Industrial IoT

Advantech Watchdog

KMDF Driver

User Manual

For Microsoft Windows

Version <1.11>

Revision History

Date	Version	Description
2023/03/23	1.11	Install Picture
2022/07/01	1.10	For new driver to update chap. 3, 6.1.15, 6.2.1, 6.3.2, 7.1.3, 7.2.2 And figures
2019/10/29	1.01	Change page format.
2017/06/18	1.00	Initial draft.

Table of Contents

Advantech Watchdog KMDF Driver

1. Welcome to Advantech Watchdog KMDF Driver

1.1 About This Manual

This manual contains the information you need to get started with the Advantech Watchdog KMDF Driver.

The Advantech Watchdog KMDF Driver allows you to easily perform versatile WDT operations through properties, methods, and events in programs developed with Microsoft Visual C++, Microsoft Visual Basic, and Microsoft C#.

This manual describes Application Programming Interface (API), including calling procedure of operating Watchdog device and Control Panel Program (CPL) of the driver.

This manual also contains step-by-step instructions for building applications with the Advantech Watchdog KMDF Driver. With the help of Advantech Watchdog KMDF Driver, you can develop applications with tools like VC, VC.NET, VB.NET, and C#.NET in different Windows operating systems (Windows 2000/XP/Vista/7/8/8.1/10/Windows embedded).

The Advantech Watchdog KMDF Driver provides examples explaining how to use Advantech Watchdog with series of real examples and offers reference to develop your own applications. You can modify these sample applications to meet your needs.

This manual does not show you how to solve every possible programming problem. To use this manual, you should already be familiar with at least one of the supported programming environments and Windows 2000/XP/Vista/7/8/8.1/10/Windows Embedded.

1.2 Organization of This Manual

This user manual is divided into the following sections:

- Welcome to Advantech Watchdog KMDF Driver
- Advantech Watchdog KMDF Driver Overview
- Control Panel Program
- Getting Started with Advantech Watchdog KMDF Driver
- Programming Guide
- Function Reference
- Device Driver Programming Examples

Welcome to Advantech Watchdog KMDF Driver

Give an overview of this manual.

Advantech Watchdog KMDF Driver Overview

Give an overview of Advantech Watchdog KMDF Driver.

Control Panel Program

Give a thorough description of the Control Panel Program.

Getting Started with Advantech Watchdog KMDF Driver

Give the novice a clear concept of the Advantech Watchdog KMDF Driver and a walk-through in creating a simple application. Step-by-step instructions are provided for an application written in Win32 MFC development environments.

Programming Guide

Show function call flowchart for working in System Mode or Application Mode.

Functions Reference

- ***Function Description***

  Give a thorough description of all the functions supported by Advantech Watchdog KMDF Driver.

- ***Data Structure***

Describe the data structures that related to the provided functions.

- ***Error Codes***

  Explain the error codes that might be returned when calling functions provided by the Advantech Watchdog KMDF Driver. Refer to this section when debugging your application.

Device Driver Programming Examples

This chapter gives an overview of the examples we provided.

2. Advantech Watchdog KMDF Driver Overview

The Advantech Watchdog KMDF Driver provides functions to maximize the hardware's performance. It is freely bundled with Advantech Watchdog Device.

With this Driver, you don't have to use hardware-specific register commands and it gives you a powerful programming API for using with a variety of programming environments and languages.

2.1 Overview

Advantech Watchdog KMDF Driver contains a set of functions and related structures that can be used in various application programs for interfacing with KMDF Drivers. The APIs support Microsoft Visual C++, Microsoft Visual Basic, and Microsoft C# development environments. You can directly write applications with windows API. Examples of VC, VC.NET, VB.NET, and C#.NET are supplied in the package, providing a reference for you to develop applications. When developing work is completed, you can use test tools to verify if functions of the application are correct.

Figure 1 shows the relationship between the Advantech Watchdog KMDF driver with the user application and Watchdog hardware.

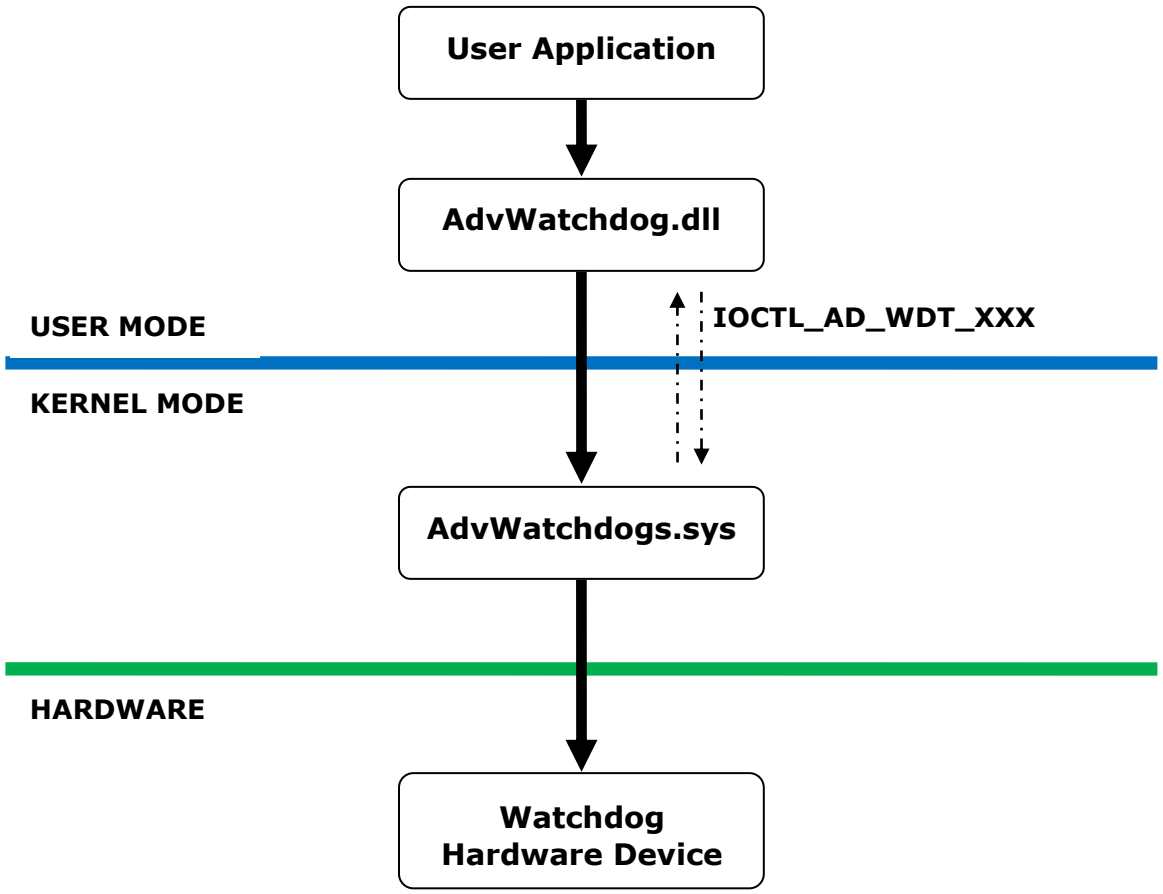


Figure 1

2.2 Watchdog Functions

2.2.1 Driver Functions

The Advantech Watchdog KMDF driver supports the following functions:

(1). Set the Watchdog timer span.

Below is a list of the timer span supported by the Advantech Watchdog KMDF driver. Depends on the Watchdog chipset, some Watchdog chipsets can only support timer span up to 4 Minutes 15 Seconds.

- ✧ 15 Seconds
- ✧ 45 Seconds
- ✧ 1 Minute 15 Seconds
- ✧ 2 Minutes 15 Seconds

- ✧ 3 Minutes 15 Seconds
- ✧ 4 Minutes 15 Seconds
- ✧ 5 Minutes 15 Seconds
- ✧ 10 Minutes 15 Seconds
- ✧ 20 Minutes 15 Seconds
- ✧ 30 Minutes 15 Seconds
- ✧ 40 Minutes 15 Seconds
- ✧ 50 Minutes 15 Seconds
- ✧ 1 Hour 15 Seconds
- ✧ 2 Hours 15 Seconds

(2). Get current Watchdog timer span

This operation retrieves current Watchdog timer span index and timer span value in minute-second.

(3). Set the Watchdog watch mode.

The Watchdog can be used to watch the whole system (System Mode) and a specified critical function call (Application Mode). If the Watchdog runs in the first mode, the OS (Watchdog Driver) will reset the Watchdog periodically. The machine will reboot if the system is hanged for a while. In the second mode, the specified critical function is responsible for resetting the Watchdog. The machine will reboot if the specified critical function call failed.

The Watchdog mode is designed for special industrial needs. On most occasions, the Watchdog is set to monitor the whole system. If the system is deadlocked then the Watchdog will reboot the machine. However, some specified applications are very important for the industrial solutions. If those applications failed then other works on the system will become nonsense. Therefore, monitor those applications is important. If they failed or crashed then the Watchdog should reboot the machine.

For example, in a network based DAQ solution, it is very important to make sure the connection between the client and the server is ready before any network actions start, if this connection failed and cannot be reconnected during a proper time span, the user application should reboot the machine. User can call the strobe Watchdog functions to verify the network connection is valid. This application is then under monitoring by the Watchdog.

(4). Get current Watchdog watch mode

This operation retrieves the current watch mode of the Watchdog, system-watch mode, or application-watch mode.

(5). Enable the Watchdog

Enable the Watchdog. If the Watchdog runs in application-watch mode without calling the strobe function during Watchdog timer span then the system will reboot.

(6). Disable the Watchdog

Disable the Watchdog.

(7). Strobe the Watchdog once

Strobe (reset) the Watchdog hardware once.

(8). Reboot the machine by Watchdog

If you issue this command, then you cannot recall this operation by disabling the Watchdog. The only way to prevent this computer from rebooting is to stop this service immediately.

2.2.2 Watchdog Timer Span

The following are timer spans for the Watchdog, most of chipsets only support up to 4 Minutes 15 Seconds.

(1) 15 Seconds
(2) 45 Seconds
(3) 1 Minute 15 Seconds
(4) 2 Minutes 15 Seconds
(5) 3 Minutes 15 Seconds
(6) 4 Minutes 15 Seconds
(7) 5 Minutes 15 Seconds
(8) 10 Minutes 15 Seconds
(9) 20 Minutes 15 Seconds
(10) 30 Minutes 15 Seconds
(11) 40 Minutes 15 Seconds
(12) 50 Minutes 15 Seconds

(13) 1 Hour 15 Seconds

(14) 2 Hours 15 Seconds

2.3 Environments

2.3.1 Hardware

This Software API and utility support only Advantech IAG x86 hardware platform products which with Watchdog design.

2.3.2 Operating Systems

☐ Microsoft Windows 2000

☐ Microsoft Windows XP Professional

☐ Microsoft Windows Vista

☐ Microsoft Windows 7

☐ Windows XP Embedded

☐ Windows Embedded Standard 2009

☐ Windows Embedded Standard 7

☐ Microsoft Windows 8

☐ Microsoft Windows 8.1

☐ Windows Embedded 8 Standard

☐ Windows Embedded 8.1 Industry

☐ Windows 10

2.4 Installation

2.4.1 Install KMDF Driver

Installation is required. If there is no existing installation of Advantech Watchdog KMDF driver on your computer, take the following steps to install Advantech Watchdog KMDF driver.

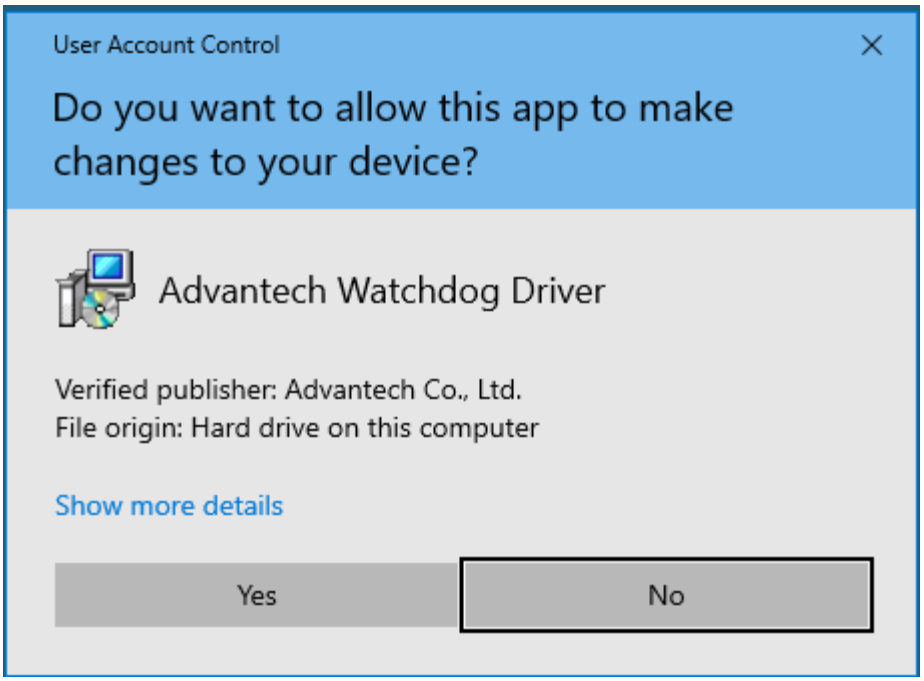
How to install Advantech Watchdog KMDF driver

1) Verify that your computer meets the hardware and software requirements to run Advantech Watchdog KMDF driver.
   For more information, see Environments.
2) If you do not already have the installer of Advantech Watchdog KMDF driver, download it from Advantech official Web site.
3) From Control Panel, remove any existing installation of Advantech Watchdog driver from your computer.
4) With administrator-level privilege on your computer, run the installer of Advantech Watchdog KMDF driver.

Below is an example of Advantech Watchdog KMDF driver Setup. If you want to stop the setup, press the "Cancel" button in the setup program. The Setup program will stop the procedure automatically.

Step 1: Run the Setup program. When the setup program is running, click the "Next >" button in "Advantech Watchdog Driver Setup Wizard".

Step 2: To continue the installation, click “Yes” and click Install to complete the driver installation.



Step 3: Click on "Restart" button or "Close" button on the "Advantech Watchdog Driver" to complete the setup program.

2.4.2 Installed files

- **SYS Driver Binary File**

  File Name: AdvWatchdogs.sys

  Advantech Watchdog KMDF Driver

- **DLL Binary File**

  File name: AdvWatchdog.dll

  Advantech Watchdog external export Library (API)

- **Control Panel Program**

  File name: AdvWatchdogConfig.cpl

  Advantech Watchdog Service configuration

- **Demo Program source code**

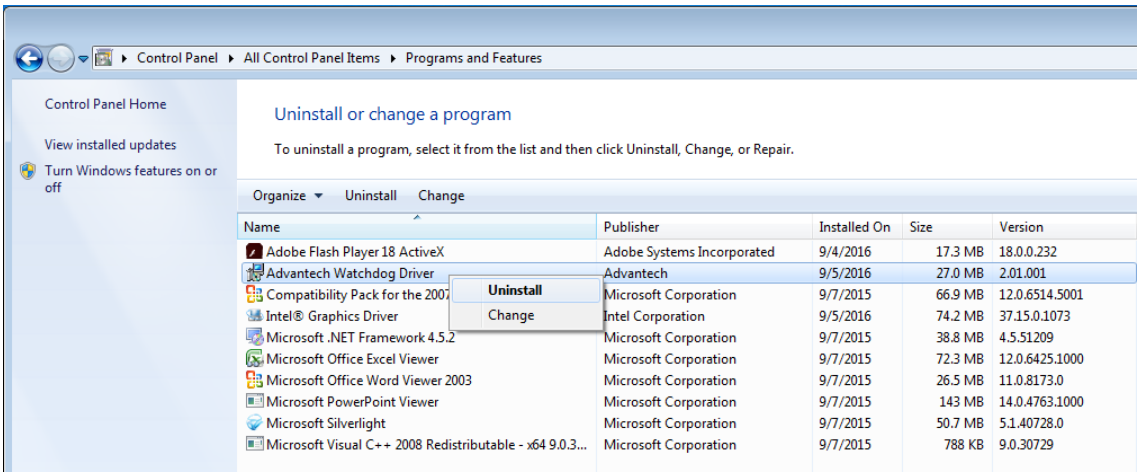
  File Patch: C:\Program Files\ADVANTECH\Watchdog\Example

  Advantech Watchdog program examples

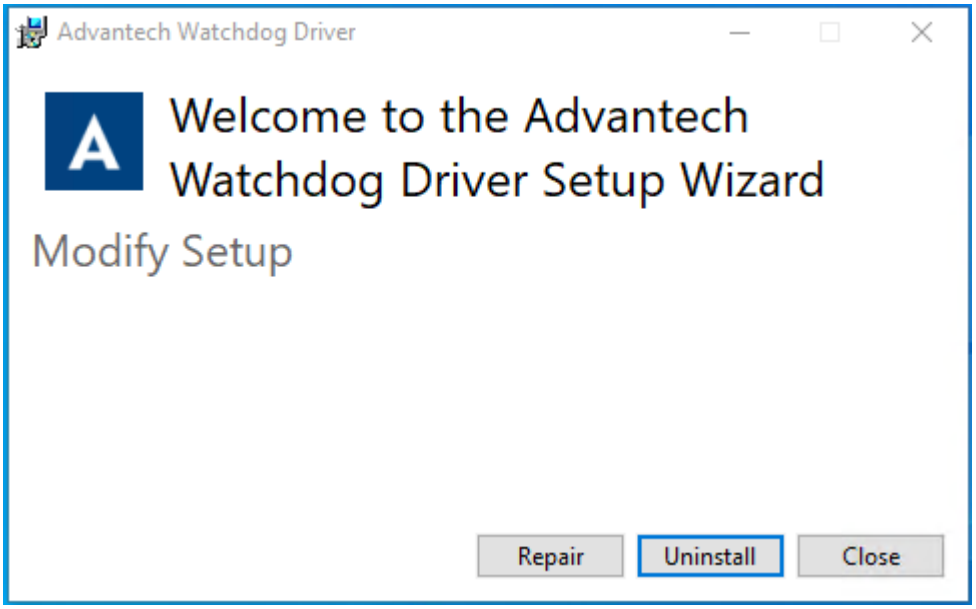
2.5 Uninstallation

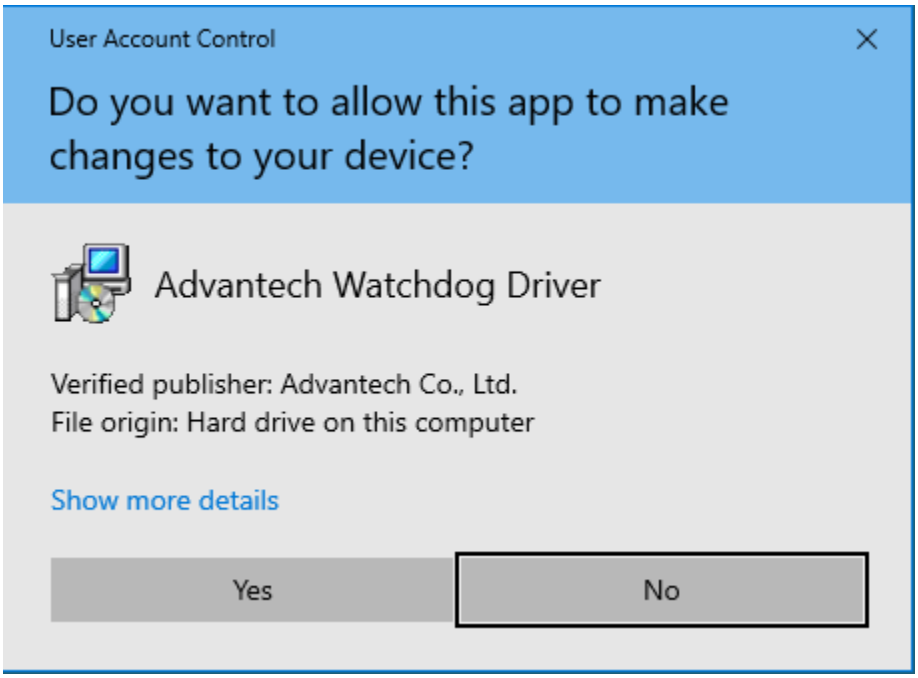
How to uninstall Advantech Watchdog KMDF driver

1. Control panel ->"Add or Remove Programs". Choose the "Advantech Watchdog Driver" to remove it.



2. Click on the "Uninstall" button. The setup program will start to remove the Advantech Watchdog KMDF Driver.





3. Click on the "Restart" or "close" button to complete the Uninstallation.

3. Control Panel Program

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

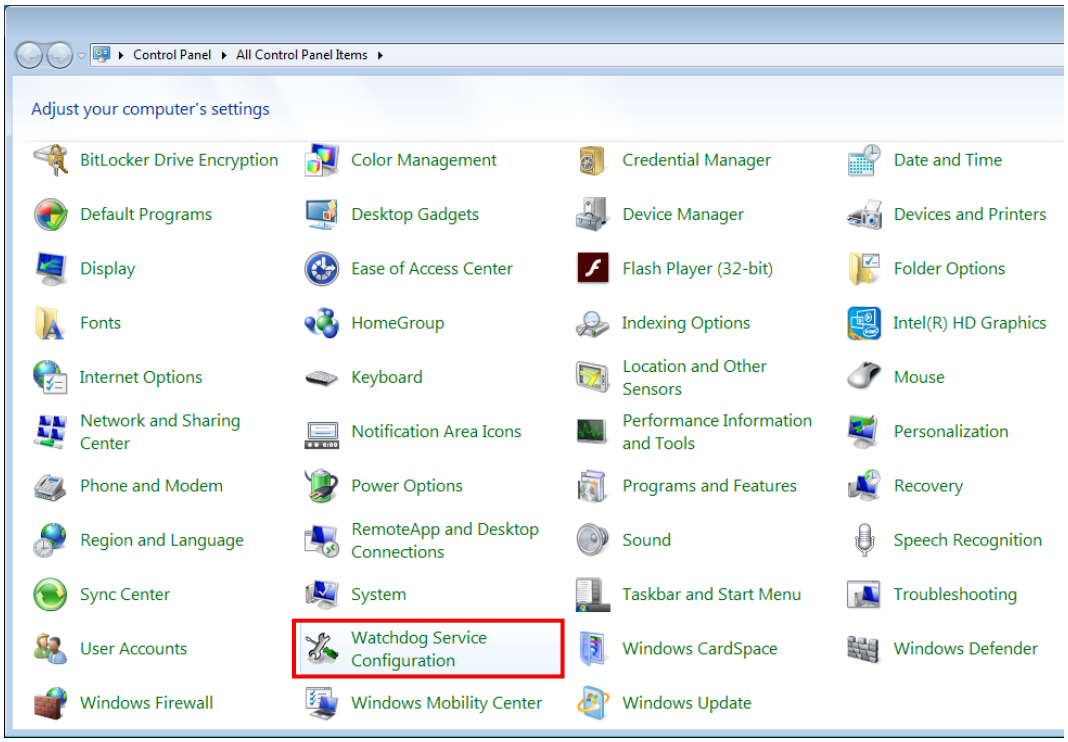


Figure 3.1-1

After completing the Watchdog Service Configuration in Control Panel, the settings will be written into the system registry. Later when your application requests a Watchdog operation, the Watchdog functions will read out the settings from the registry and do some actual Watchdog operations.

3.1 The Main Dialog

Double clicking on the icon "Watchdog Service Configuration" in Control Panel on your computer, you will see a popup dialog as shown in the figure below:

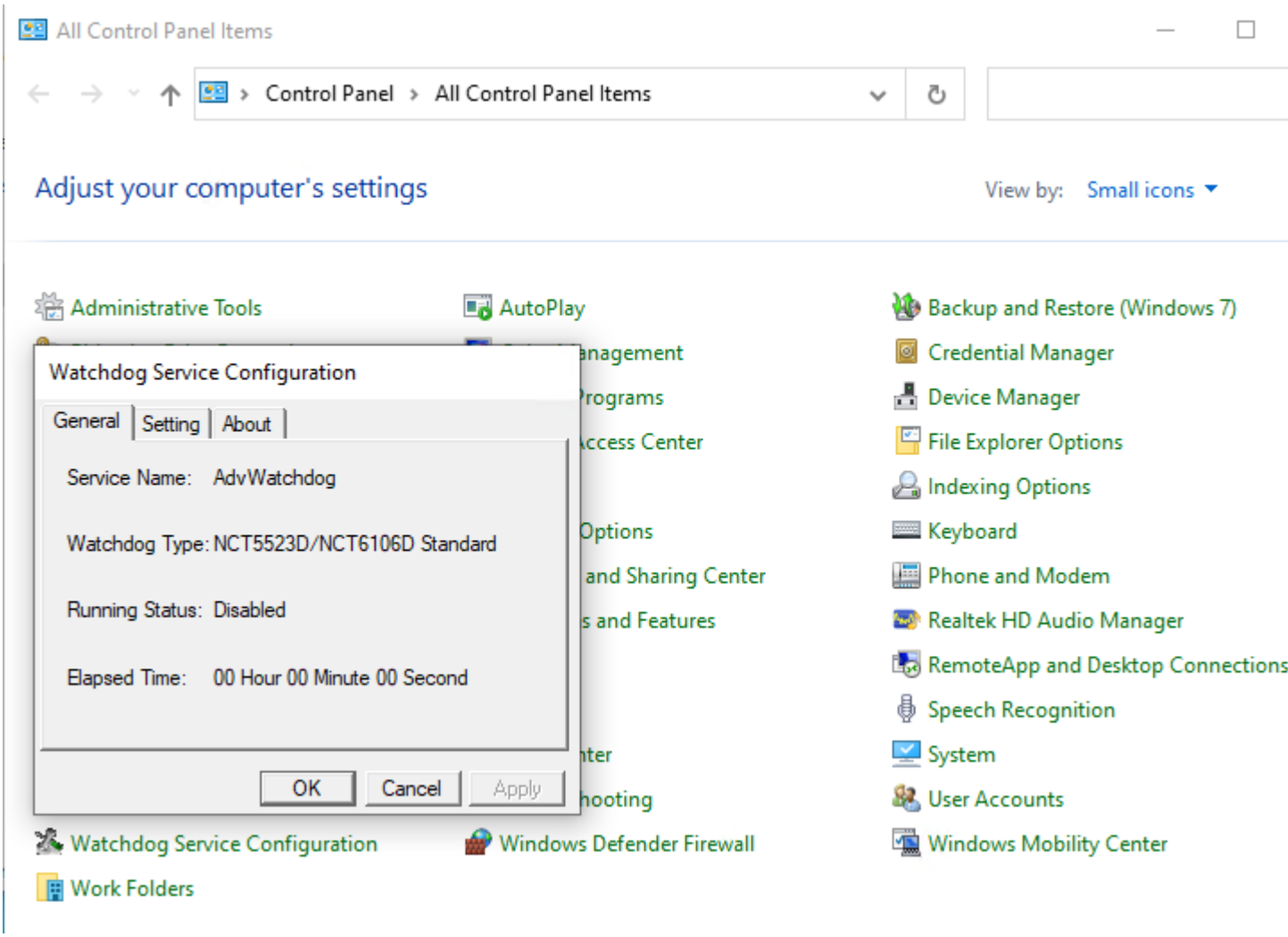


Figure 3.1-2

In the main dialog, there are three tab pages and three buttons:

(1). "General" tab page: Display some general information on the Advantech Watchdog service.

(2). "Setting" tab page: Display all the setting information on the Advantech Watchdog service.

(3). "About" tab page: Display the copyright information of the Advantech Watchdog service.

(4). "OK" button: Apply all the changes in the three tab pages to the Advantech Watchdog service and then close the main dialog.

(5). "Cancel" button: Discard all the changes in the three tab pages and then close the main dialog.

(6). "Apply" button: Apply all the changes in the three tab pages but do not close the main

dialog.

3.2 The "General" Tab Page

In the "General" Tab page, there are four static labels.
(1). Service Name: Display the name of the Advantech Watchdog service in the Service Control Manager (SCM) database.

(2). Watchdog Type: Display the Watchdog chipset type, such as NTC5523D (Figure 3.2-1)

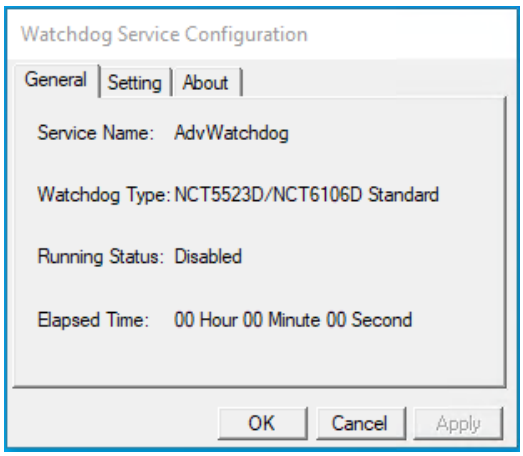


Figure 3.2-1

(3). Running Status: Display the Watchdog current status, enabled, disabled or resetting. Resetting means Watchdog chip will be reset after enabled in few seconds if you stop strobe in application mode or press "Hardware Reset" button in system mode.
(4). Elapsed time: It is the time elapsed from Watchdog startup or strobe till the current time. If the Watchdog is disabled, the elapsed time will be 00 hour 00 minute 00 second.

3.3 The "Setting" Tab Page

In this tab page, there are the following items:
(1). "Timer Span" combo box:
You can select one timer span for the Watchdog and apply the changes when the Watchdog

is disabled. There are 14 timer span arranges from 15 seconds to 2 hours 15 seconds.

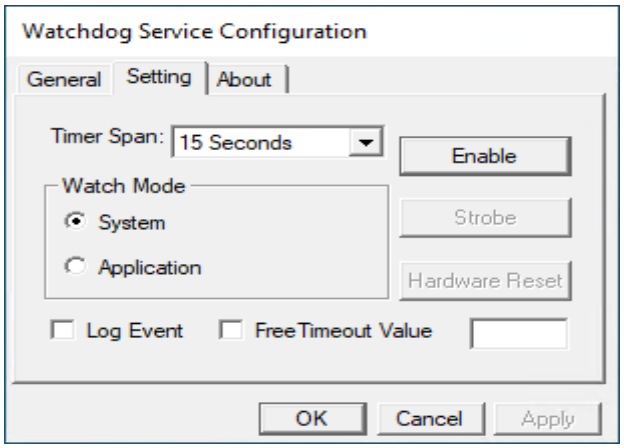


Figure 3.3-1

(2). "Watch Mode" radio box group:

There are two watch modes: System and Application.

If you select the "System" mode and apply settings by clicking on the "Apply" button in the main dialog, once you enable the Watchdog later, the Watchdog will watch the OS and run in Windows background.

If you select the "Application" mode and apply settings by clicking on the "Apply" button in the main dialog, once you enable the Watchdog later, it will pop up a warning message as figure 3.3-2 to notify you to strobe the Watchdog manually by left clicking on the "Strobe" button, otherwise the machine will reboot when the timer span expires.

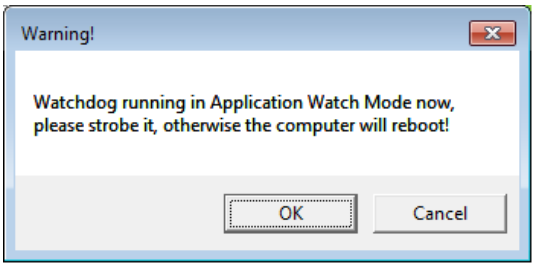


Figure 3.3-2

Definition:

System mode: Watchdog timer is running in Windows background. The OS (Watchdog Driver) will reset the Watchdog periodically. If the hardware hangs up, the Watchdog will time out and reboot system automatically.

Application mode: Watchdog timer is monitoring your specific application. Your application

is responsible for resetting the Watchdog. If your application hangs up, the Watchdog will time out and reboot system automatically.

(3). "Log Event" check box:

If this check box is checked and the settings are applied by clicking on the "Apply" button of the main dialog then the "Enabled", "Disable", "Reboot" operations of the Watchdog will be logged into the system event base, otherwise the three operations will not be logged into the system event base.

- **Log Operation Information into System Event Base**

  You can view the logged operations in the system base by running the event viewer named "eventvwr.msc". Select the items with the source name "AdvWatchdog" (Figure 3.3-3) and double click on them to open the detailed event messages.

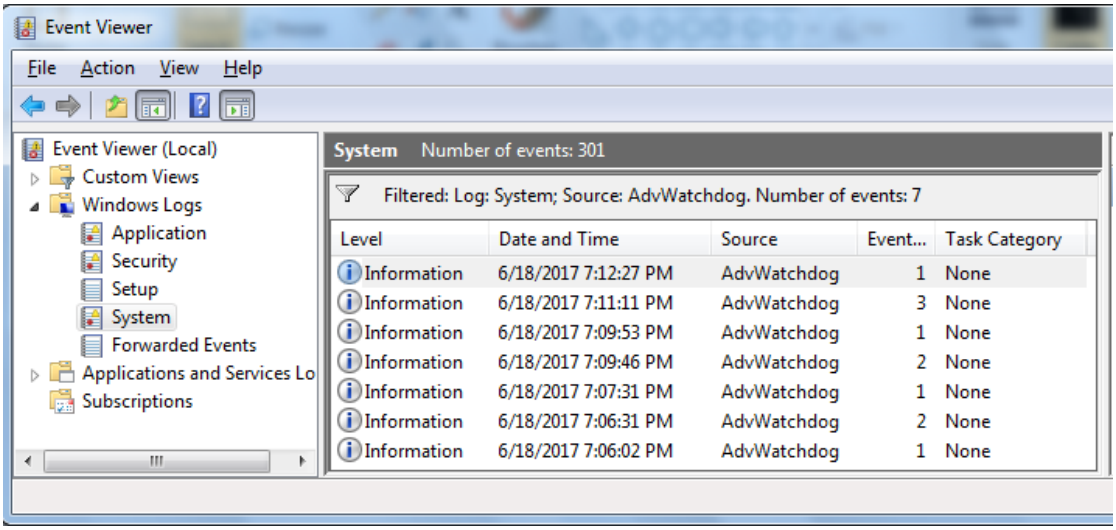


Figure 3.3-3

"Enable" the Watchdog will log a message "Advantech Watchdog Now Enabled" into the system event base as figure 3.3-4.

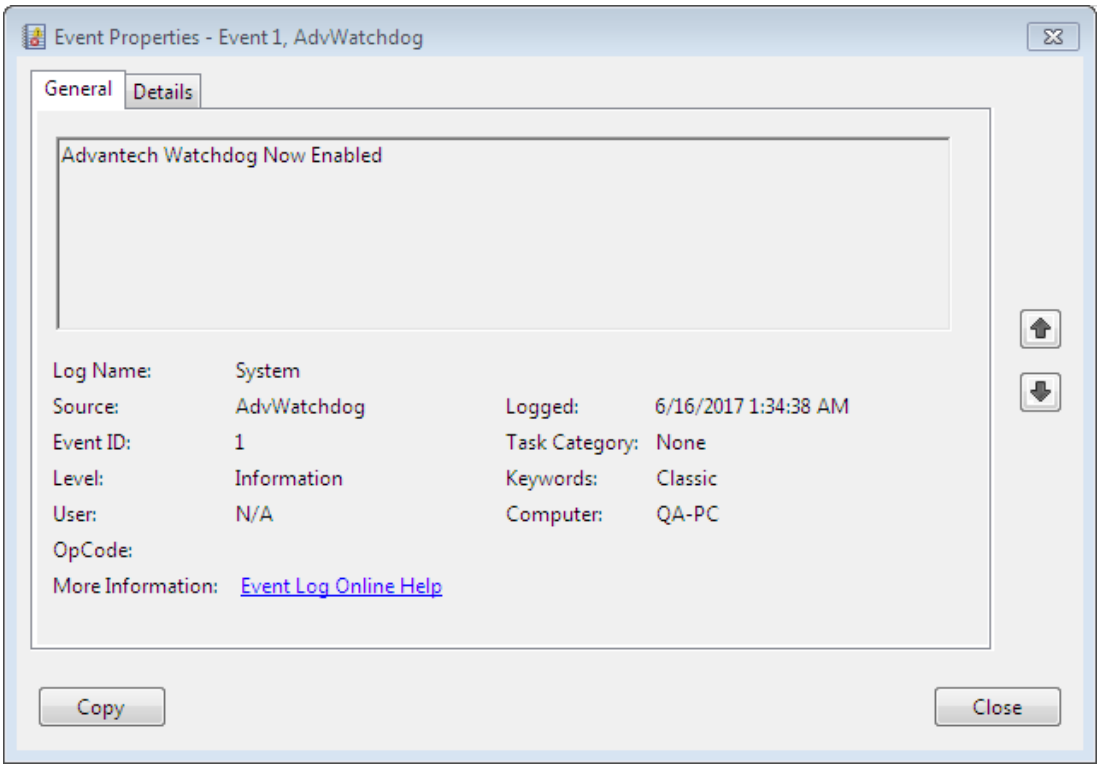


Figure 3.3-4

"Disable" the Watchdog will log a message "Advantech Watchdog Now Disabled" into the system event base as figure 3.3-5.

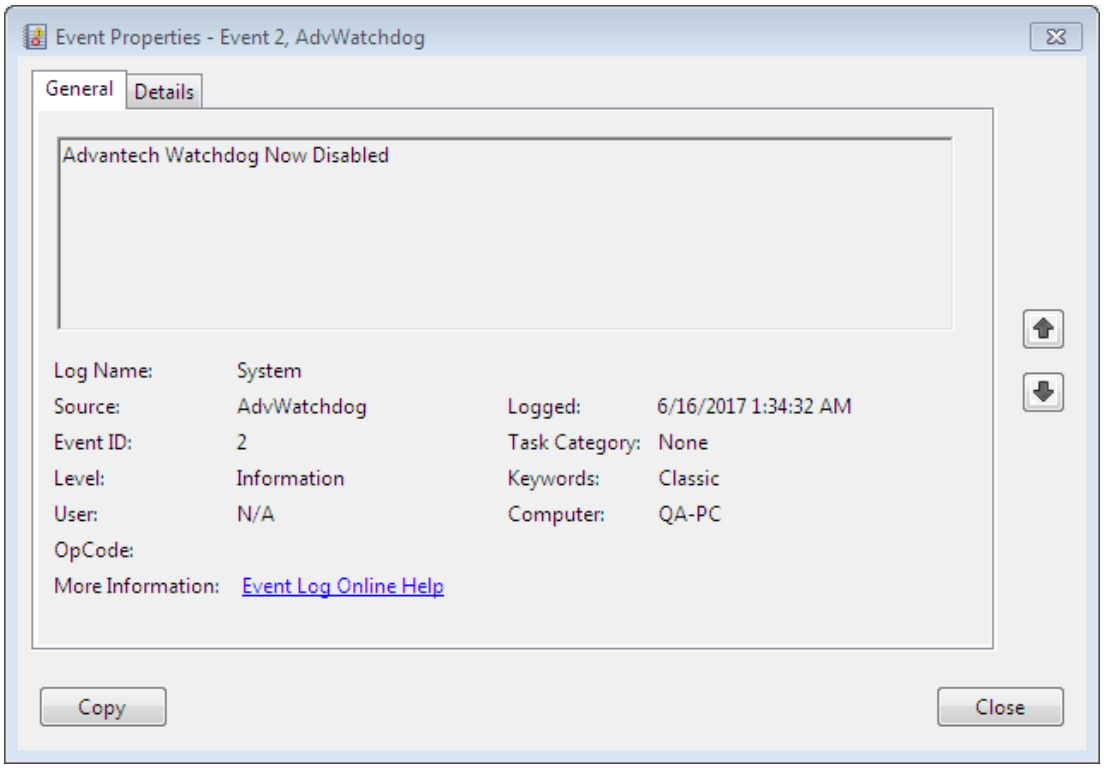

Event Properties - Event 2, AdvWatchdog
General
Details
Advantech Watchdog Now Disabled
Log Name: System
Source: AdvWatchdog
Logged: 6/16/2017 1:34:32 AM
Event ID: 2
Task Category: None
Level: Information
Keywords: Classic
User: N/A
Computer: QA-PC
OpCode:
More Information: Event Log Online Help
Copy
Close

Figure 3.3-5

"Reboot" the machine by the Watchdog will log a message "Advantech Watchdog Now Rebooting the Machine" into the system event base as figure 3.3-6.

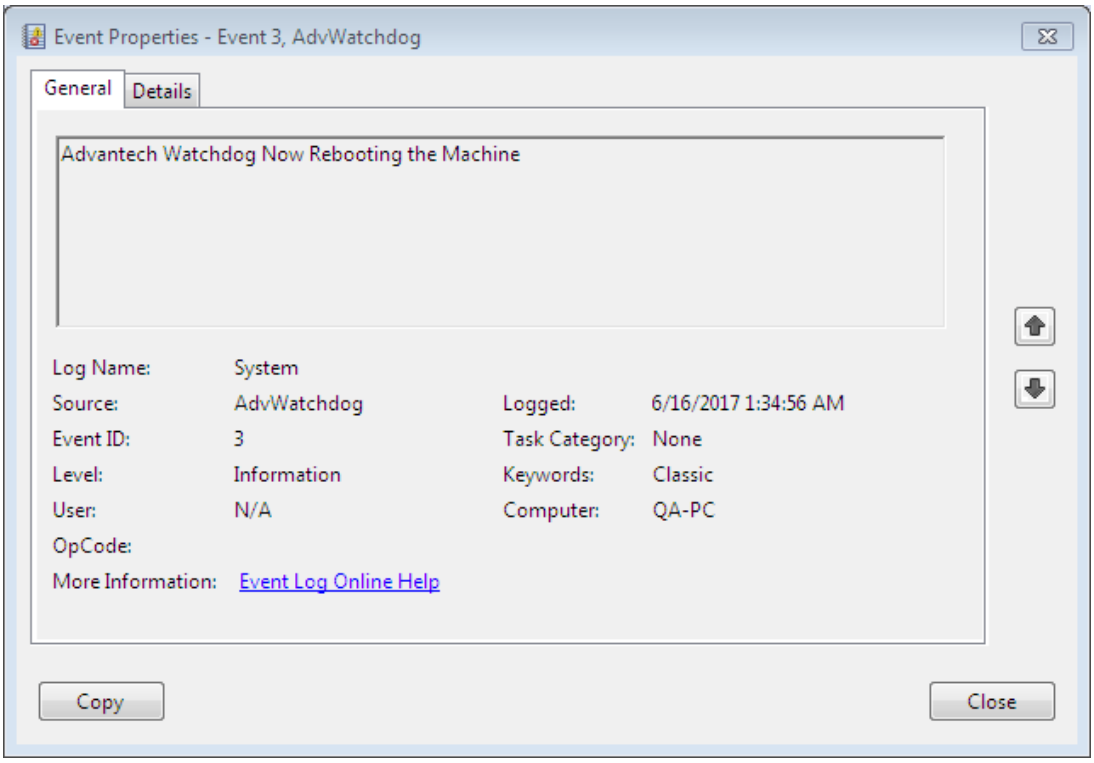


Figure 3.3-6

(4). "Enable/Disable" button:

Enable or disable the Watchdog. If the Watchdog is enabled, you cannot change the watch mode and the timer span of the Watchdog, so these related controls become grayed. These controls resume to their normal status when the Watchdog becomes disabled.

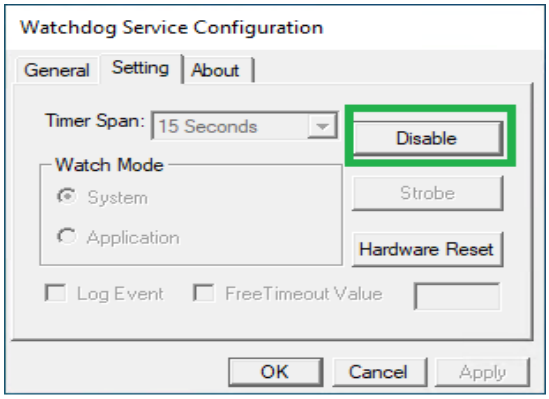


Figure 3.3-7

(5). "Strobe" button:

Strobe the Watchdog. This button becomes available only when the Watchdog runs in application-watch mode and the Watchdog is enabled.

(6). "Hardware Reset" button:

Reset the machine without strobing the Watchdog hardware. This button is not available when the Watchdog is disabled. If the Watchdog is enabled and you click on this "Hardware Reset" button then all the three buttons: "Enable/Disable", "Strobe", and "Reset" become grayed. No operations can cancel the resetting machine operation unless stopping the Advantech Watchdog service.

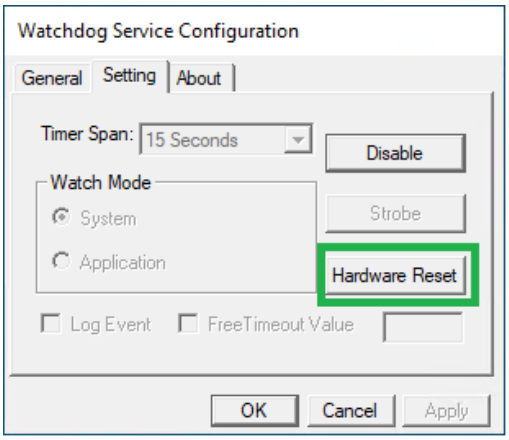


Figure 3.3-8

3.4 The "About" Tab Page

This page displays some copyright information of the Advantech Watchdog service.

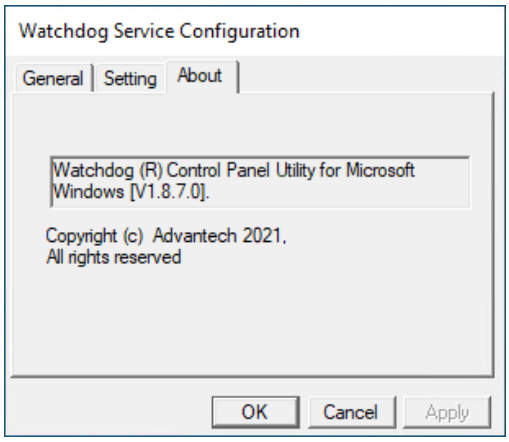


Figure 3.4-1

4. Getting Started with Advantech Watchdog KMDF Driver

This chapter provides a step-by-step instruction to demonstrate how to build an application using Device Driver from scratch in Microsoft Visual C++.

The 32/64bit Windows 2000/XP/Vista/7/8/8.1/10/Windows Embedded device driver is a set of dynamic-link library.

The following is the list of the necessary files for programming:

File Description and Location

- AdvWatchdog.h: Function declaration, constant definition for Microsoft Visual C++ 6.0.
  <Your installation path>\Watchdog\Include\AdvWatchdog.h

- AdvWatchdog.lib: Library files for C/ C++.
  <Your installation path>\Watchdog\Lib\AdvWatchdog.lib

4.1 For Microsoft Visual C++

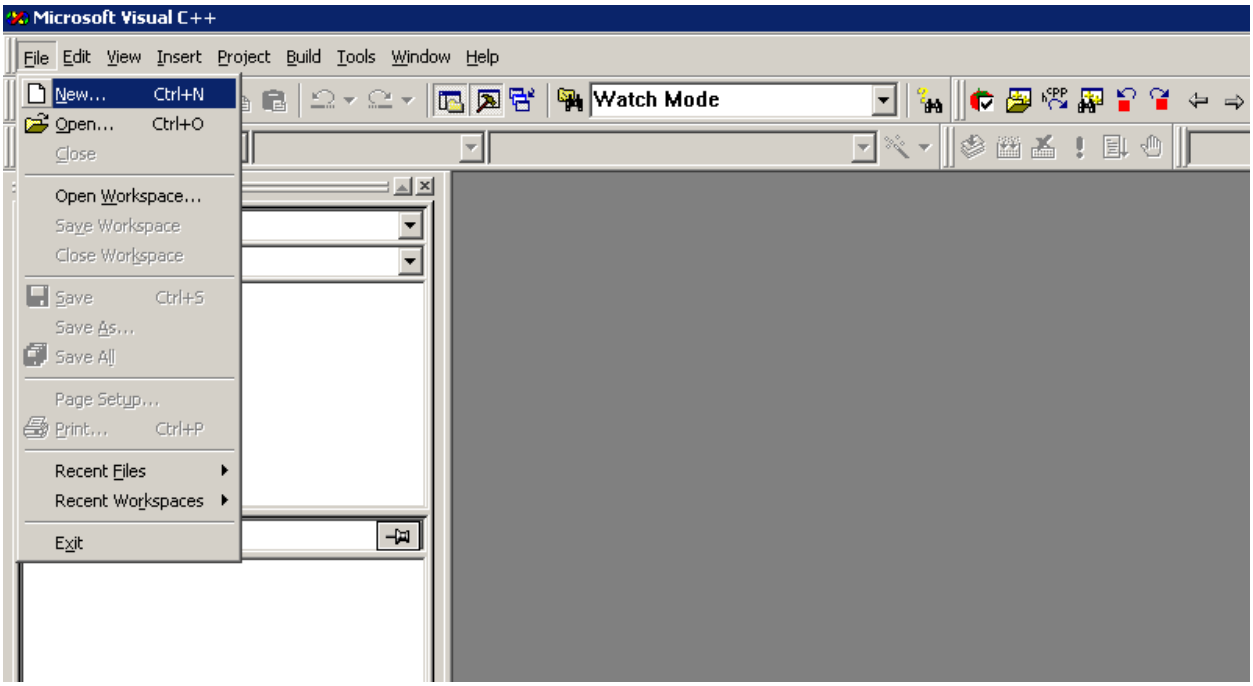
4.1.1 Create an Empty Visual C++ Project

To use the Watchdog functions, you must use the DLL routines. Follow this procedure:
1. Create your source files as you would for other Windows programs written in C++ by calling DLL functions as typical function calls.
2. Include the DLL header file, as shown in the following example:
#include "AdvWatchdog.h"
(Installation C:\Program Files\Advantech\Watchdog\Include\AdvWatchdog.h)
3. Import library: "AdvWatchdog.lib"
(Installation C:\Program Files\Advantech\Watchdog\Lib\AdvWatchdog.lib) to the project module.

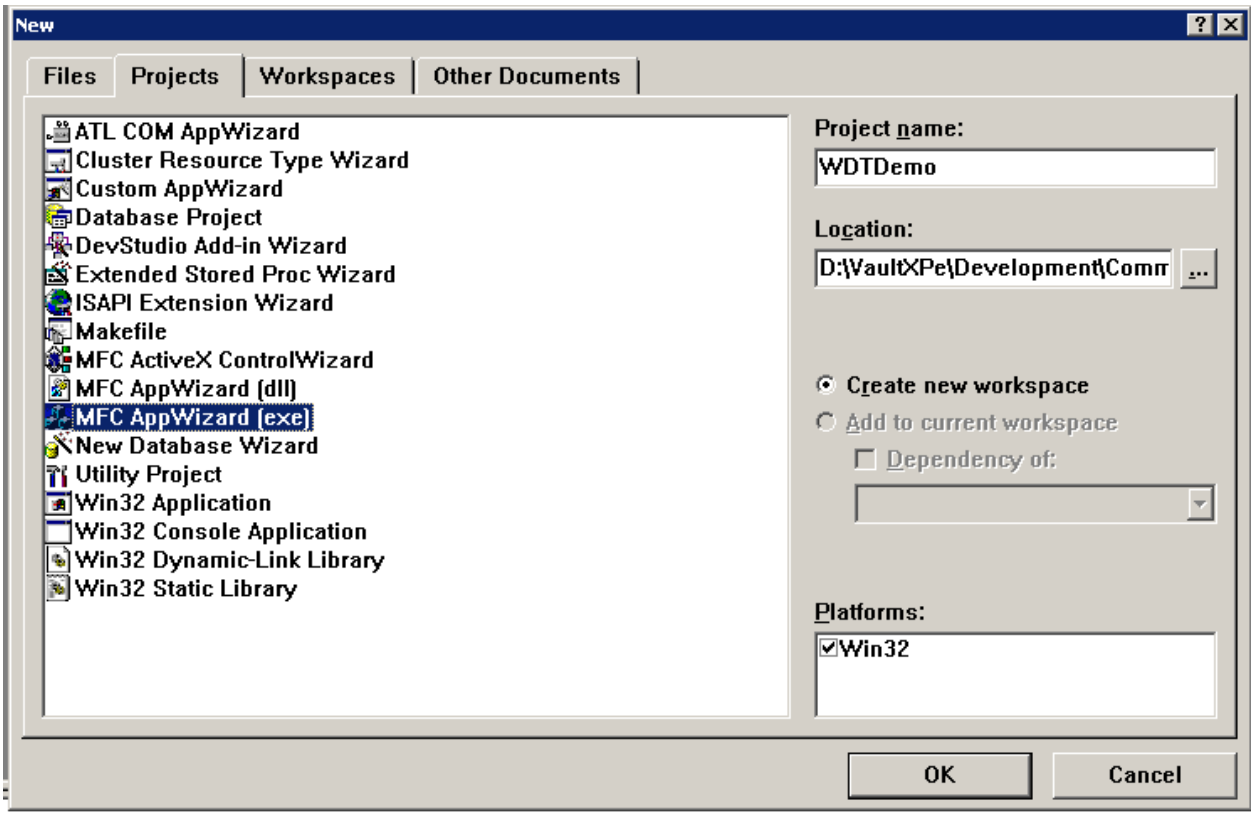
For a general outline of creating a Visual C++ Windows programs, complete the following

procedure:

1. Click **File/New** from the main menu to create your application project and source code as you would for any other Visual C++ program.



2. Define the type of new project as "MFC AppWizard (exe)" and assign a project file directory

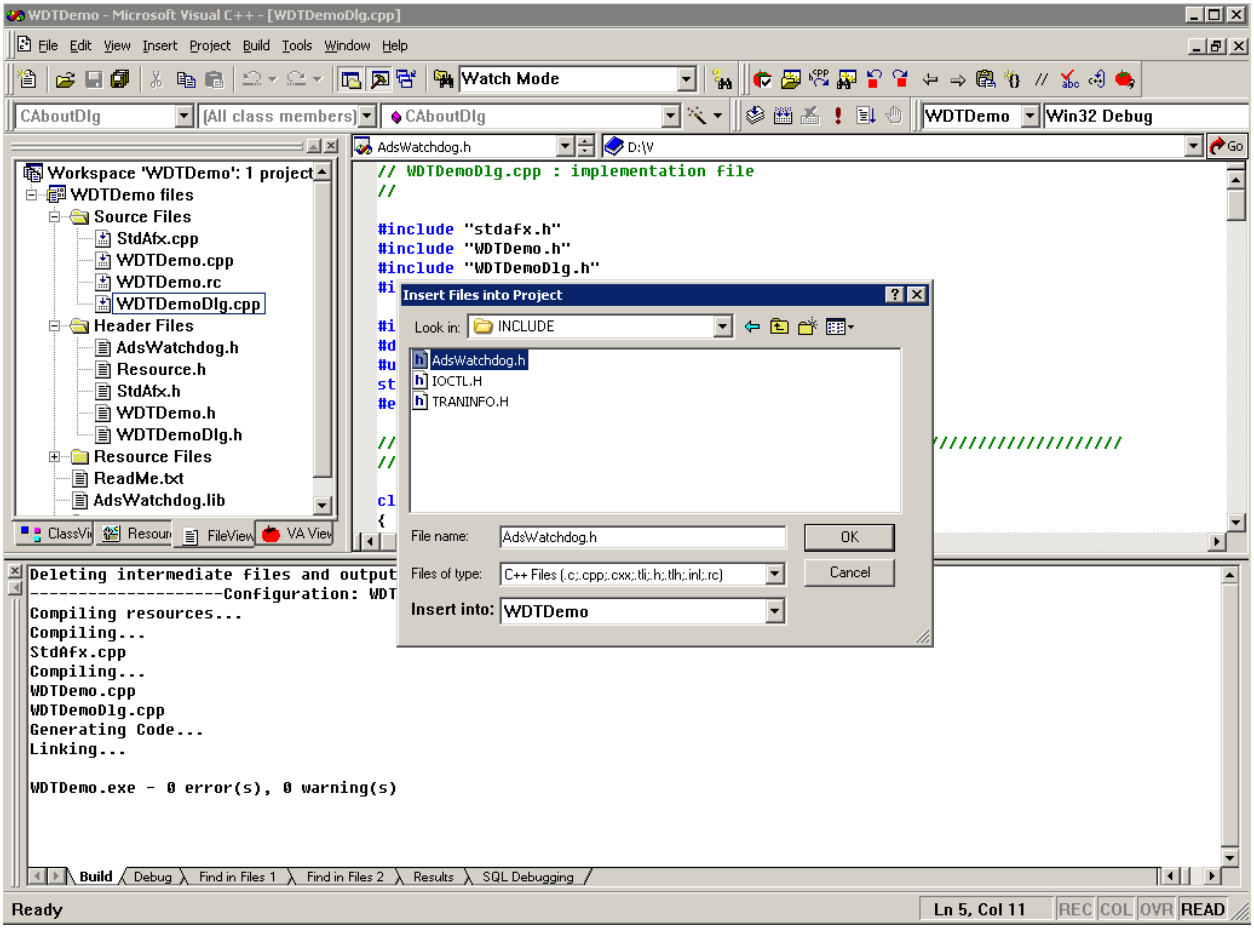


Run through the wizard to create the new project from Empty.

4.1.2 Adding Necessary Files

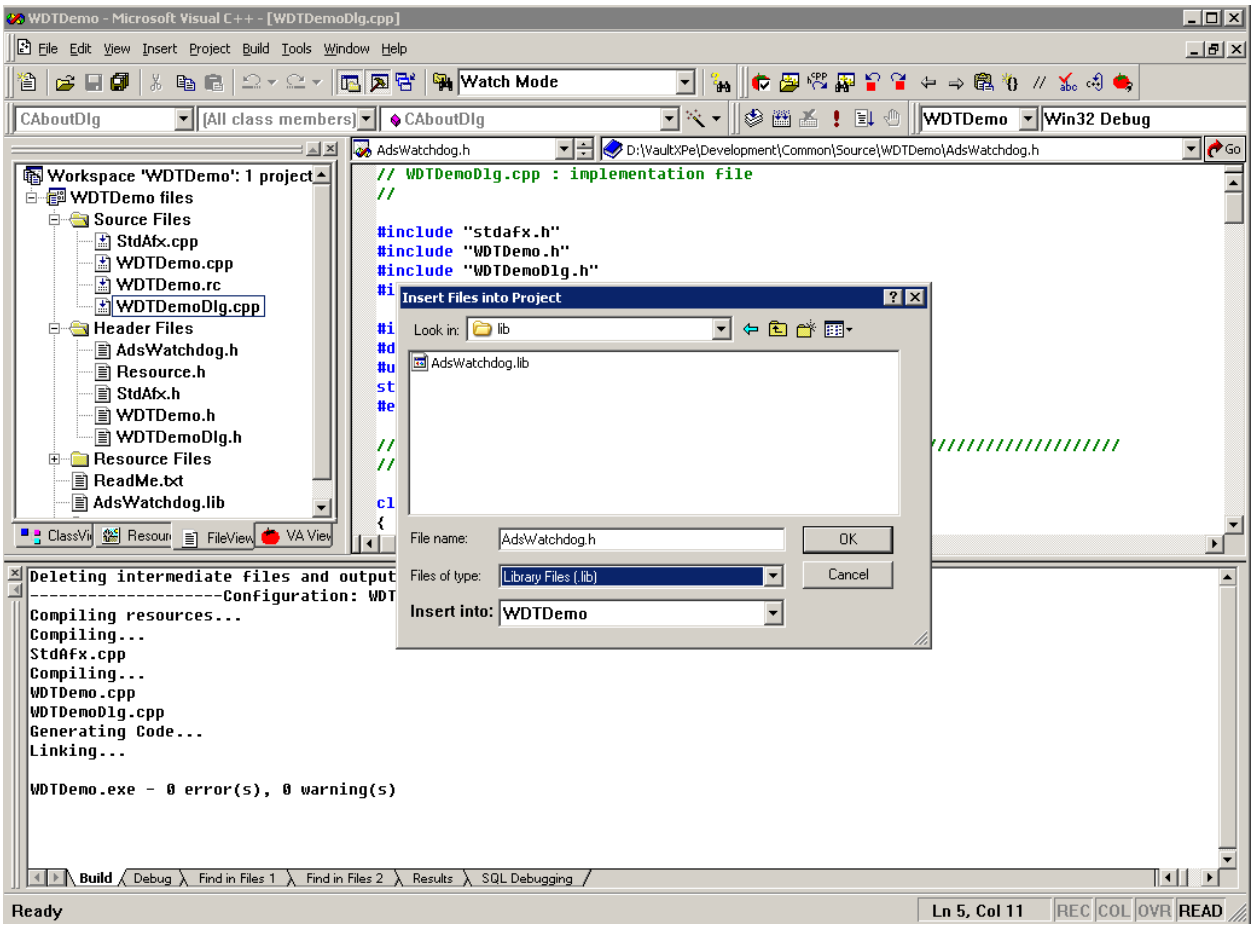
In order to develop Watchdog applications with Advantech Watchdog KMDF Driver, you have to add necessary files first.

1. Include the Advantech Watchdog KMDF Driver for Visual C++ header files (AdvWatchdog.h). The way to include the header file into your project is to right click and select Add Files to Folder from the Visual C++ main menu.



After adding the header file, you can view the Watchdog constant definition, parameter declaration, and DLL function calls that are defined in this header file. These definitions can all be used in your application programs.

2. Insert AdvWatchdog.lib in the same way. The Device Driver library will be linked with your application object file and be built into an application execution file through the Build menu.



4.1.3 Writing Codes

Write your application source code. For more detailed program development information, please refer to the Visual C++ User's Manual.

4.1.4 Test Your Program

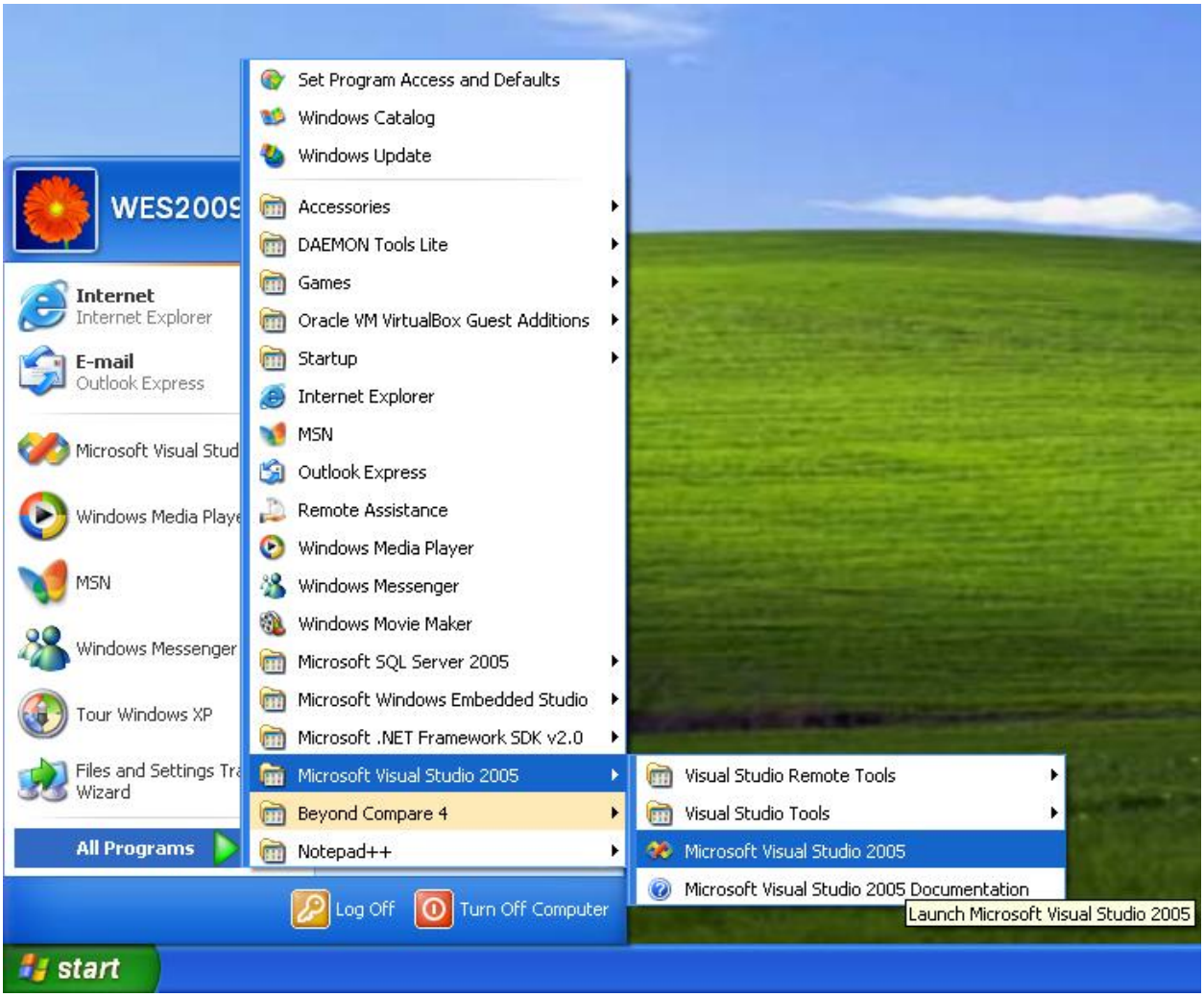
1. Click on Compile under the Build menu to compile your code.

2. Run your saved ***.exe on you target platform.

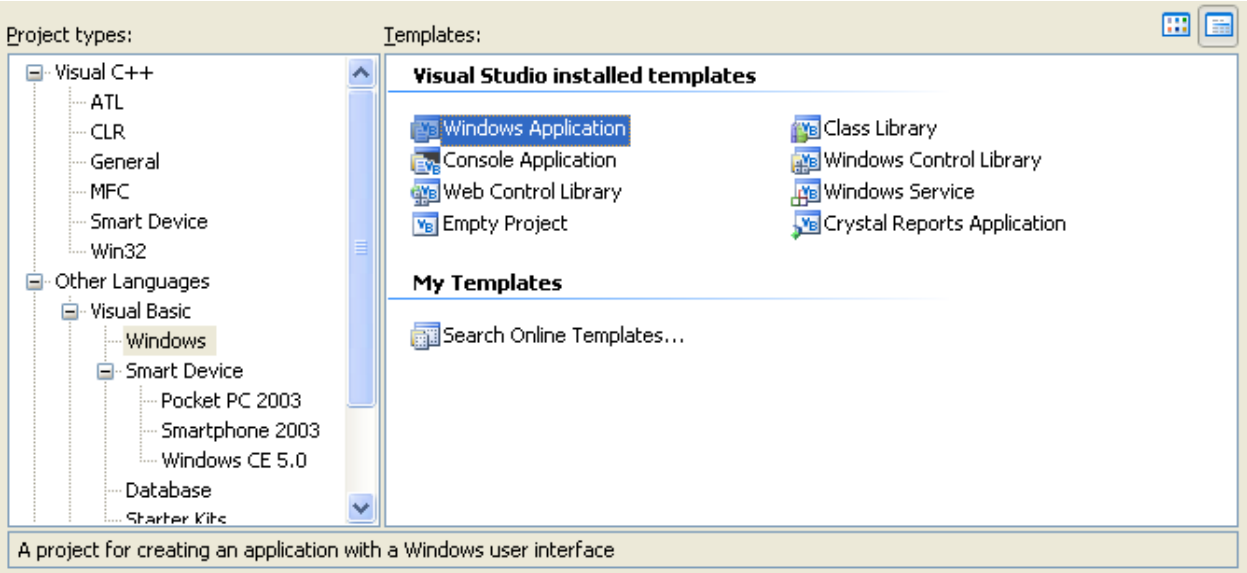
4.2 For Microsoft Visual Basic

4.2.1 Create an Empty Visual Basic Application

1. Go into the Start menu and click on Microsoft Visual Studio 2005.

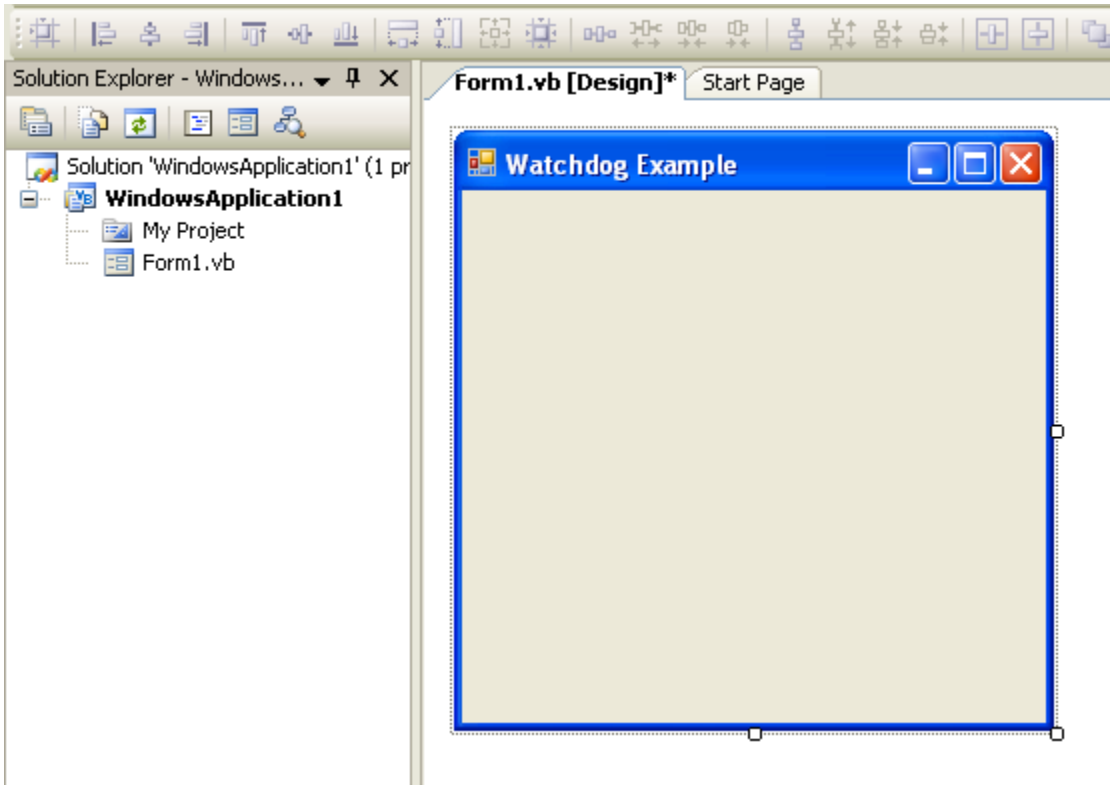


2. The Visual Basic .NET development environment will be loaded as follows:

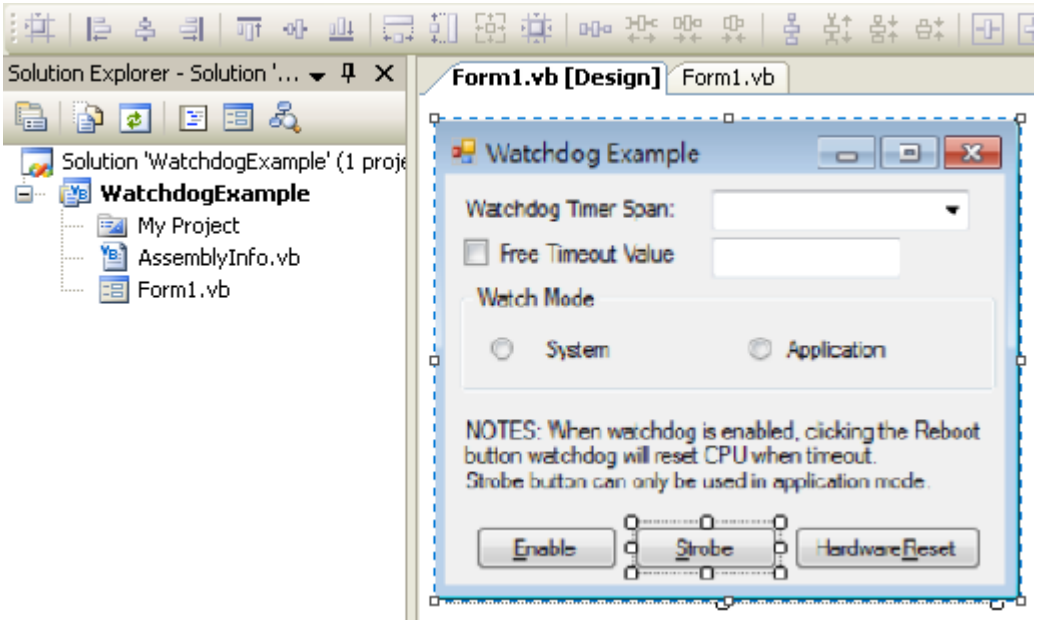


3. Select the **Windows Application** icon and press the **Open** button. A new project is created.

4.2.2 Adding Files and Designing the Form



Design your form. Your form should look similar to the one shown below:



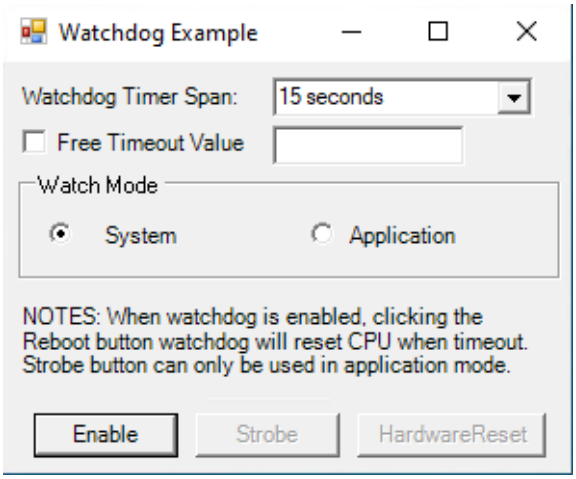
4.2.3 Writing Codes for VB Application

Write your application source code. For more detailed program development information, please refer to the Visual Basic User's Manual.

4.2.4 Test Your Program

1. Click on Compile under the Build menu to compile your code.

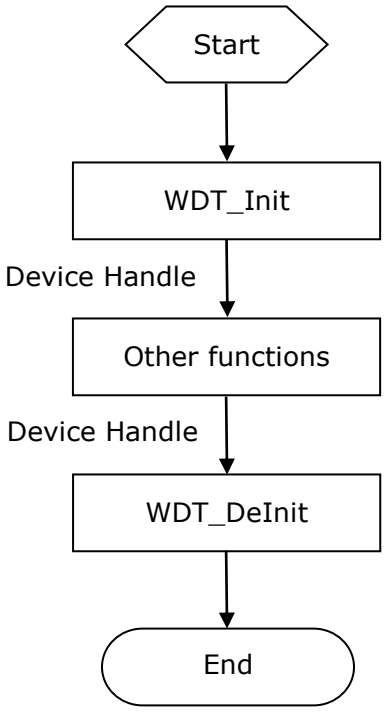
2. Run your saved ***.exe on you target platform.



5. Programming Guide

Device Function Group

The following figure describes the common function call flow of the Watchdog which is necessary for all WDT operations:



- **WDT_Init and WDT_DeInit Functions**

WDT_Init initializes the device. This function must be called before any other method that performs WDT operations. **WDT_DeInit** is the counterpart function of the **WDT_Init** function to close the device.

- **Device Handle (Type: Long, Size: 4 bytes)**
  Device Handle is returned by **WDT_Init**. The subsequent function calls use Device Handle to represent the desired device.

- **Error code and WDT_GetErrMsg**
  Each driver function returns an Error Code that indicates whether the function was performed successfully. When a function returns a code that is neither 0 nor 500, it means that the function failed to perform. You can pass the error code to the WDT_GetErrMsg function to retrieve its error message.

- **Other Device Functions**
  WDT_SetMode/WDT_SetTimerSpan/WDT_LogEvent/WDT_SetType
  These functions configure the Watchdog timer.

  WDT_GetMode/WDT_GetTimerSpan/WDT_IsEnabled/WDT_IsLogged/WDT_IsReadyToReboot/WDT_GetStartTime/WDT_GetType/WDT_GetTimerSpanDescription
  These functions retrieve the device-specific information about the current configuration and state of the Watchdog.

  WDT_Enable
  This function Enable the Watchdog function.

  WDT_Disable
  This function Disable the Watchdog function.

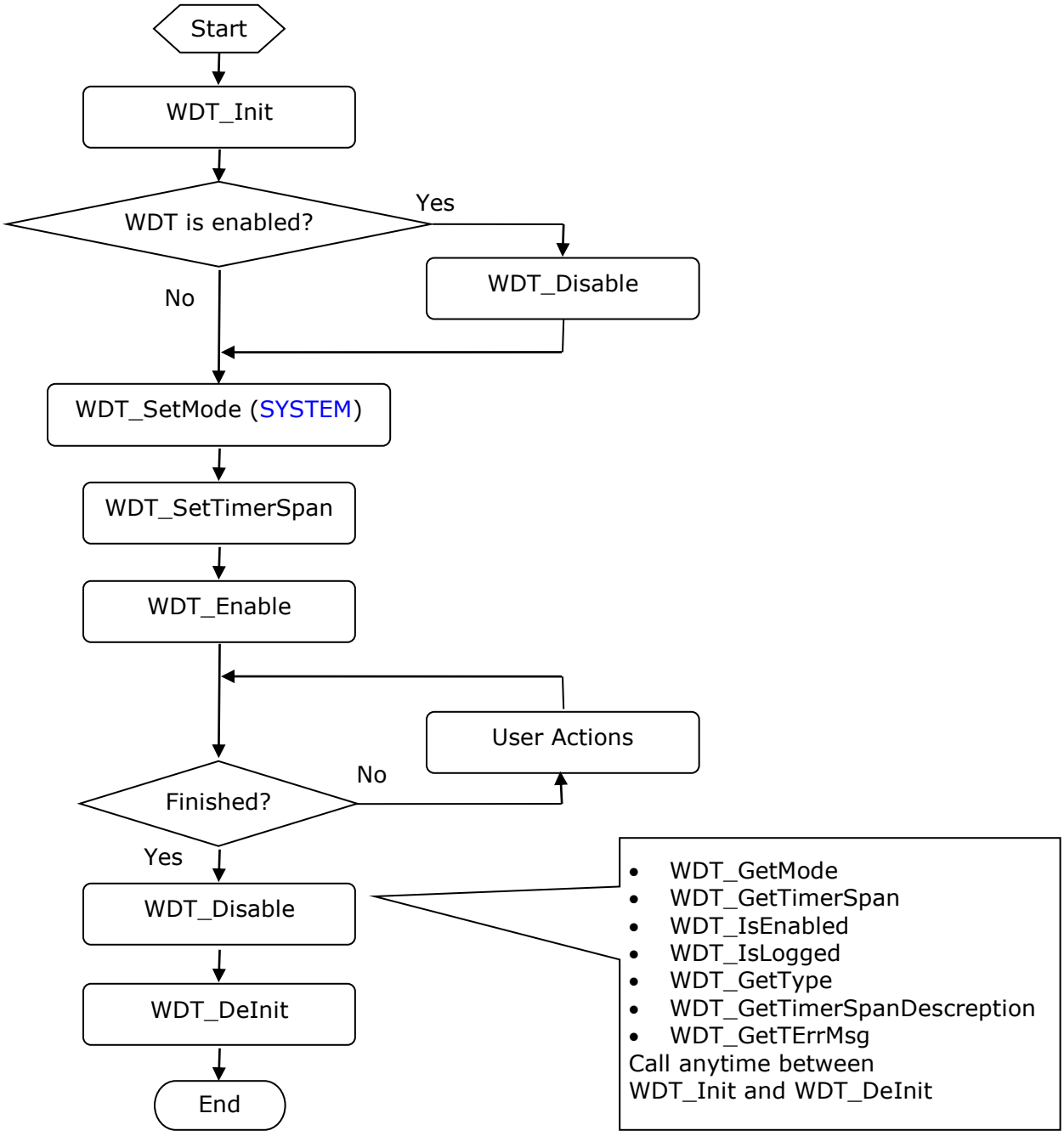
  WDT_Strobe
  This function reset the Watchdog.

  WDT_Reboot
  This function stops to feed the Watchdog. It will reboot the machine by the Watchdog.

5.1 Function Call Procedures in System Watch Mode

When the Watchdog be used in System Mode, which is when a single timeout period is set for the WDT; the Watchdog driver monitors the whole system. The WDT driver will perform **WDT reset** periodically. CPU hangs or OS kernel crashes will cause Watchdog timer to time out; therefore a CPU reset will be issued when Watchdog timeout reaches final level.

The following figure describes the common function call flow of the Advantech Watchdog which is working in System Mode:

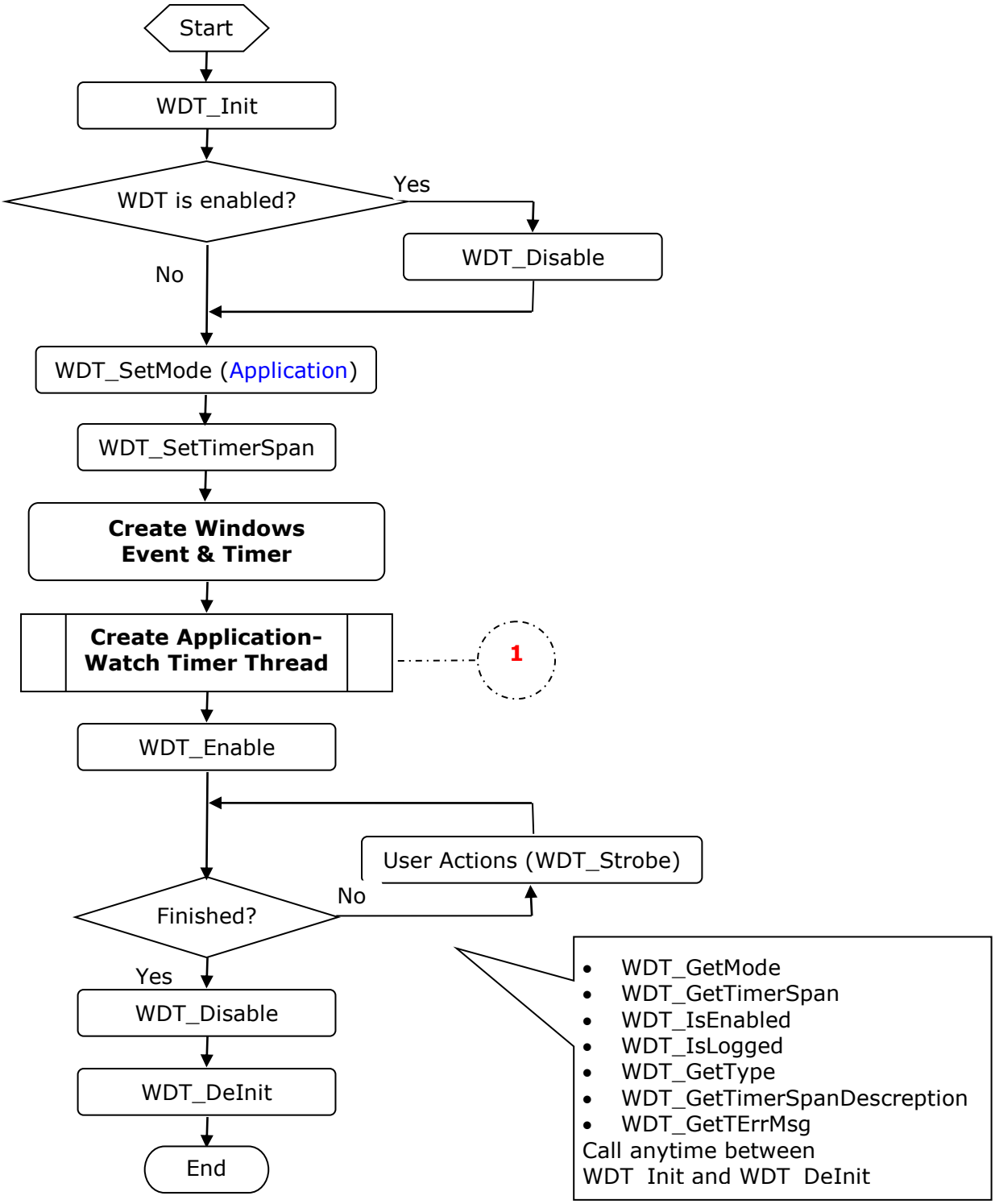


Function Call Procedures in System Mode

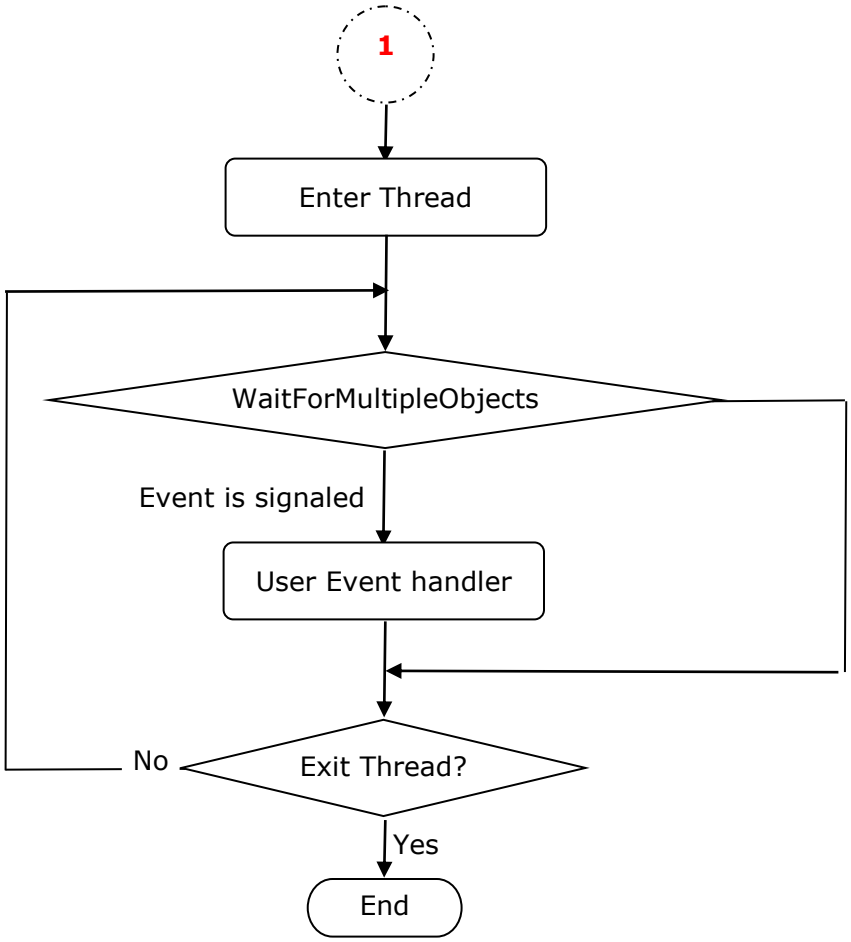
5.2 Function Call Procedures in Application Watch Mode

When the Watchdog is used in the Application Mode, it means application takes control for Watchdog timer reset (strobe). Event will be issued when Watchdog timeout reaches different level. You can execute the WDT event handler, instead of resetting the system. In WDT Application Mode, you should supply your thread to feed the dog; you can stop or strobe the WDT functions at any time.

The following figure describes the common function call flow of the Advantech Watchdog which is working in Application Mode:



Function Call Procedures in Application Mode



Function Call Procedures in Application Thread

6. Function Reference

Advantech Watchdog Driver contains a set of functions and associated structures that can be used in various applications. The APIs support many development environments and programming languages, including Microsoft Visual C++. Installing the driver is necessary to successfully use Advantech Watchdog. This documentation describes our driver's application programming interface (API).

6.1 Function Description

You should manipulate Watchdog through the functions from AdvWatchdog.dll, thus you can use the Watchdog device through your existing applications and examples without any change.

In user applications, call **WDT_SetMode/WDT_SetTimerSpan/WDT_LogEvent/WDT_SetType** with specific timeout values to start the Watchdog timer countdown. Meanwhile, create a thread or timer to periodically refresh the timer with **WDT_Strobe** before it expires. If the application ever hangs, it will fail to refresh the timer and the Watchdog reset will cause system to reboot.

The following table describes all the user interfaces the driver supports.

Item	Name	Note
1)	WDT_Init	Initialize the Watchdog
2)	WDT_DeInit	De-Initialize the Watchdog
3)	WDT_Enable	Enable the Watchdog
4)	WDT_Disable	Disable the Watchdog
5)	WDT_SetMode	Set the Watchdog watch mode
6)	WDT_GetMode	Get the Watchdog watch mode
7)	WDT_SetTimerSpan	Set the Watchdog timer span index
8)	WDT_GetTimerSpan	Get the Watchdog timer span index
9)	WDT_Reboot	Reboot the machine by the Watchdog
10)	WDT_IsEnabled	Get the Watchdog current running status: Enabled or Disabled.
11)	WDT_LogEvent	Set the Watchdog operations: Enable, Disable and

		Reboot to be logged into the system event base or not.
12	WDT_IsLogged	Get the Watchdog event log information: The “Enable”, “Disable”, “Reboot” operation logged into system event base or not.
13	WDT_IsReadyToReboot	Get the Watchdog current reboot status.
14	WDT_GetStartTime	Get the Watchdog enabled time.
15	WDT_Strobe	Strobe the Watchdog once
16	WDT_SetType	Set the Watchdog type. In current version this function is reserved for further extension and no implementation is available
17	WDT_GetType	Get the Watchdog chip type.
18	WDT_GetTimerSpanDescription	Get the description of the specified timer span index.
19	WDT_GetErrMsg	Get the error message of a specified error code.
20	WDT_SetFreeTimeoutValue	Support FreeTimeout range for EC chip type
21	WDT_GetWDTConfig	Get Max and min of timeout value in specific chiptype for WDT and Get FreeTimeOut Value

6.1.1 WDT_Init

LONG status = WDT_Init (LONG * DriverHandle)

Purpose

Initialize the Watchdog.

Retrieves parameters pertaining to the device's operation from the Registry or configuration file, and allocates memory to store it for quick reference. This function must be called before any other functions.

Parameters

Name	Direction	Type	Rage	Description
DriverHandle	Output	long pointer	default	Handle of the Watchdog driver

Return

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. MemoryAllocateFailed if memory allocation failure.

3. CreateFileFailed if fail to open low level driver.

6.1.2 WDT_DeInit

LONG status = WDT_DeInit (LONG * DriverHandle)

Purpose

De-initialize the Watchdog.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	long	default	Handle of the Watchdog. Assigned by WDT_Init

Return

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. INVALIDDEVICEHANDLE if DriverHandle is NULL.

6.1.3 WDT_Enable

LONG status = WDT_Enable (LONG DriverHandle)

Purpose

Enable the Watchdog function.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input / Output	long pointer	default	IN, Handle of the Watchdog driver. Assigned by WDT_Init

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

3. ADS_WATCHDOG_ERROR_WDT_REBOOTTING if the Watchdog is Rebooting.

6.1.4 WDT_Disable

LONG status = WDT_Disable (LONG DriverHandle);

Purpose

Disable the Watchdog function.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input / Output	long pointer	default	IN, Handle of the Watchdog driver. Assigned by WDT_Init

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

3. ADS_WATCHDOG_ERROR_WDT_REBOOTTING if the Watchdog is Rebooting.

6.1.5 WDT_SetMode

LONG status = WDT_SetMode(LONG DriverHandle, WatchMode i_WatchMode);

Purpose

Set the watch mode of the Watchdog.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	long	default	IN, Driver handle. Assigned by WDT_Init
WatchMode	Input	WatchMode	WATCH_MODE_SYSTEM /	IN, the WDT working mode.

			WATCH_MODE_APPLICATION	WATCH_MODE_SYSTEM: WDT System Reset Mode. WATCH_MODE_APPLICATION: WDT Application Mode.

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

3. ADS_WATCHDOG_ERROR_WDT_RUNNING if Watchdog is enabled. The Watchdog is running now and cannot change mode.

6.1.6 WDT_GetMode

LRESULT LONG = WDT_GetMode (LONG DriverHandle, WatchMode * o_pWatchMode);

Purpose

Get the current running mode of the Watchdog.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	long	default	IN, Driver handle. Assigned by WDT_Init
WatchMode	Output	pointer to WatchMode	default	The current watch mode of the Watchdog.

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

6.1.7 WDT_SetTimerSpan

LONG status = WDT_SetTimerSpan (LONG DriverHandle, DWORD i_dwIndex)

Purpose

Set the Watchdog timer span index.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	long	default	IN, Handle of the Watchdog driver. Assigned by WDT_Init
dwIndex	Input	USHORT	1-255	IN, the value for the Watchdog timer span index.

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

3. ADS_WATCHDOG_ERROR_WDT_RUNNING if Watchdog is enabled. The Watchdog is running now and cannot set the timer span of the Watchdog.

4. ADS_WATCHDOG_ERROR_INVALID_PARAMETER the Watchdog timer span index value is invalid. The Watchdog timer span index should be "(0 < i_dwIndex < 15)".

6.1.8 WDT_GetTimerSpan

LONG status = WDT_GetTimerSpan (LONG DriverHandle, DWORD * o_pIndex, DWORD * o_pValue);

Purpose

Get the Watchdog timer span index.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	long	default	IN, Handle of the Watchdog driver. Assigned by WDT_Init

o_pIndex	Output	pointer to USHORT	default	Pointer that indicates the address for retrieving the Watchdog The timer span index.
o_pValue	Output	pointer to BOOL	default	Pointer that indicates the address for retrieving the Watchdog current time span value of Watchdog timer

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

6.1.9 WDT_Reboot

LONG status = WDT_Reboot (LONG DriverHandle)

Purpose

Reboot the machine by the Watchdog.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input / Output	long pointer	default	IN, Driver handle. Assigned by WDT_Init

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

3. ADS_WATCHDOG_ERROR_WDT_REBOOTTING if the Watchdog is Rebooting.

4. ADS_WATCHDOG_ERROR_WDT_NOTRUNNING if the Watchdog is disabled. Watchdog not running now and cannot reboot the machine.

6.1.10 WDT_IsEnabled

LONG status = WDT_IsEnabled (LONG DriverHandle, BOOL * o_bEnabled)

Purpose

Get the Watchdog current running status: Enabled or Disabled.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	long	default	IN, Handle of the Watchdog driver. Assigned by WDT_Init
o_bEnabled	Output	pointer to BOOL	default	The Watchdog current running status, TRUE for enabled and FALSE for disabled.

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

6.1.11 WDT_LogEvent

LONG status = WDT_LogEvent (LONG DriverHandle, BOOL i_bLog)

Purpose

Set the Watchdog operations: Enable, Disable and Reboot to be logged into the system event base or not.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	long	default	IN, Handle of the Watchdog driver. Assigned by WDT_Init
o_bEnabled	Output	pointer to BOOL	default	TRUE for log the three operations into system base, FALSE for not logging.

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

3. ADS_WATCHDOG_ERROR_WDT_RUNNING if Watchdog is enabled. The Watchdog is running now and cannot set the timer span of the Watchdog.

4. ADS_WATCHDOG_ERROR_WDT_REBOOTTING if the Watchdog is Rebooting.

6.1.12 WDT_IsLogged

LONG status = WDT_IsLogged (LONG DriverHandle, BOOL * o_bLogged)

Purpose

Get the Watchdog event log information: The "Enable", "Disable", "Reboot" operation logged into system event base or not.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	long	default	IN, Handle of the Watchdog driver. Assigned by WDT_Init
o_bLogged	Output	pointer to BOOL	default	TRUE for log the "Enabled", "Disable", "Reboot" operations into system base, FALSE for not logging.

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

6.1.13 WDT_IsReadyToReboot

LRESULT status = WDT_IsReadyToReboot (LONG DriverHandle, BOOL * Reboot)

Purpose

Get the Watchdog current reboot status.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	long	default	IN, Driver handle. Assigned by WDT_Init
Reboot	Output	pointer to BOOL	default	Pointer that indicates the address for retrieving the Watchdog current reboot status.

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

6.1.14 WDT_GetStartTime

LONG status = WDT_GetStartupTime (LONG DriverHandle, LARGE_INTEGER * o_pSartupTime);

Purpose

Get the Watchdog enabled time.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	long	default	IN, Handle of the Watchdog driver. Assigned by WDT_Init.
o_pSartupTime	Output	pointer to USHORT	default	The count of 100-nanosecond intervals that the Watchdog is enabled.

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

6.1.15 WDT_Strobe

LRESULT status = WDT_Strobe (LONG DriverHandle)

Purpose

Strobe the Watchdog.
In WDT Application Mode, after enabling the Watchdog using **WDT_Enable**, the application must continuously call **WDT_Strobe** to trigger the Watchdog.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input / Output	long pointer	default	IN, Handle of the Watchdog driver. Assigned by WDT_Init.

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful, else maybe get STATUS_DEVICE_BUSY if the chip is busy, please call the function in few seconds.

2. INVALIDDEVICEHANDLE if DriverHandle is NULL.

3. ADS_WATCHDOG_ERROR_WDT_NOTRUNNING if the Watchdog is disabled. Watchdog not running now and cannot reboot the machine

6.1.16 WDT_SetType

LONG status = WDT_SetType (LONG DriverHandle, WatchdogType i_type)

Purpose

Set the Watchdog type. In current version this function is reserved for further extension and no implementation is available Set the Watchdog type.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	long	default	IN, Handle of the Watchdog driver. Assigned by WDT_Init.
i_type	Input	WatchdogType	default	IN, The type of the Watchdog. Valid WatchdogType: WATCHDOG_TYPE_W83977AF; WATCHDOG_TYPE_W83627HF;

				WATCHDOG_TYPE_SOM443; WATCHDOG_TYPE_SOM5780; WATCHDOG_TYPE_SCH3114; WATCHDOG_TYPE_NCT6776F; WATCHDOG_TYPE_EC; WATCHDOG_TYPE_NCT5523D; (include NCT6106D)

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

3. ADS_WATCHDOG_ERROR_WDT_RUNNING if Watchdog is enabled.

6.1.17 WDT_GetType

LONG status = WDT_GetType (LONG DriverHandle, WatchdogType * o_pType);

Purpose

Get the Watchdog chip type.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	long	default	IN, Handle of the Watchdog driver. Assigned by WDT_Init.
o_pType	Output	pointer to WatchdogType	default	The type of the Watchdog. Watchdog Type: WATCHDOG_TYPE_W83977AF; WATCHDOG_TYPE_W83627HF; WATCHDOG_TYPE_SOM443; WATCHDOG_TYPE_SOM5780; WATCHDOG_TYPE_SCH3114; WATCHDOG_TYPE_NCT6776F WATCHDOG_TYPE_EC;

				WATCHDOG_TYPE_NCT5523D; (include NCT6106D) WATCHDOG_TYPE_NCT6116D; WATCHDOG_TYPE_ EIO; WATCHDOG_TYPE_ NCT6126D; WATCHDOG_TYPE_ SOC;

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

6.1.18 WDT_GetTimerSpanDescription

LONG status = WDT_GetTimerSpanDescription (LONG DriverHandle, DWORD i_dwIndex, LPTSTR o_pDescription);

Purpose

Get the description of the specified timer span index.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	long	default	IN, Handle of the Watchdog driver. Assigned by WDT_Init.
i_dwIndex	Input	DWORD	default	The timer span index.
o_pDescription	Output	long pointer to string	default	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 Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

3. ADS_WATCHDOG_ERROR_INVALID_PARAMETER the Watchdog timer span index value is

invalid. The Watchdog timer span index should be "(0 < i_dwIndex < 15)".

6.1.19 WDT_GetErrMsg

LONG status = WDT_GetErrMsg (LONG i_hHandle, LONG i_lErrCode, LPTSTR o_pErrMsg)

Purpose

Retrieves the message of error according to the error code and returns it to the message buffer.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	LONG	default	IN, Handle of the Watchdog driver. Assigned by WDT_Init.
i_lErrCode	Input	LONG	default	The error code returned by a function call.
o_pErrMsg	Output	long pointer to string	default	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

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

Note: ErrorCode and **ErrorMessage** refer to: **Error Codes**.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

3. ADS_WATCHDOG_ERROR_INVALID_PARAMETER The error code is invalid.

6.1.20 WDT_SetFreeTimeoutValue

LONG status = WDT_SetFreeTimeoutValue (LONG i_hHandle, pFREETIMEOUT_CONFIG pFreeTimeoutConfig)

Purpose

Support FreeTimeout Range for EC WATCHDOG.

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	LONG	default	IN, Handle of the Watchdog driver. Assigned by WDT_Init.
pFreeTimeoutConfig	Input	pointer to pFREETIMEOUT_CONFIG	default	Pointer to Freetimeout configuration value include ID and number of config value. ID Type: WDT_EC_ID_TIMEOUT_MAX; WDT_EC_ID_TIMEOUT_MIN; WDT_EC_ID_FREETIMEOUT_VALUE;

Return

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

Note: ErrorCode and **ErrorMessage** refer to: **Error Codes**.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if Driver Handle is NULL.

6.1.21 WDT_GetWDTConfig

LONG status = WDT_GetWDTConfig (LONG DriverHandle, pFREETIMEOUT_CONFIG pFreeTimeoutConfig, ULONG * BufferLen);

Purpose

Get Max and min of timeout value in specific chiptype for WDT and Get FreeTimeOut Value

Parameters

Name	Direction	Type	Range	Description
DriverHandle	Input	long	default	IN, Handle of the Watchdog driver. Assigned by WDT_Init

pFreeTimeoutConfig	Input	pointer to pFREETIMEOUT_CONFIG	default	Pointer to Freetimeout configuration value include ID and number of config value. ID Type: WDT_EC_ID_TIMEOUT_MAX; WDT_EC_ID_TIMEOUT_MIN; WDT_EC_ID_FREETIMEOUT_VALUE;
BufferLen	Input	pointer to ULONG	default	Pointer to size of configuration value

Return Value

1. ADS_WATCHDOG_ERROR_SUCCESS if successful.

2. ADS_WATCHDOG_ERROR_INVALID_HANDLE if DriverHandle is NULL.

6.2 Error Codes

This section lists the status codes returned by these driver functions, including the name and description. Each driver function returns a status code that indicates whether the function was performed successfully. When a function returns a code that is neither 0 nor 500, it means that the function performed failed. You can pass it to WDT_GetErrMsg function to return its error message.

6.2.1 Error Code List

This section lists the status codes returned by these driver functions, including the name and description. Each driver function returns a status code that indicates whether the function was performed successfully. When a function returns a code that is neither 0 nor 500, it means that the function failed to perform. You can pass the error code to WDT_GetErrMsg function to return its error message.

A summary of the status codes is listed below:

Error Code	**Error ID**	**Description** **(Error Message)**
1	DrvErrorCode	
DrvErrorCode + 0	MemoryAllocateFailed	Memory allocation failure.
DrvErrorCode + 2	InvalidDeviceHandle	Invalid device handle
DrvErrorCode + 25	CreateFileFailed	fail to open low level driver
500	WDT_DevErrorCode	This operation is success.
WDT_DevErrorCode + 0	ADS_WATCHDOG_ERROR_SUCCESS	This operation is success
WDT_DevErrorCode + 1	ADS_WATCHDOG_ERROR_INITFAILED	Initialize Watchdog failed.
WDT_DevErrorCode + 2	ADS_WATCHDOG_ERROR_DEINITFAILED	De-initialize the Watchdog failed.
WDT_DevErrorCode + 3	ADS_WATCHDOG_ERROR_INVALID_HANDLE	Invalid Device Handle.
WDT_DevErrorCode + 4	ADS_WATCHDOG_ERROR_INVALID_PARAMETER	Invalid input parameter.
WDT_DevErrorCode + 5	ADS_WATCHDOG_ERROR_WDT_RUNNING	The Watchdog is running now and cannot do this kind of operation.
WDT_DevErrorCode + 6	ADS_WATCHDOG_ERROR_WDT_NOTRUNNING	The Watchdog is not running now, cannot do this kind of operation.

WDT_DevErrorCode + 7	ADS_WATCHDOG_ERROR_WDT_REBOOTTING	The Watchdog is ready to reboot now, cannot do this kind of operation.
WDT_DevErrorCode + 1000	ADS_WATCHDOG_ERROR_DEUBG_CODE	
0x80000011	STATUS_DEVICE_BUSY	The chip is busy in few seconds, cannot be called continuously. Please wait a few seconds to call. It is hardware limited, especially for strobe function.

6.3 Data Structure

6.3.1 WatchMode

The watch mode of the Watchdog

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 Watchdog.

6.3.2 WatchdogType

This Watchdog type may be extended without any modification of the application source code.

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,
WATCHDOG_TYPE_SOM5780 = ADS_WATCHDOG_CHIPSET_SOM5780,
WATCHDOG_TYPE_SCH3114 = ADS_WATCHDOG_CHIPSET_SCH3114,
WATCHDOG_TYPE_NCT6776F = ADS_WATCHDOG_CHIPSET_NCT6776F,
WATCHDOG_TYPE_EC = ADS_WATCHDOG_CHIPSET_EC,
WATCHDOG_TYPE_NCT5523D= ADS_WATCHDOG_CHIPSET_NCT5523D_NCT6106D,
WATCHDOG_TYPE_NCT6116D= ADS_WATCHDOG_CHIPSET_NCT6116D,
WATCHDOG_TYPE_EIO= ADS_WATCHDOG_CHIPSET_EIO,
WATCHDOG_TYPE_NCT6126D= ADS_WATCHDOG_CHIPSET_NCT6126D,
WATCHDOG_TYPE_SOC= ADS_WATCHDOG_CHIPSET_SOC
};

Description:

(1). WATCHDOG_TYPE_UNKNOWN: Unknown Watchdog type.
(2). WATCHDOG_TYPE_W83977AF: The Winbond SuperIO W83977AF Watchdog Chip.
(3). WATCHDOG_TYPE_W83627HF: The Winbond SuperIO W83627HF Watchdog Chip.
(4). WATCHDOG_TYPE_SOM443: The Advantech 443 standard Watchdog Chip.
(5). WATCHDOG_TYPE_SOM5780: The Fintek F75111R/F75111RG/F75111N general purpose IO chip Watchdog.
(6). WATCHDOG_TYPE_SCH3114: The SMSC SCH311X (SCH3112, SCH3114 and SCH3116) product Watchdog Chip.
(7). WATCHDOG_TYPE_NCT6776F: The Nuvoton NCT6776F/D LPC I/O Chip.
(8). WATCHDOG_TYPE_EC: The Advantech EC Watchdog Chip.
(9). WATCHDOG_TYPE_NCT5523D: The Nuvoton NCT5523D/NCT6106D product Watchdog

Chip.

(10). WATCHDOG_TYPE_NCT6116D: The Nuvoton NCT6116D product Watchdog Chip

(11). WATCHDOG_TYPE_EIO: The Advantech EIO Watchdog Chip

(12). WATCHDOG_TYPE_NCT6126D: The Nuvoton NCT6126D product Watchdog Chip

(13). WATCHDOG_TYPE_SOC: The SOC Watchdog Chip

7. Device Driver Programming Examples

This chapter gives an overview of the examples we provided.

The following is the list of example programs we offered, which can be used for the reference of software development.

Here "Console" contains standard console mode DOS examples. ".NETV2.0" contains supporting languages by Windows.

The examples' source codes are located at:
C:\Program Files\ADVANTECH\Watchdog\Example directory.

The following is the programs list.

Console	**.NETV2.0**
AdsWatchdogUtil	WatchdogExample
ElapsedTime	
RebootMachine	
SetLog	
SetMode	
SetTimerSpan	
WatchApplication	
WatchSystem	

7.1 AdvWatchdogUtil Sub-Functions

The test example is a console program. This program supply thirteen operations, below is the command line usage of the test program:

7.1.1 Watchdog enable

Enable Watchdog.

Command line: Watchdog Enable

7.1.2 Watchdog disable

Disable Watchdog.

Command line: Watchdog Disable

7.1.3 Watchdog reboot (Hardware Reset)

Do not strobe the Watchdog timer and let it timeout to reset the computer.

Command line: Watchdog Reboot

7.1.4 Watchdog strobe

Strobe the Watchdog timer once.

Command line: Watchdog Strobe

7.1.5 Watchdog Set

Display Set help message.

Command line: Watchdog Set

7.1.6 Watchdog Get

Display Get help message.

Command line: Watchdog Get

7.1.7 Watchdog Set Timer

Set Watchdog timer span.

Command line: Watchdog Set Timer of i_time.

The value of i_time should be in the range of from 1 to 14.

7.1.8 Watchdog Set mode

Set the Watchdog mode.

Command line: Watchdog Set Mode i_mode

The value of i_mode should be "sys" or "app"

7.1.9 Watchdog Set Log

Set the Watchdog log status.

Command line: Watchdog Set Log i_LogEvent

7.1.10 Watchdog get timer

Get the Watchdog timer span.

Command line: Watchdog Get Timer

7.1.11 Watchdog Get mode

Get the Watchdog Mode.

Command line: Watchdog Get Mode

7.1.12 Watchdog Get Log

Get the Watchdog log status.

Command line: Watchdog Get Log

7.1.13 Watchdog

Display Overall help message

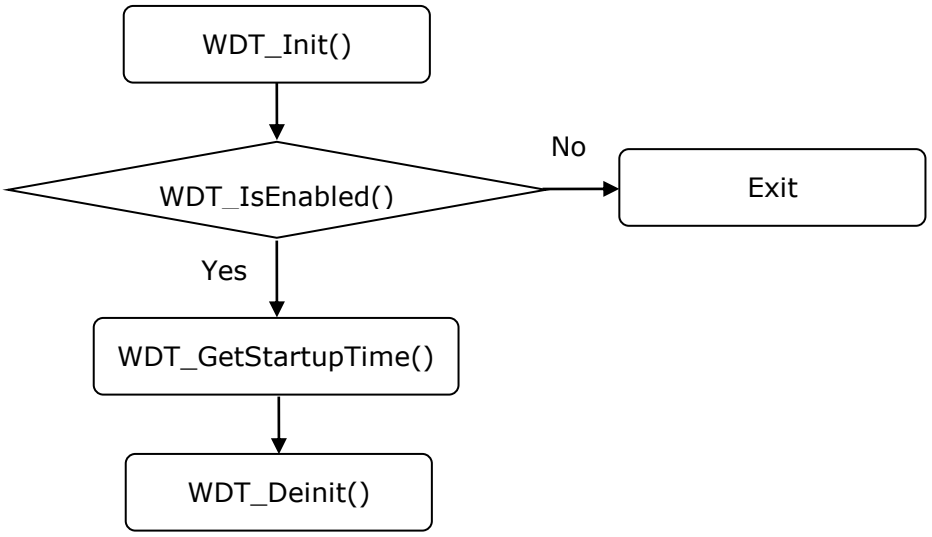
Command line: Watchdog /?

7.2 Example Function Call Flowchart

7.2.1 ElapsedTme

Path:

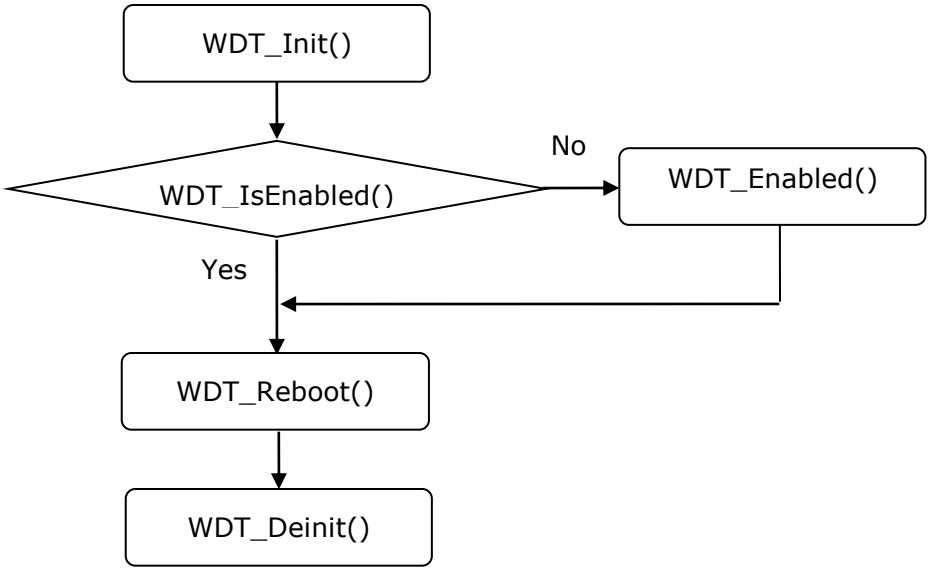
C:\Program Files\ADVANTECH\Watchdog\Example\Console\ElapsedTime\Elapsedtime.cpp



7.2.2 RebootMachine(Reset)

Path:

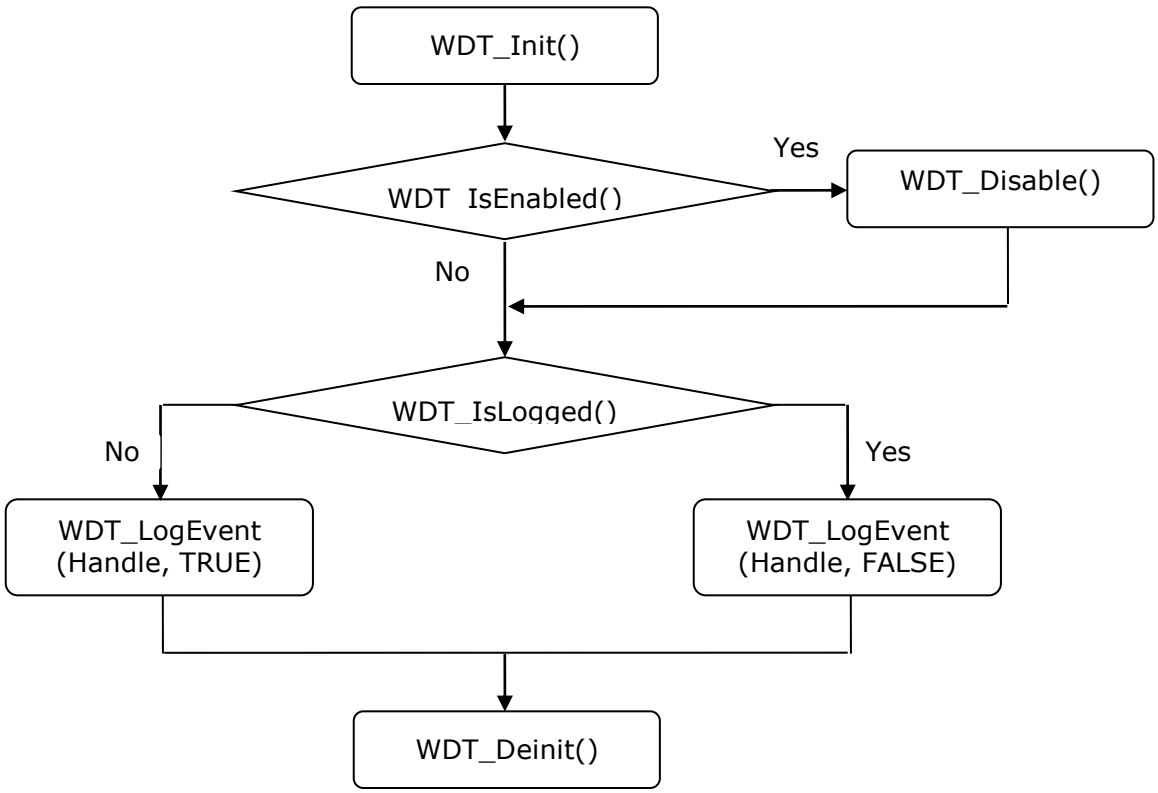
C:\Program Files\ADVANTECH\Watchdog\Example\Console\Rebootmachine\Rebootmachine.cpp



7.2.3 SetLog

Path:

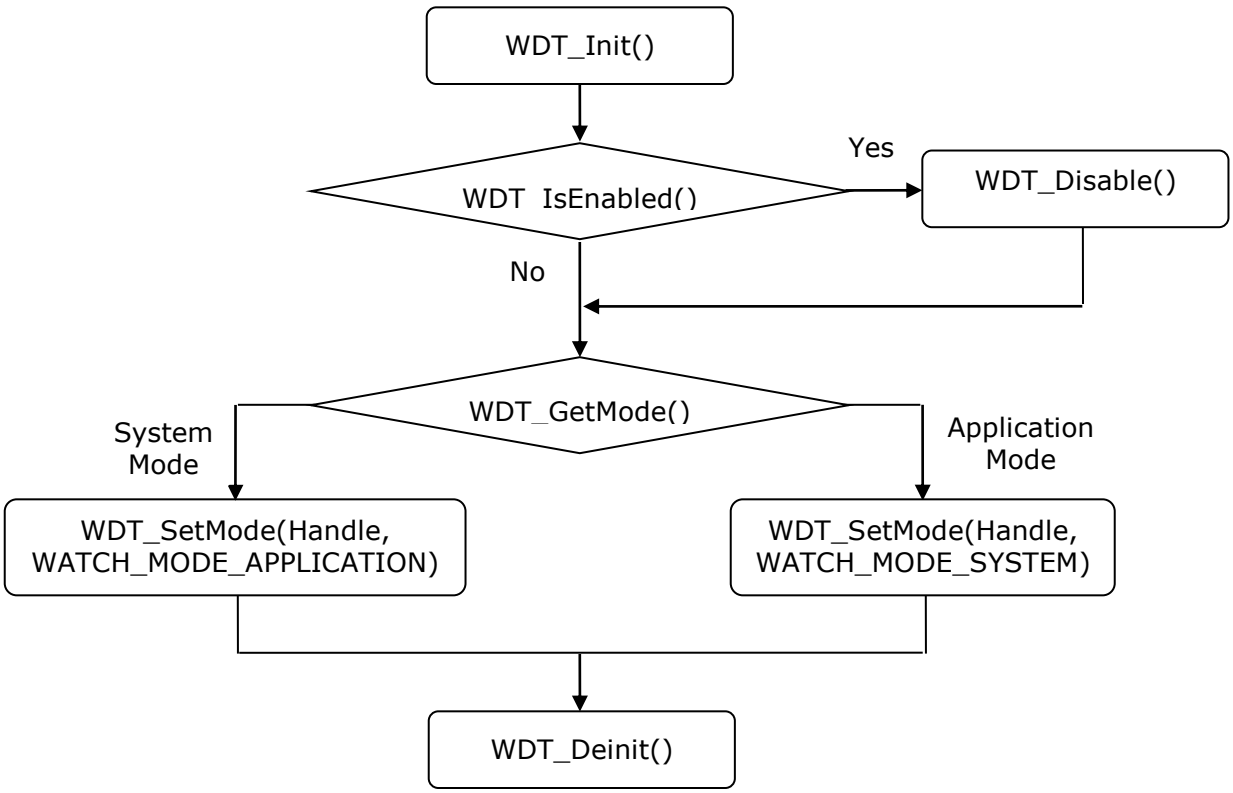
C:\Program Files\ADVANTECH\Watchdog\Example\Console\Setlog\Setlog.cpp



7.2.4 SetMode

Path:

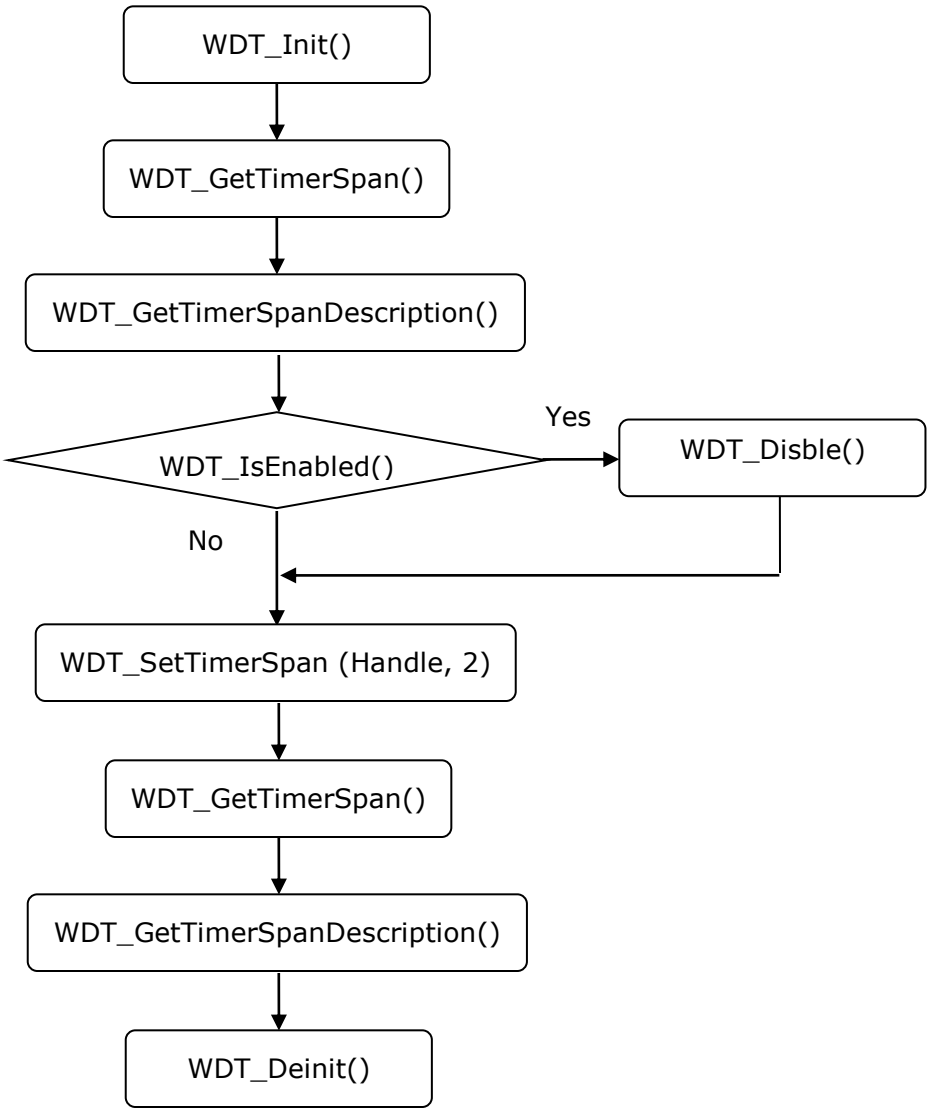
C:\Program Files\ADVANTECH\Watchdog\Example\Console\Setmode\Setmode.cpp



7.2.5 SetTimerSpan

Path:

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

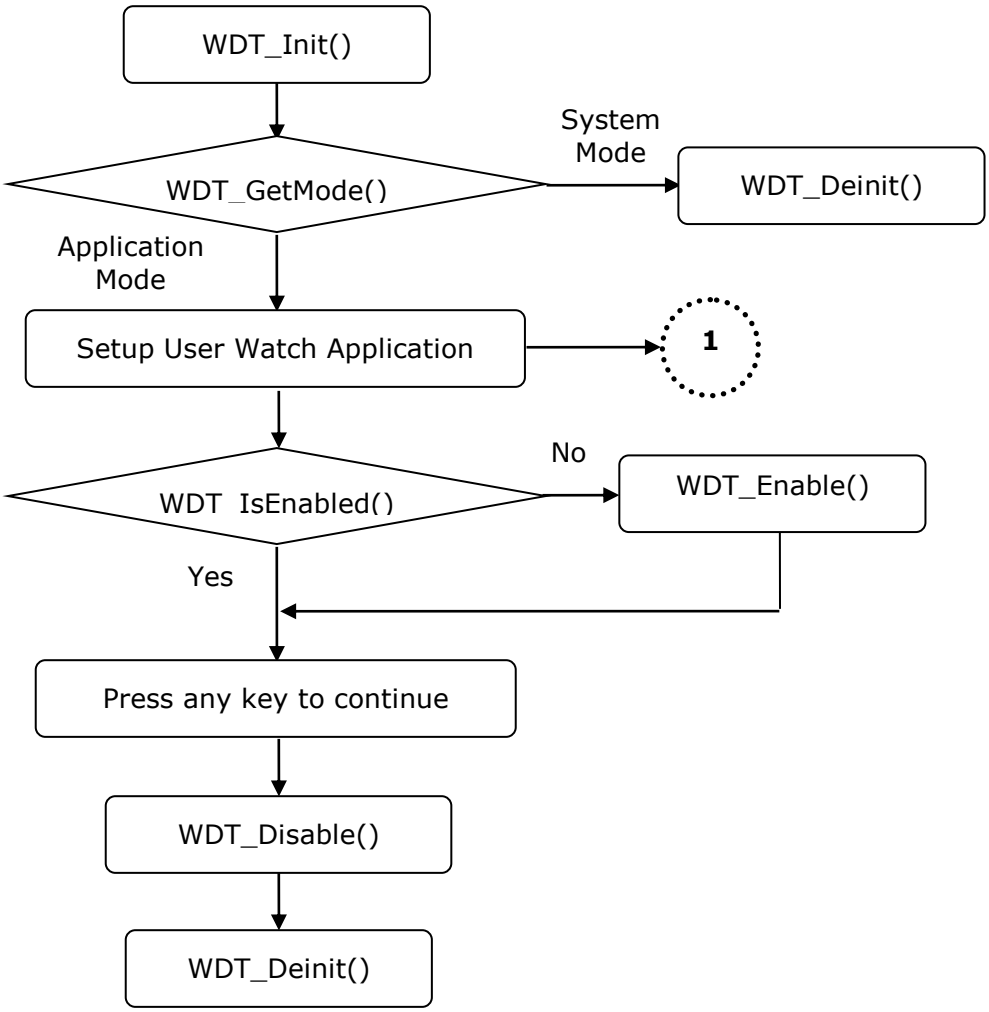


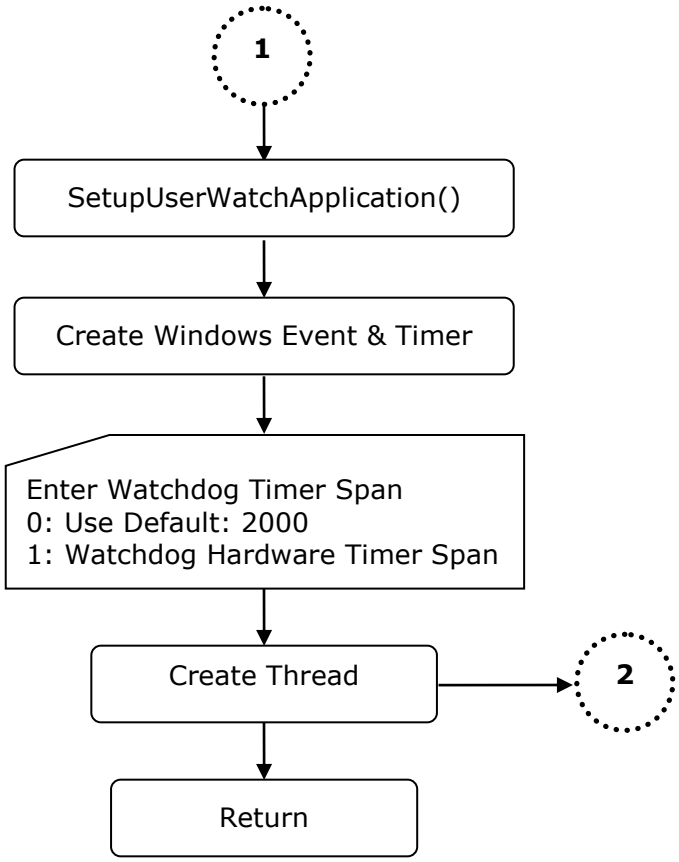
7.2.6 WatchApplication

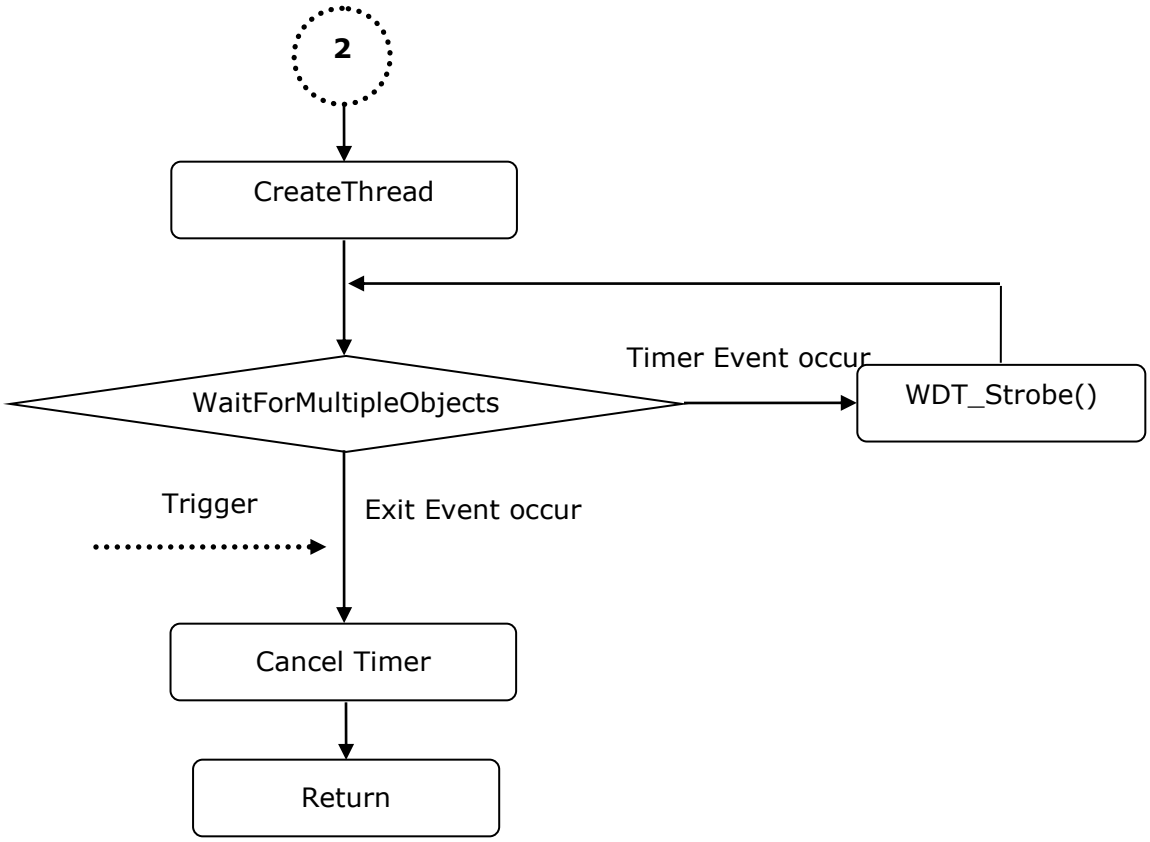
Path:

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

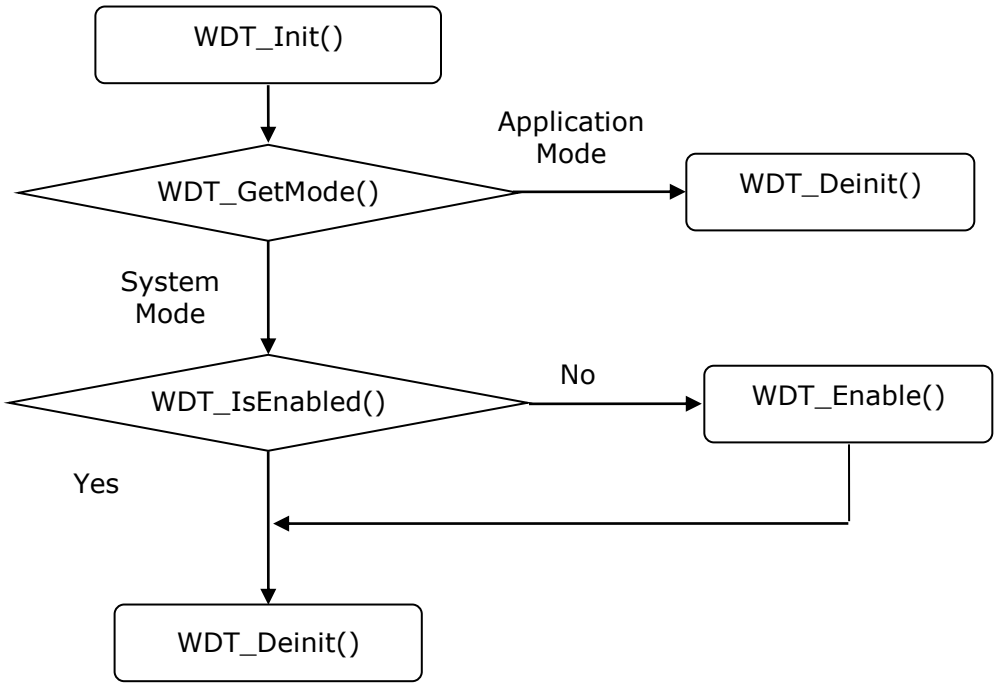

2
CreateThread
WaitForMultipleObjects
Timer Event occur
WDT_Strobe()
Trigger
Exit Event occur
Cancel Timer
Return

7.2.7 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.



7.2.8 SetFreeTimeoutValue

Path:

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