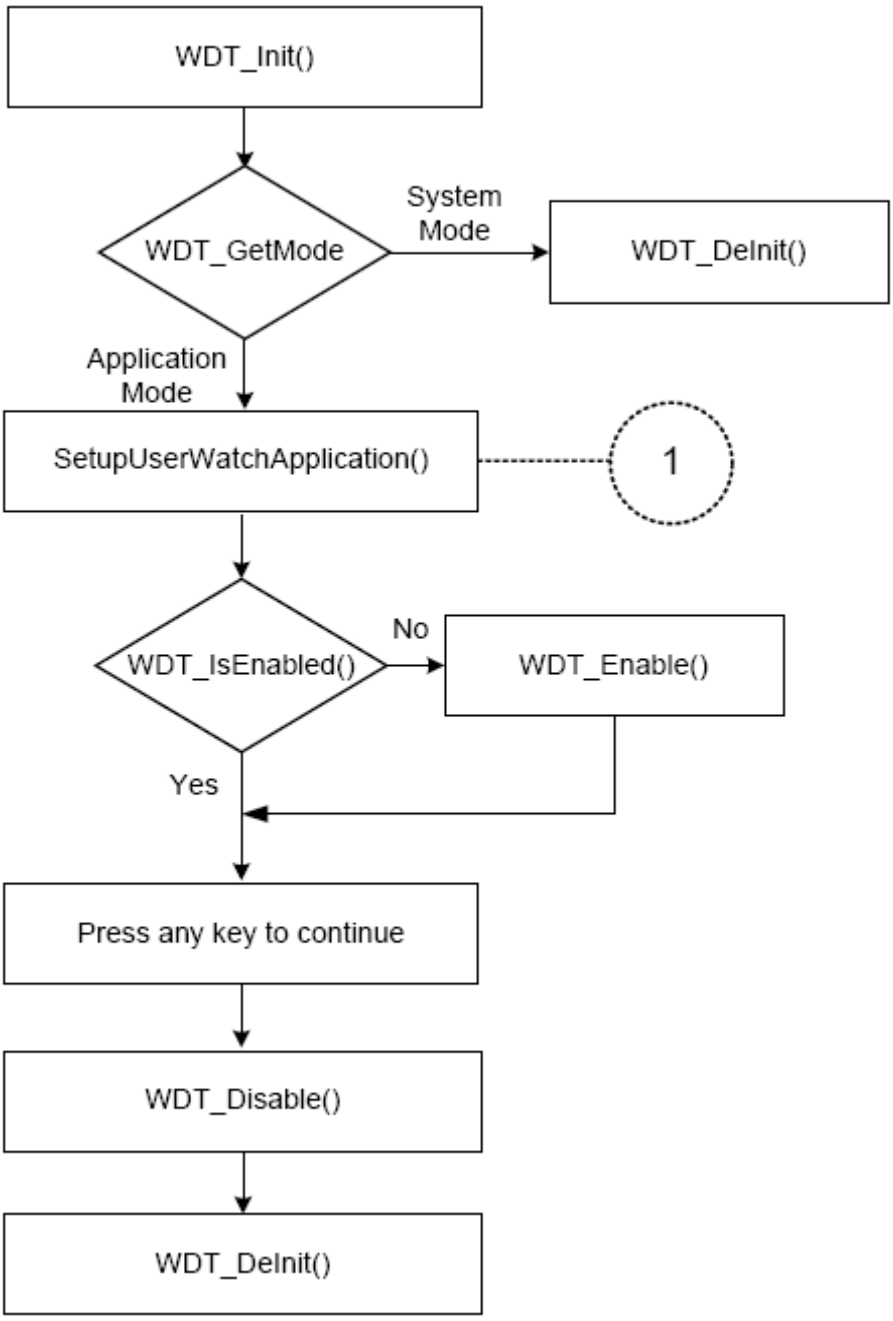


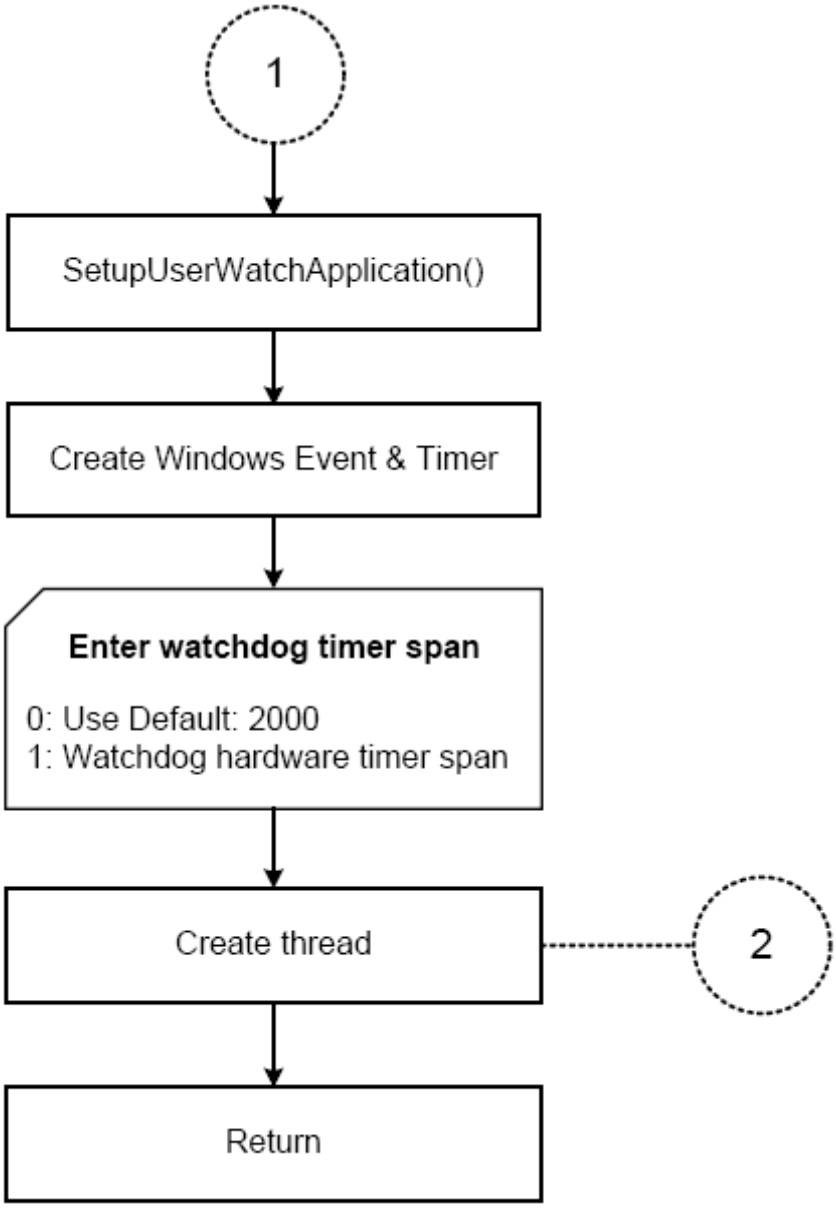
3.3.6 WatchApplication

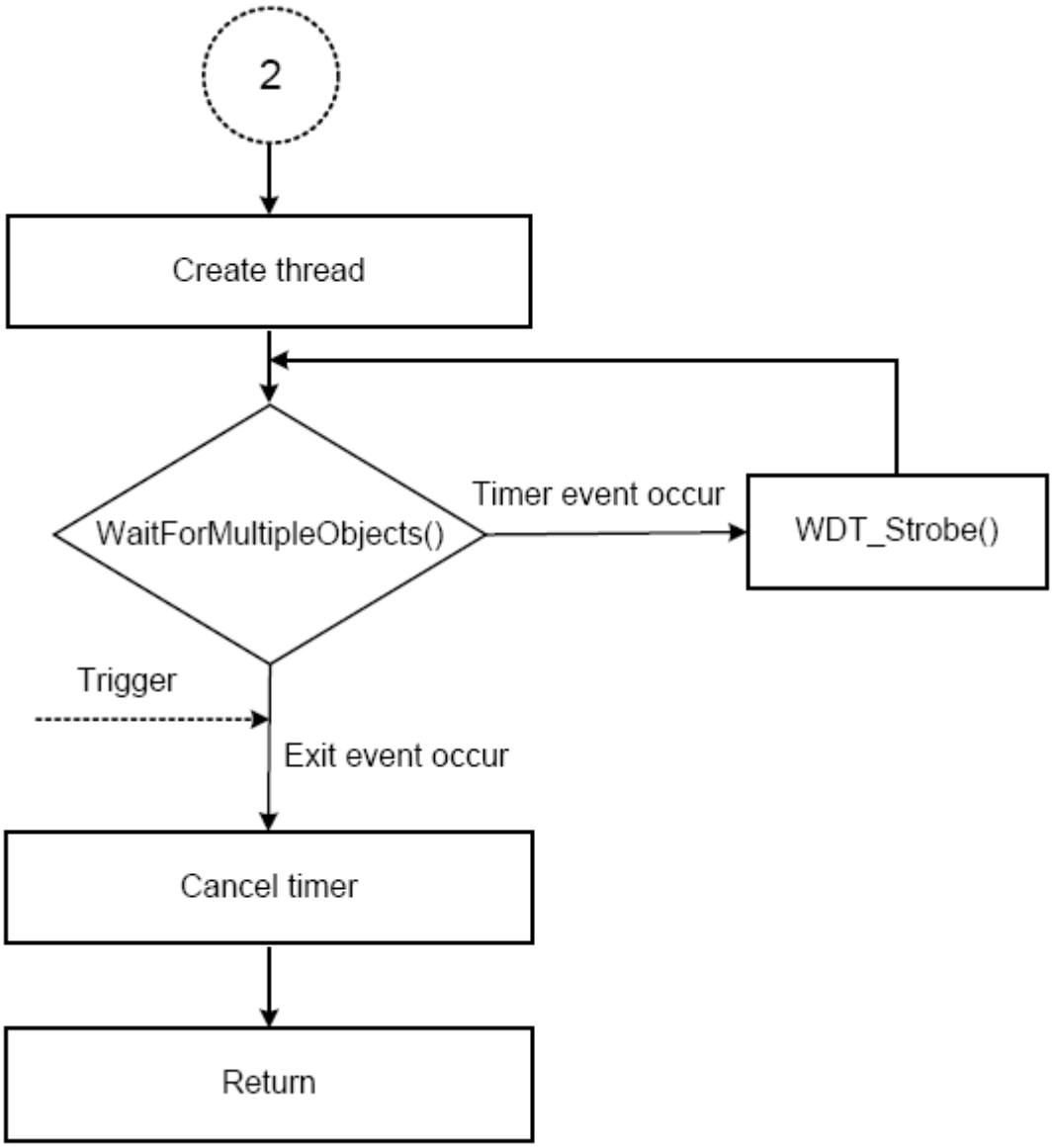
Path:

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.




1
SetupUserWatchApplication()
Create Windows Event & Timer
Enter watchdog timer span
0: Use Default: 2000
1: Watchdog hardware timer span
Create thread
2
Return


2
Create thread
WaitForMultipleObjects()
Timer event occur
WDT_Strobe()
Trigger
Exit event occur
Cancel timer
Return

3.3.7 WatchSystem

Path:

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

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