Industrial IoT

Advantech Brightness

Windows KMDF Driver

User Manual

Version <1.00>

Revision History

Date	Version	Description
2016/07/14	0.90	Initial draft
2023/03/08	1.00	Install Driver, Brightness Example

Table of Contents

1. Welcome to Advantech Brightness Windows KMDF Driver

1.1 About This Manual

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

This manual supplies information about driver interfaces of Advantech Brightness device, including calling procedure of operating Advantech Brightness device and descriptions of each function, parameter, and data structure.

This manual contains step-by-step instructions for building applications with the Advantech Brightness Windows KMDF Driver with Microsoft Visual C++ Win32. With the help of Advantech Brightness Windows KMDF Driver, you can develop applications with tools like VC++ Win32 in different Windows operating systems (Windows 7/8/8.1/10/Embedded Standard).

This manual also provides examples for Advantech Brightness Windows KMDF Driver, explaining how to use the driver with series of real examples and offering reference for you to develop your own applications.

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 7/8/8.1/10/Embedded Standard.

1.2 Organization of This Manual

This user manual is divided into the following sections:

- Welcome to Advantech Brightness Windows KMDF Driver
- Advantech Brightness Windows KMDF Driver Overview
- Getting Started with Advantech Brightness Windows KMDF driver

- Programming Guide
- Function Reference
- Software Utility & Programming Example

Welcome to Advantech Brightness Windows KMDF Driver

This section gives you an overview of this manual.

Advantech Brightness Windows KMDF Driver Overview

This section gives you a basic concept of Advantech Brightness Device Driver.

Getting Started with Advantech Brightness Windows KMDF Driver

This section gives the novice a clear concept of the Advantech Brightness Windows KMDF Driver and a walk-through in creating a simple application. Step-by-step instructions are given for an application written in Win32 MFC development environments.

Programming Guide

This section shows basic code Flow for the Brightness control and management.

Functions Reference

- ***Function Description***

  This section gives a brief introduction of each function (WINDOWS Native API) used in current development.

- ***CTL_CODE***

  This section describes all the control codes the Advantech Brightness Windows KMDF driver supports.

- ***Data Structure***

  This section describes the data structures that related to the provided functions.

Software Utility & Programming Example

This section gives an overview of the provided utility and example.

2. Advantech Brightness Windows KMDF Driver Overview

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

The Driver allows you to easily perform versatile Brightness operations in programs developed with tools like Microsoft Visual C++ 6.0 and other programming languages in different Windows operating systems. With this Driver, you don't have to use hardware-specific register commands.

This Driver also provides a sample application. You can modify the sample application to meet your needs.

The usage of windows driver is in the following aspects:

- **Driver Installation**

  The user should refer to Install Driver to install the driver.

- **Driver Uninstallation**

  The user should refer to Uninstall Driver to uninstall the driver.

- **Interface**

  The interface that windows driver used is as following.

  \\\\.\\AdvBrightnessDev

  Functions involved: CreateFile

2.1 Environments

2.1.1 Brightness

2.1.1.1 Hardware

It supports only Advantech IAG x86 hardware platform products with Brightness design; please see the release notes to check the hardware support list before using it.

2.1.1.2 Operating Systems

☐ Windows Embedded Standard 2009
☐ 32-bit/64-bit Microsoft Windows 7/8/8.1/10
☐ 32-bit/64-Bit Windows Embedded Standard 7
☐ 32-bit/64-Bit Windows Embedded 8 Standard
☐ 32-bit/64-Bit Windows Embedded 8.1 Industry Pro
☐ 32-bit/64-Bit Windows 10 Enterprise 2015 LTSB

2.1.1.3 Common Driver

The Brightness is based on Common driver (AdvBrightness for Windows) in Windows and the Common driver (AdvCOMMON for Windows) must be installed first.

2.2 Product Features

The Advantech Brightness Windows KMDF driver mainly includes the following features:

- **Brightness Information:**
  - **Brightness Current, Minimum, and Maximum Values:**
    Gets the current brightness value and supported minimum and maximum values.
  - **Auto Brightness support:**
    Gets the auto-brightness function supported status.

- **Brightness Configuration**
  - **Brightness Value**

Brightness can be set to any value between minimum and maximum.

- **Auto Brightness Enable/Disable**

  The auto-brightness function can be enabled or disabled if the target platform supports this function.

- **Brightness Tool Example Source Code**
  - A tool to manage Brightness.

    You can use it to configure the Brightness.
  - Example program

    The example programs which can be used for the reference of software development.

2.3 Installation

2.3.1 Install Driver

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

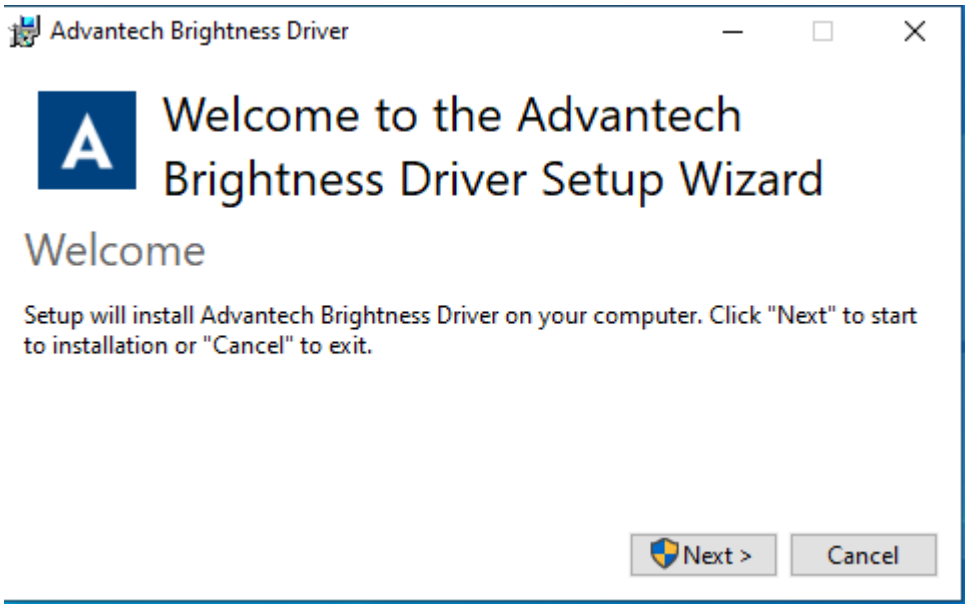
How to install Advantech Brightness Windows KMDF driver

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

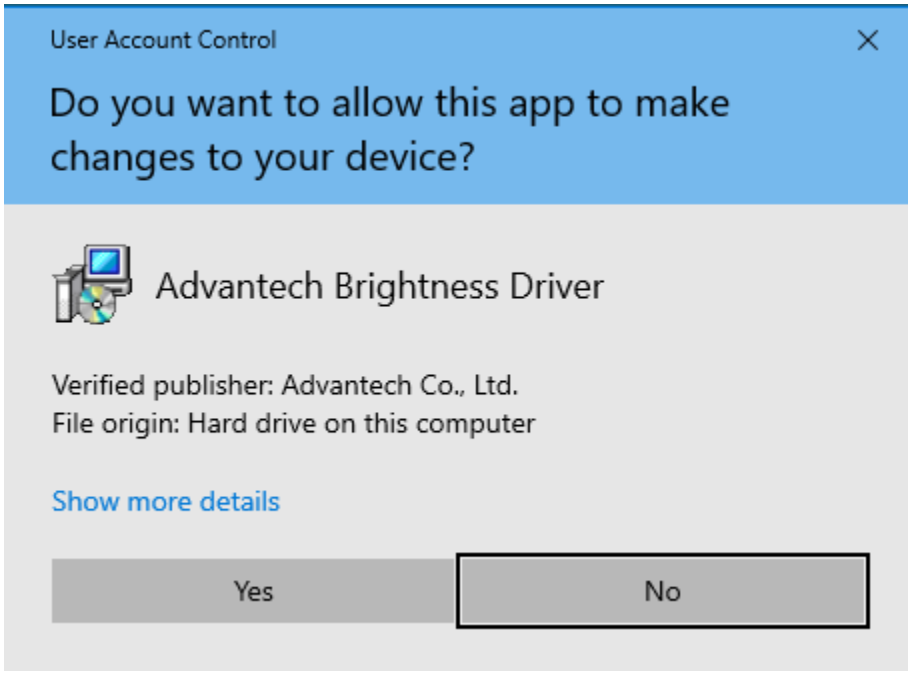
Below is an example of Advantech Brightness Windows KMDF driver setup. If you want to stop the setup, click the "Cancel" button in the setup program. The Setup program will stop the procedure automatically.

1. Run the Setup program.

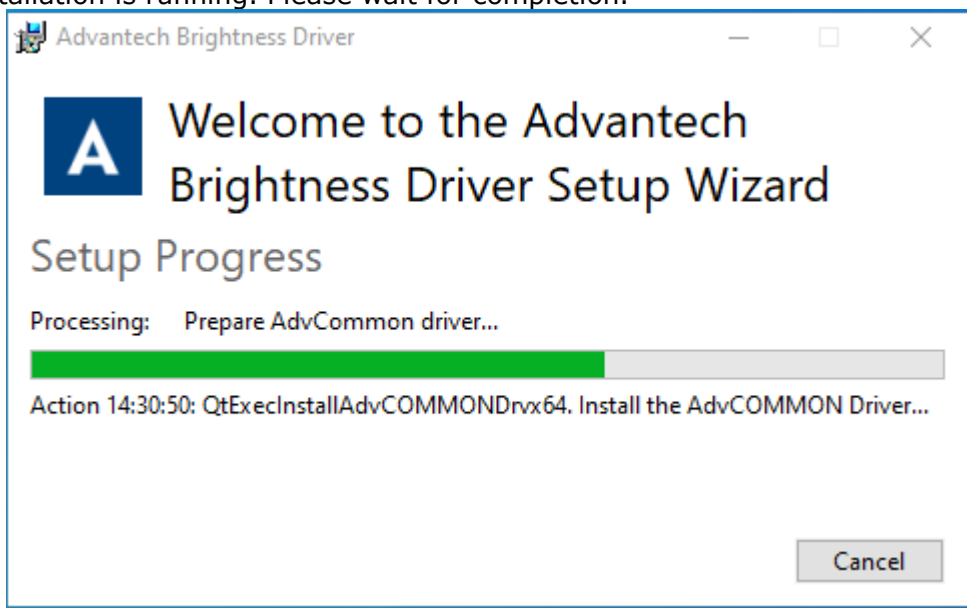
2. When the setup program is running, click the “Next” button.



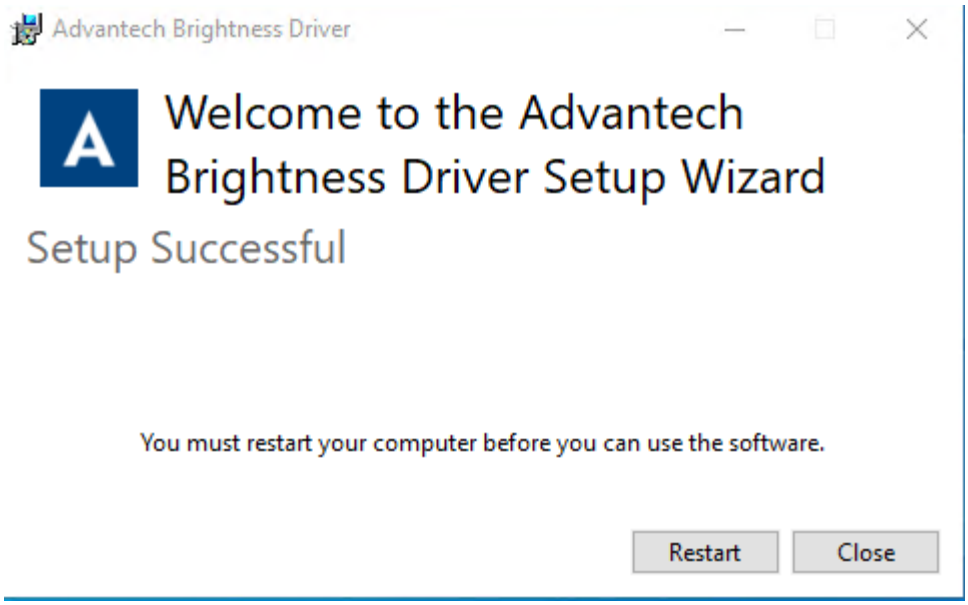
3. To continue the installation, click **“Yes”** and click **Install** to complete the driver installation.



4. The installation is running. Please wait for completion.



5. Click **Restart** to reboot and finish the installation of Advantech Brightness Windows KMDF Driver.

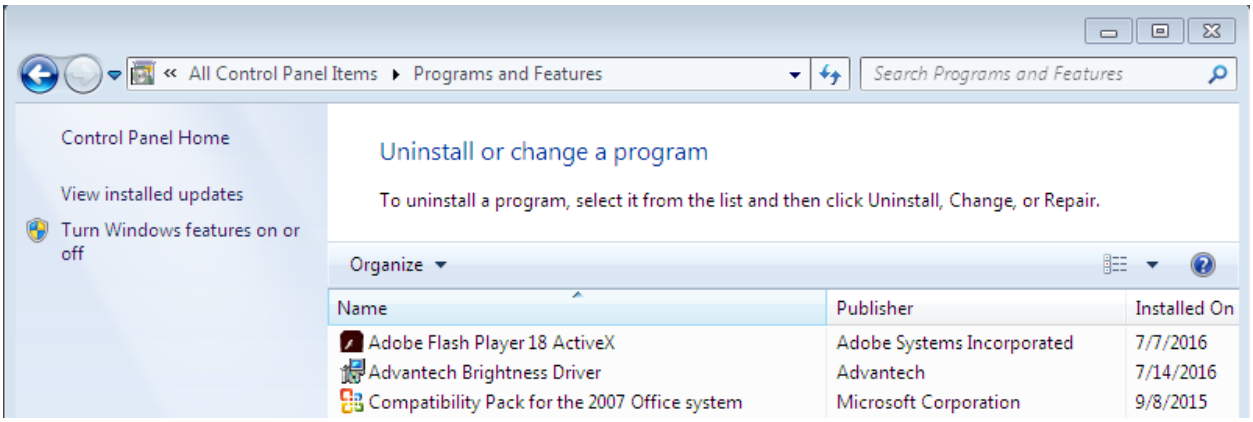


2.4 Uninstallation

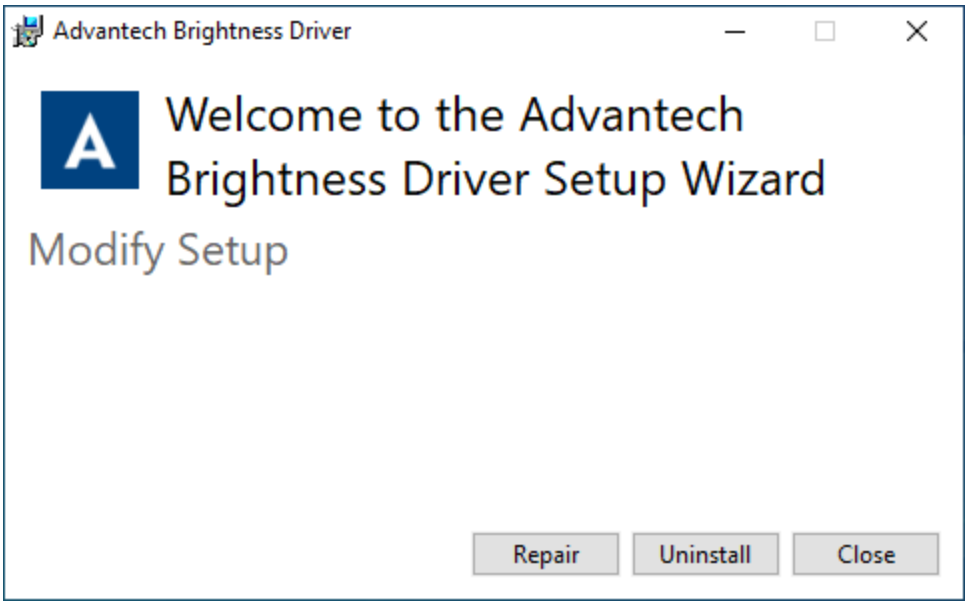
2.4.1 Uninstall Driver

How to uninstall Advantech Brightness Windows KMDF driver

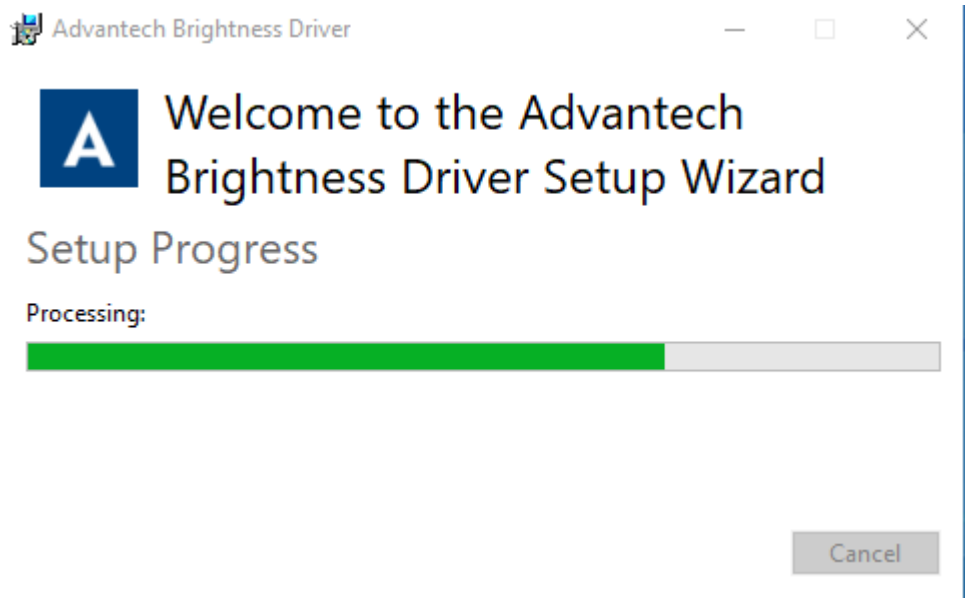
1. Go to **Control panel > Programs and Features** and select the **Advantech Brightness Windows KMDF driver**. Click **Uninstall** to launch uninstallation wizard.



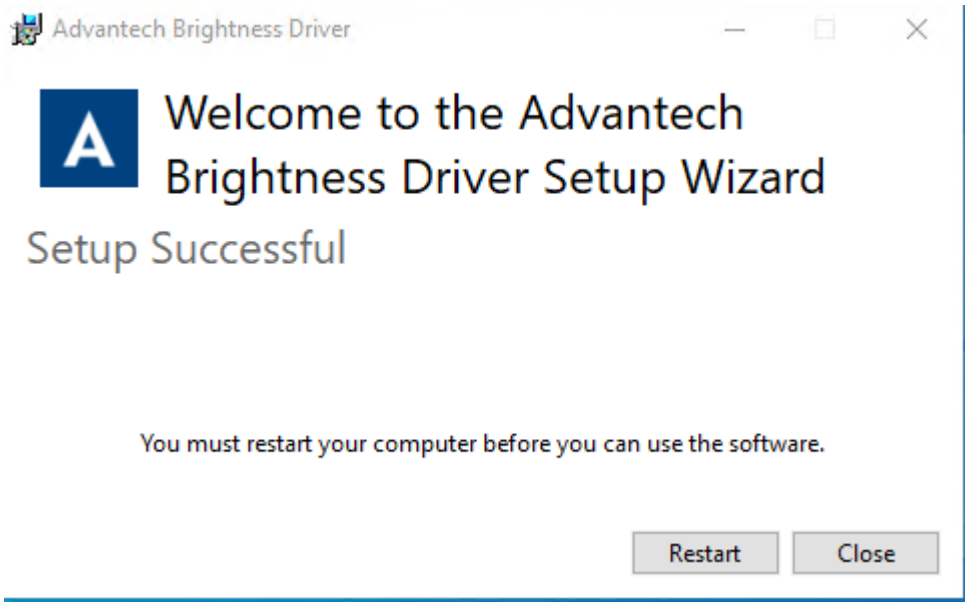
2. Click **Uninstall** to remove the Advantech Brightness Windows KMDF driver.



3. Then the setup program will start to remove the Advantech Brightness Windows KMDF Driver.



4. Just click **Restart** to complete the setup program.



3. Getting Started with Advantech Brightness Windows KMDF driver

This chapter provides a step-by-step example to demonstrate how to build an application using Device Driver from scratch in Microsoft Visual C++ 2005.

3.1 For Microsoft Visual C++ Win32

3.1.1 Create an Empty Visual C++ Project

To use the Brightness functions, follow this procedure:

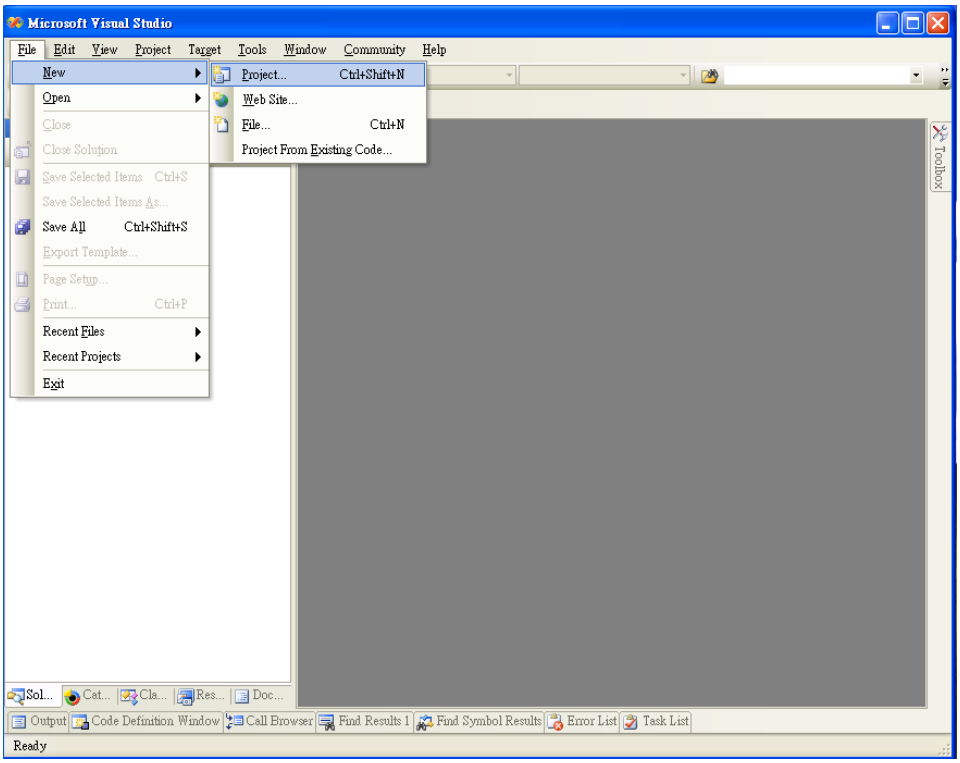
1. Create your source files as you would for other Windows programs written in C++ by calling DLL functions as typical function calls.

2. Include the header file and the IOCTL definitions as shown in the following example:

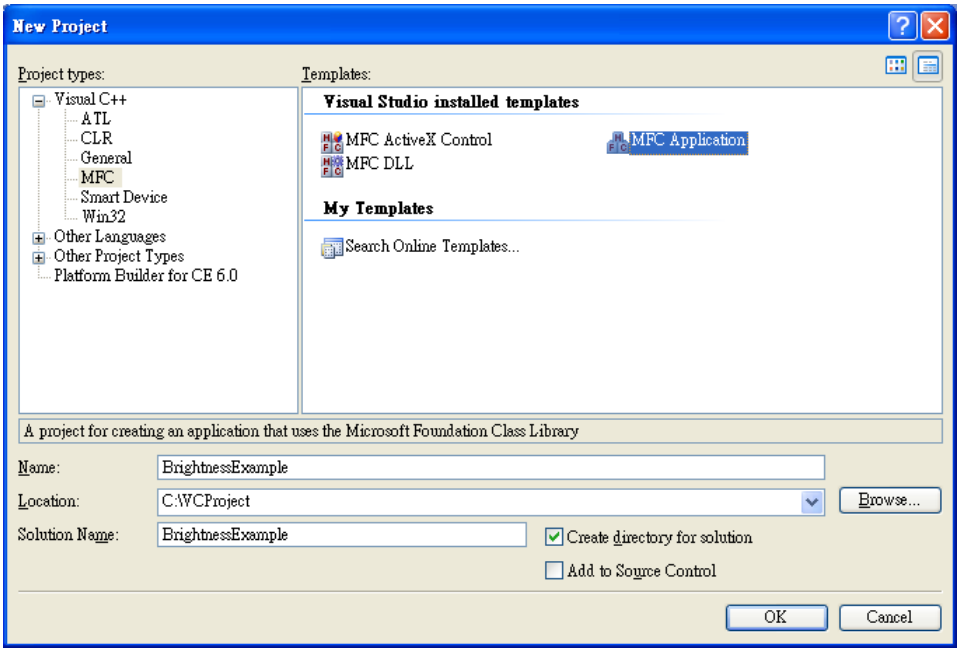
```
//
#define FILE_DEVICE_BACKLIGHT            FILE_DEVICE_UNKNOWN
#define IOCTL_ADVBRIGHTNESS_GET_VALUE \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9b0, METHOD_BUFFERED, FILE_WRITE_ACCESS )
#define IOCTL_ADVBRIGHTNESS_SET_VALUE \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9b1, METHOD_BUFFERED, FILE_WRITE_ACCESS )

#define IOCTL_ADVBRIGHTNESS_GET_CONFIG \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9c1, METHOD_BUFFERED, FILE_WRITE_ACCESS )
#define IOCTL_ADVBRIGHTNESS_SET_CONFIG \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9c2, METHOD_BUFFERED, FILE_WRITE_ACCESS )

typedef struct _BRIGHTNESS_CONFIG {
    ULONG Id;
    ULONG Value;
} BRIGHTNESS_CONFIG, *PBRIGHTNESS_CONFIG;

#define BRIGHTNESS_ID_AUTO_BRIGHTNESS  1
```

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

1. Click **File/New/Project…** 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 Application", and assign a project file location.



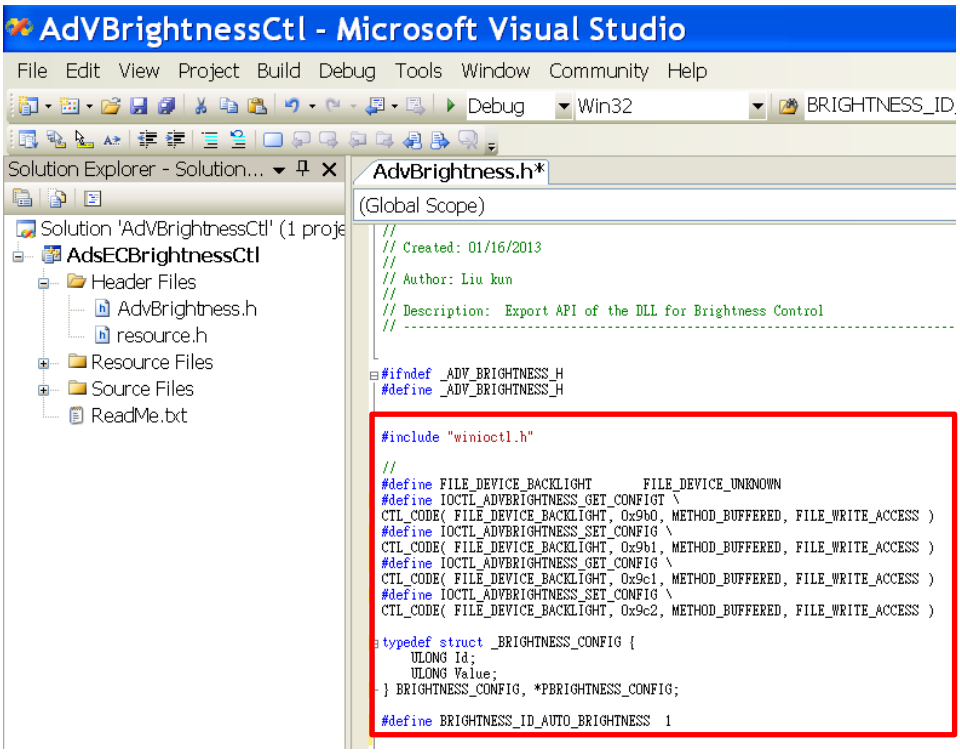
Run through the wizard to create the new project from Empty.

3.1.2 Adding Necessary Definitions

In order to develop Brightness applications with Advantech Brightness Windows KMDF drivers, you have to firstly add necessary definitions.

1. Copy the below codes to your project, like following example:

```
#include <winioctl.h>
//
#define FILE_DEVICE_BACKLIGHT             FILE_DEVICE_UNKNOWN
#define IOCTL_ADVBRIGHTNESS_GET_VALUE \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9b0, METHOD_BUFFERED, FILE_WRITE_ACCESS )
#define IOCTL_ADVBRIGHTNESS_SET_VALUE \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9b1, METHOD_BUFFERED, FILE_WRITE_ACCESS )

#define IOCTL_ADVBRIGHTNESS_GET_CONFIG \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9c1, METHOD_BUFFERED, FILE_WRITE_ACCESS )
#define IOCTL_ADVBRIGHTNESS_SET_CONFIG \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9c2, METHOD_BUFFERED, FILE_WRITE_ACCESS )

typedef struct _BRIGHTNESS_CONFIG {
    ULONG Id;
    ULONG Value;
} BRIGHTNESS_CONFIG, *PBRIGHTNESS_CONFIG;

#define BRIGHTNESS_ID_AUTO_BRIGHTNESS  1
```



These definitions can all be used in your application programs.

3.1.3 Writing Codes

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

3.1.4 Test Your Program

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

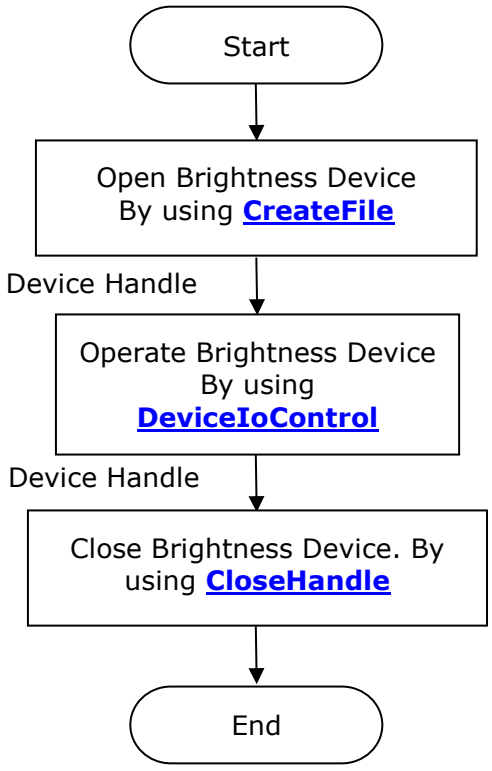
2. Run your saved ***.exe on you target platform.

4. Programming Guide

You can directly access drivers with WINDOWS Native API. In the following, we will provide an example by opening Brightness device and reading its current status to explain how to write basic applications in VC environment. Essential files for developing applications are listed below. Suppose the installation path of all files in the example is C:\Program Files\Advantech\AdvBrightness\Examples\VC++\BrightnessControl (or C:\Program Files (x86)\Advantech\AdvBrightness\Examples\VC++\BrightnessControl).

Device Function Group

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



5. Function Reference

Advantech Brightness Windows Driver contains a set of control codes and associated structures that can be used in various applications. The control codes support many development environments and programming languages, including Microsoft Visual C++ Win32 Program.

5.1 Function Description

You can manipulate Brightness through the WINDOWS Native APIs, thus makes use of the Brightness device through your existing application and examples without any change.
In your application, use the **CreateFile** function to open Brightness device; call the **DeviceIoControl** function to send a control code directly to the Advantech Brightness Windows KMDF driver to make the Brightness device to perform the corresponding operation; call the **CloseHandle** when operation is completed to close the opened Brightness device.

The following tables describe the main WINDOWS Native APIs used in current development.

Item	Name	Note
1)	CreateFile	Open Brightness device.
2)	CloseHandle	Close the opened Brightness device when operation is completed.
3)	DeviceIoControl	Send a control code directly to the Advantech Brightness Windows KMDF driver.

Only brief introduction is given in this manual to replace detailed usage of each function. Notes are made to remind you of important operations. For more detailed information about the usage, please see MSDN.

5.1.1 CreateFile

You can use the **CreateFile** function to open Brightness device. The function returns a handle that can be used to access the Brightness device.

Syntax

```
HANDLE CreateFile(
  LPCTSTR lpFileName,
  DWORD dwDesiredAccess,
  DWORD dwShareMode,
  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  DWORD dwCreationDisposition,
  DWORD dwFlagsAndAttributes,
  HANDLE hTemplateFile
);
```

Parameters

Name	Direction	Description
lpFileName	Input	[in] A pointer to a null-terminated string that specifies the name of the Brightness device to open. ***Note** In WINDOWS, use \\\\.\\AdvBrightnessDev.
dwDesiredAccess	Input	[in] The access to the Brightness device. Ways of opening the Brightness device. It is usually GENERIC_READ | GENERIC_WRITE.
dwShareMode	Input	[in] The sharing modes of the Brightness device which can be read, write, both, or none. It is usually FILE_SHARE_READ | FILE_SHARE_WRITE.
lpSecurityAttributes	Input	[in] A pointer to a **SECURITY_ATTRIBUTES** structure that determines whether or not the returned handle can be inherited by child processes. ***Note** The handle cannot be inherited. It must be set to **NULL**.
dwCreationDisposition	Input	[in] An action to take on files that exist and do not exist, which is usually **OPEN_EXISTING**.
dwFlagsAndAttributes	Input	[in] The file attributes and flags. ***Note** The Brightness device is not being opened or created for asynchronous I/O. It must be set to **0**.

hTemplateFile	Input	NULL

Return Value

If the function succeeds, the return value is an open handle to the Brightness device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call **GetLastError** function.

Remarks

Use the **CloseHandle** function to close the opened Brightness device handle that **CreateFile** returns when operation is completed.

Example Code

```
#include <winioctl.h>

// -----------------------------------------------------------------------
// DESCRIPTION: Open the Brightness Device
// -----------------------------------------------------------------------
//-------------------------------------------------------------
// Function    : Brightness_DeviceOpen
//
// PURPOSE     : Open the Brightness Device
//
// Parameters  : DriverHandle (OUT)
//                                Handle of device
//
// Return      : NULL and DriverHandle (success)
//
//-------------------------------------------------------------
HANDLE Brightness_DeviceOpen()
{
        HANDLE DriverHandle = NULL;
        TCHAR BrightnessName[] = TEXT("\\\\.\\AdvBrightnessDev");
        DriverHandle = CreateFile(
                BrightnessName,
                GENERIC_READ | GENERIC_WRITE,
                FILE_SHARE_READ | FILE_SHARE_WRITE,
                NULL,
                OPEN_EXISTING,
                0,
                NULL );

        return DriverHandle;
}
```

See Also

CloseHandle

5.1.2 CloseHandle

Close the Brightness device by calling this function when operation is completed.

Syntax

BOOL CloseHandle(

HANDLE *hObject*

);

Parameters

Name	Direction	Description
hObject	Input	[in] Handle to the Brightness device which was opened.

Return Value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call **GetLastError** function.

Example Code

```
#include <winioctl.h>

// ---------------------------------------------------------------------------
// DESCRIPTION: Close the Brightness Device
// ---------------------------------------------------------------------------
//-------------------------------------------------------------
// Function    : Brightness_DeviceClose
//
// PURPOSE     : Close Brightness Device by handle.
//
// Parameters  : DriverHandle (IN)
//               Handle of device
//
// Return      : TRUE (success)
//
//-------------------------------------------------------------
BOOL Brightness_DeviceClose ( HANDLE DriverHandle )
{
        if (DriverHandle != INVALID_HANDLE_VALUE)
        {
                CloseHandle(DriverHandle);
```

```
            // reset DeviceHandle
            DriverHandle = NULL;
        }

        return TRUE;
}
```

See Also

CreateFile

5.1.3 DeviceIoControl

You can use the **DeviceIoControl** function to send a control code directly to the Advantech Brightness Windows KMDF driver to make the Brightness device to perform the corresponding operations For example, configure Brightness direction, get Brightness current direction, configure Brightness status, get current Brightness status, etc.

Syntax

BOOL DeviceIoControl(
  HANDLE *hDevice*,
  DWORD *dwIoControlCode*,
  LPVOID *lpInBuffer*,
  DWORD *nInBufferSize*,
  LPVOID *lpOutBuffer*,
  DWORD *nOutBufferSize*,
  LPDWORD *lpBytesReturned*,
  LPOVERLAPPED *lpOverlapped*
);

Parameters

Name	Direction	Description
hDevice	Input	[in] Handle to the Brightness device on which the operation is to be performed. To retrieve a Brightness device handle, use the **CreateFile** function
dwIoControlCode	Input	[in] Control code for the specific operation. This value identifies the specific operation to be performed.

		For a list of the supported control codes, see **CTL_CODE**.
lpInBuffer	Input	[in] Pointer to the input buffer that contains the data required to perform the operation. This parameter can be NULL if *dwIoControlCode* specifies an operation that does not require input data.
nInBufferSize	Input	[in] Size of the input buffer, in bytes.
lpOutBuffer	Output	[out] Pointer to the output buffer that is to receive the data returned by the operation. This parameter can be NULL if *dwIoControlCode* specifies an operation that does not return data.
nOutBufferSize	Input	[in] Size of the output buffer, in bytes.
lpBytesReturned	Output	[out] Pointer to a variable that receives the size of the data stored in the output buffer, in bytes.
lpOverlapped	Input	[in] Pointer to an OVERLAPPED structure. ***Note** *lpOverlapped* must be set to **NULL**.

Return Value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call **GetLastError** function.

5.2 CTL_CODE

The following table describe all the control code the Brightness device driver supports.

Item	Name	Note
1)	**錯誤! 找不到參照來源。**	Gets the Brightness values including brightness value, brightness minimum, and brightness maximum.
2)	**錯誤! 找不到參照來源。**	Sets the Brightness value.
3)	**錯誤! 找不到參照來源。**	Gets the Brightness configuration value of

		the specified Id.
4)	錯誤! 找不到參照來源。	Gets the current state of the specified Brightness.

```
// The IOCTL Definitions
#define FILE_DEVICE_BACKLIGHT             FILE_DEVICE_UNKNOWN
#define IOCTL_ADVBRIGHTNESS_GET_VALUE \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9b0, METHOD_BUFFERED, FILE_WRITE_ACCESS )
#define IOCTL_ADVBRIGHTNESS_SET_VALUE \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9b1, METHOD_BUFFERED, FILE_WRITE_ACCESS )
#define IOCTL_ADVBRIGHTNESS_GET_CONFIG \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9c1, METHOD_BUFFERED, FILE_WRITE_ACCESS )
#define IOCTL_ADVBRIGHTNESS_SET_CONFIG \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9c2, METHOD_BUFFERED, FILE_WRITE_ACCESS )
```

5.2.1 IOCTL_ADVBRIGHTNESS_GET_VALUE

The 錯誤! 找不到參照來源。 control code gets the Brightness values including brightness value, brightness minimum, and brightness maximum.

To perform this operation, call the **DeviceIoControl** function with the following parameters.

```
BOOL DeviceIoControl(
  (HANDLE) hDevice,              // handle to device
 IOCTL_ADVBRIGHTNESS_GET_VALUE,  // dwIoControlCode
  NULL,                          // lpInBuffer
  0,                             // nInBufferSize
  (LPVOID) lpOutBuffer,          // output buffer
  (DWORD) nOutBufferSize,        // size of output buffer
  (LPDWORD) lpBytesReturned,     // number of bytes returned
  NULL,                          // OVERLAPPED structure
);
```

Parameters

hDevice

[in] Handle to the Brightness device. To obtain a Brightness device handle, call the **CreateFile** function.

dwIoControlCode

[in] Control code for the operation. Use 錯誤! 找不到參照來源。 for this operation.

lpInBuffer

Not used with this operation; set to NULL.

nInBufferSize

Not used with this operation; set to zero.

lpOutBuffer

[out] Pointer to an array of three unsigned long values.

nOutBufferSize

[in] Size of the output buffer, in bytes.

lpBytesReturned

[out] Pointer to a variable that receives the size of the data stored in the output buffer, in bytes.

lpOverlapped

NULL. *lpOverlapped* must be set to **NULL**.

Return Values

If the operation succeeds, **DeviceIoControl** returns a nonzero value.

If the operation fails, **DeviceIoControl** returns zero. To get extended error information, call **GetLastError** function.

Example Code

```
#include <winioctl.h>

#define FILE_DEVICE_BACKLIGHT          FILE_DEVICE_UNKNOWN
#define IOCTL_ADVBRIGHTNESS_GET_VALUE \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9b0, METHOD_BUFFERED, FILE_WRITE_ACCESS )
// ----------------------------------------------------------------------------
// DESCRIPTION: Gets Brightness value/minimum/maximum.
// ----------------------------------------------------------------------------
BOOL Brightness_GetValues ( HANDLE DriverHandle )
{
        ULONG results[3] = {0};

        DWORD dwReturn = 0;
        BOOL  bRet = DeviceIoControl(
                DriverHandle,
                IOCTL_ADVBRIGHTNESS_GET_VALUE,
                NULL,
                0,
                &results,
                sizeof(results),
                &dwReturn,
                NULL );
        /*
                results[0] = Brightness Value;
```

```
            results[1] = Brightness Min;
            results[2] = Brightness Max;
        */
        return bRet;
}
```

Requirements

N/A

See Also

DeviceIoControl

5.2.2 IOCTL_ADVBRIGHTNESS_SET_VALUE

The 錯誤! 找不到參照來源。 control code sets the Brightness value.

To perform this operation, call the **DeviceIoControl** function with the following parameters.

```
BOOL DeviceIoControl(
  (HANDLE) hDevice,                 // handle to device
 錯誤! 找不到參照來源。,      // dwIoControlCode
  (LPVOID) lpInBuffer,              // lpInBuffer
  (DWORD) nInBufferSize,            // nInBufferSize
  NULL,                             // output buffer
  0,                                // size of output buffer
  (LPDWORD) lpBytesReturned,        // number of bytes returned
  NULL,                             // OVERLAPPED structure
);
```

Parameters

hDevice

[in] Handle to the Brightness device. To obtain a Brightness device handle, call the **CreateFile** function.

dwIoControlCode

[in] Control code for the operation. Use 錯誤! 找不到參照來源。 for this operation.

lpInBuffer

[in] Pointer to an unsigned long value.

nInBufferSize

[in] Size of the input buffer, in bytes.

lpOutBuffer

Not used with this operation; set to NULL.

nOutBufferSize

Not used with this operation; set to zero.

lpBytesReturned

[out] Pointer to a variable that receives the size of the data stored in the output buffer, in bytes.

lpOverlapped

NULL. *lpOverlapped* must be set to **NULL**.

Return Values

If the operation succeeds, **DeviceIoControl** returns a nonzero value.

If the operation fails, **DeviceIoControl** returns zero. To get extended error information, call **GetLastError** function.

Example Code

```
#include <winioctl.h>

#define FILE_DEVICE_BACKLIGHT          FILE_DEVICE_UNKNOWN
#define IOCTL_ADVBRIGHTNESS_SET_VALUE \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9b1, METHOD_BUFFERED, FILE_WRITE_ACCESS )
// -----------------------------------------------------------------------
// DESCRIPTION: Sets Brightness value.
// -----------------------------------------------------------------------
BOOL Brightness_SetValue ( HANDLE DriverHandle, ULONG ulValue )
{

        DWORD dwReturn = 0;
        BOOL  bRet = DeviceIoControl(
                DriverHandle,
                IOCTL_ADVBRIGHTNESS_SET_VALUE,
                &ulValue,
                sizeof(ulValue),
                NULL,
                0,
                &dwReturn,
                NULL );

        return bRet;
}
```

Requirements

N/A

See Also

DeviceIoControl

5.2.3 IOCTL_ADVBRIGHTNESS_GET_CONFIG

The **錯誤! 找不到參照來源。** control code gets the Brightness configuration value of the specified Id.

The input parameter structure, 錯誤! 找不到參照來源。, indicates which configuration value of the specified Id is going to get.

The output parameter structure, 錯誤! 找不到參照來源。, reports the specified configuration value.

To perform this operation, call the **DeviceIoControl** function with the following parameters.

```
BOOL DeviceIoControl(
  (HANDLE) hDevice,              // handle to device
  錯誤! 找不到參照來源。,     // dwIoControlCode
  (LPVOID) lpInBuffer,           // lpInBuffer
  (DWORD) nInBufferSize,         // nInBufferSize
  (LPVOID) lpOutBuffer,          // output buffer
  (DWORD) nOutBufferSize,        // size of output buffer
  (LPDWORD) lpBytesReturned,     // number of bytes returned
  NULL,                          // OVERLAPPED structure
);
```

Parameters

hDevice

[in] Handle to the Brightness device. To obtain a Brightness device handle, call the **CreateFile** function.

dwIoControlCode

[in] Control code for the operation. Use **錯誤! 找不到參照來源。** for this operation.

lpInBuffer

[in] Pointer to a 錯誤! 找不到參照來源。 structure.

nInBufferSize

[in] Size of the input buffer, in bytes.

lpOutBuffer

[out] Pointer to a 錯誤! 找不到參照來源。 structure.

nOutBufferSize

[in] Size of the output buffer, in bytes.

lpBytesReturned

[out] Pointer to a variable that receives the size of the data stored in the output buffer, in bytes.

lpOverlapped

NULL. *lpOverlapped* must be set to **NULL**.

Return Values

If the operation succeeds, **DeviceIoControl** returns a nonzero value.

If the operation fails, **DeviceIoControl** returns zero. To get extended error information, call **GetLastError** function.

Example Code

```
#include <winioctl.h>

#define FILE_DEVICE_BACKLIGHT          FILE_DEVICE_UNKNOWN
#define IOCTL_ADVBRIGHTNESS_GET_CONFIG \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9c1, METHOD_BUFFERED, FILE_WRITE_ACCESS )

typedef struct _BRIGHTNESS_CONFIG {
    ULONG Id;
    ULONG Value;
} BRIGHTNESS_CONFIG, *PBRIGHTNESS_CONFIG;
#define BRIGHTNESS_ID_AUTO_BRIGHTNESS  1
// ----------------------------------------------------------------------------
// DESCRIPTION: Gets Brightness configuration value
// ----------------------------------------------------------------------------
BOOL Brightness_GetConfig ( HANDLE DriverHandle, ULONG Id, PULONG pValue )
{
       BRIGHTNESS_CONFIG ebc = {0};
       ebc.Id = Id;

       if(pValue == NULL) return FALSE;

       DWORD dwReturn = 0;
       BOOL  bRet = DeviceIoControl(
              DriverHandle,
              IOCTL_ADVBRIGHTNESS_GET_CONFIG,
```

```
                &ebc,
                sizeof(BRIGHTNESS_CONFIG),
                &ebc,
                sizeof(BRIGHTNESS_CONFIG),
                &dwReturn,
                NULL );
        if( bRet )
        {
                *pValue = ebc.Value;
        }
        return bRet;
}

BOOL CheckIfAutoBrightnessSupport( HANDLE DriverHandle )
{
        ULONG ulAutoBrightness = 0;
        BOOL bRet = Brightness_GetConfig(DriverHandle,
                BRIGHTNESS_ID_AUTO_BRIGHTNESS,
                &ulAutoBrightness);
        if(!bRet)
        {
                printf("Failed to get config!!\n");
                return FALSE;
        }
        return (ulAutoBrightness == 1);
}
```

See Also

DeviceIoControl

5.2.4 IOCTL_ADVBRIGHTNESS_SET_CONFIG

The **錯誤! 找不到參照來源。** control code sets the current Brightness status.

The input parameter structure, 錯誤! 找不到參照來源。, indicates which Brightness configuration of the specified Id is going to set to the new value.

To perform this operation, call the **DeviceIoControl** function with the following parameters.

```
BOOL DeviceIoControl(
  (HANDLE) hDevice,              // handle to device
  錯誤! 找不到參照來源。,     // dwIoControlCode
  (LPVOID) lpInBuffer,           // lpInBuffer
  (DWORD) nInBufferSize,         // nInBufferSize
  NULL,                          // output buffer
```

```
  0,                                  // size of output buffer
  (LPDWORD) lpBytesReturned,          // number of bytes returned
  NULL,                               // OVERLAPPED structure
);
```

Parameters

hDevice

[in] Handle to the Brightness device. To obtain a Brightness device handle, call the **CreateFile** function.

dwIoControlCode

[in] Control code for the operation. Use **錯誤! 找不到參照來源。** for this operation.

lpInBuffer

[in] Pointer to a 錯誤! 找不到參照來源。 structure.

nInBufferSize

[in] Size of the input buffer, in bytes.

lpOutBuffer

Not used with this operation; set to NULL.

nOutBufferSize

Not used with this operation; set to zero.

lpBytesReturned

[out] Pointer to a variable that receives the size of the data stored in the output buffer, in bytes.

lpOverlapped

NULL. *lpOverlapped* must be set to **NULL**.

Return Values

If the operation succeeds, **DeviceIoControl** returns a nonzero value.

If the operation fails, **DeviceIoControl** returns zero. To get extended error information, call **GetLastError** function.

Example Code

```
#include <winioctl.h>

#define FILE_DEVICE_BACKLIGHT           FILE_DEVICE_UNKNOWN
#define IOCTL_ADVBRIGHTNESS_SET_CONFIG \
CTL_CODE( FILE_DEVICE_BACKLIGHT, 0x9c2, METHOD_BUFFERED, FILE_WRITE_ACCESS )
```

```
typedef struct _BRIGHTNESS_CONFIG {
    ULONG Id;
    ULONG Value;
} BRIGHTNESS_CONFIG, *PBRIGHTNESS_CONFIG;
#define BRIGHTNESS_ID_AUTO_BRIGHTNESS  1
// ------------------------------------------------------------------------
// DESCRIPTION: Sets Brightness configuration
// ------------------------------------------------------------------------
BOOL Brightness_SetConfig ( HANDLE DriverHandle, ULONG Id, ULONG Value )
{
       BRIGHTNESS_CONFIG ebc = {0};
       DWORD dwReturn = 0;

       ebc.Id = Id;
       ebc.Value = Value;

       BOOL  bRet = DeviceIoControl(
              DriverHandle,
              IOCTL_ADVBRIGHTNESS_SET_CONFIG,
              &ebc,
              sizeof(BRIGHTNESS_CONFIG),
              &ebc,
              sizeof(BRIGHTNESS_CONFIG),
              &dwReturn,
              NULL );

       return bRet;
}

BOOL EnableAutoBrightness( HANDLE DriverHandle, BOOL bEnabled )
{
       ULONG ulAutoBrightness = bEnabled ? 1 : 0;
       BOOL bRet = Brightness_SetConfig(DriverHandle,
              BRIGHTNESS_ID_AUTO_BRIGHTNESS,
              ulAutoBrightness);
       return bRet;
}
```

See Also

DeviceIoControl

5.3 Data Structure

5.3.1 BRIGHTNESS_CONFIG

Brightness Direction Structure

DeviceIoControl's parameter uses this structure.

BRIGHTNESS_CONFIG structure is defined as follows:

```
typedef struct _BRIGHTNESS_CONFIG {
   ULONG Id;
   ULONG Value;
} BRIGHTNESS_CONFIG, *PBRIGHTNESS_CONFIG;
```

Members Description

Id

The Id you want to get or set the configuration value.

It now supports the following Ids.

Name	Value	Description
BRIGHTNESS_ID_AUTO_BRIGHTNESS	1	Enable/Disable Auto-Brightness

Value

The value of the specified Id.

6. Software Utility & Programming Example

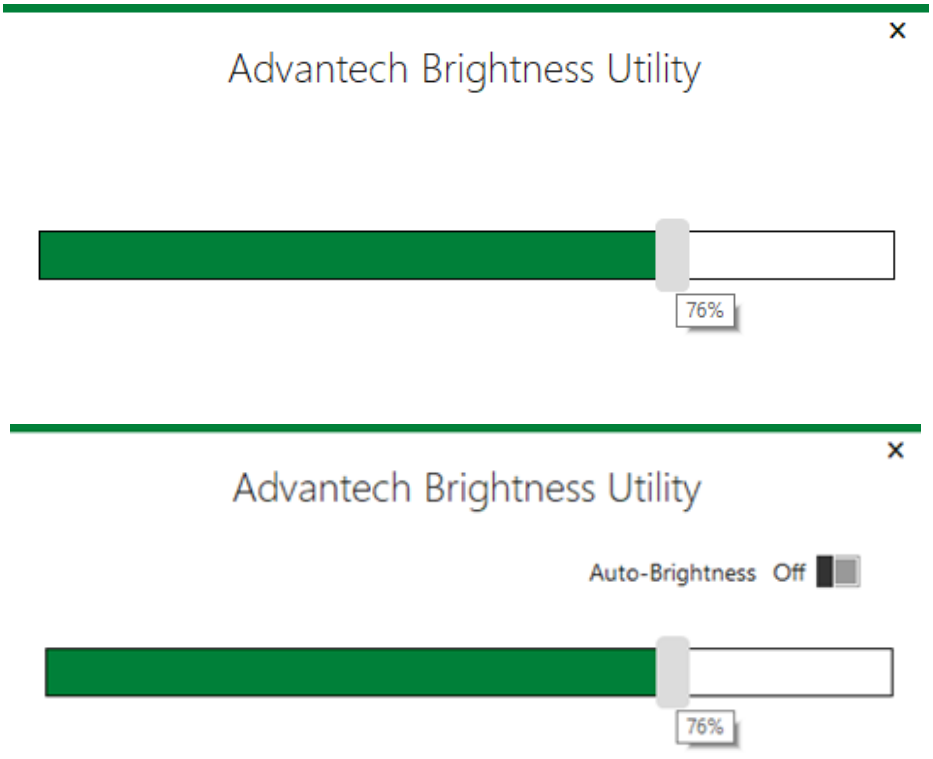
Advantech Brightness Windows Driver package contains examples of Microsoft Visual C++ Win32 Program, C#, VB.NET, .NET Class library and a utility. You can refer to these programs to develop applications.

Name	Description
BrightnessControl	This example shows how to configure/manage the Brightness.
AdvBrightnessUtility	The utility enables you to configure/manage the Brightness.

6.1 Advantech Brightness Example and Utility

Brightness Utility provides a way for you to manage brightness value and Auto-Brightness function (If applicable) And from 20% to 100% .

You can run it at: **Start menu** | **All Programs** | **Advantech** | **AdvBrightness** | **Brightness Utility**



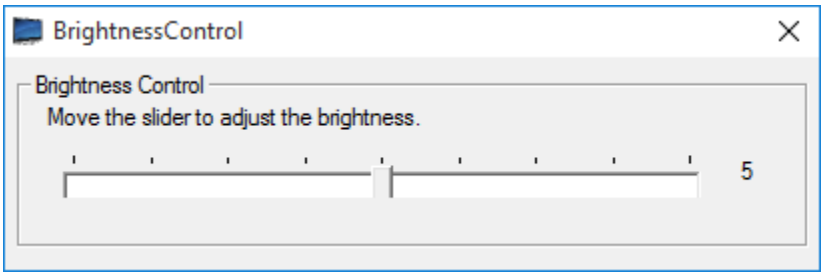
Example Source code

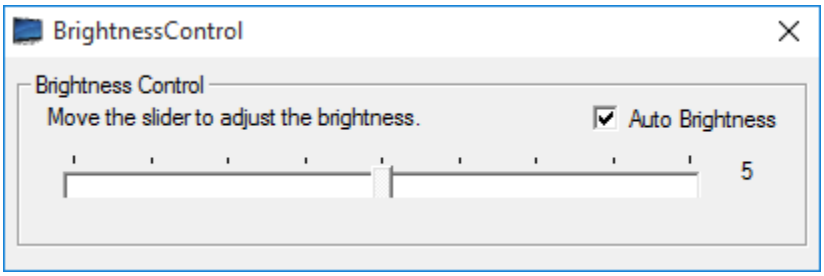
You can refer to the source code including VC++, C#, and VB.NET to develop your programs. The examples you can find it at: **Start menu** | **All Programs** | **Advantech** | **AdvBrightness** | **Brightness Examples**

Binary File (VC++)

File Name: BrightnessControl.exe

UI:





You can move the slider to adjust the brightness. If the target platform supports Auto-Brightness function, the UI will show the "**Auto Brightness**" check box as previous figure. If the Auto Brightness check box is checked, the brightness slider will be disabled.