



ver1.0 - February 5, 2003

CEUSB2 LOADER DRIVER SPECIFICATIONS

CeUsb2 Loader Driver Specifications
Document for Software Programmers

CONTENTS

CONTENTS	1
1 INTRODUCTION.....	1
1.1 About This Documentation.....	1
1.2 Prerequisites	2
1.3 Distribution	2
1.4 Overview	2
2 LOADER DRIVER PROGRAMMING INTERFACE	3
2.1 Opening and closing the driver	3
2.2 I/O Control (IOCTL) Codes Interface	5
3 PROGRAMMING INTERFACE REFERENCE.....	7
3.1 I/O Control (IOCTL) Codes Reference.....	7
3.1.1 IOCTL_CEUSB2LD_GET_DRIVER_INFORMATION.....	7
3.1.2 IOCTL_CEUSB2LD_VENDOR_REQUEST	7
3.1.3 IOCTL_CEUSB2LD_CONFIGURE_DEVICE.....	7
3.2 Interface Structures.....	8
3.2.1 CEUSB2LD_DRIVER_INFO.....	8
3.2.2 CEUSB2LD_VENDOR_REQUEST_CONTROL.....	8

1 INTRODUCTION

1.1 *About This Documentation*

CeUsb2 devices have two drivers which are loaded by the system sequentially. This document is intended to give the specifications for the second one, the main driver. Before reading this document it would be helpful to check general CeUsb2 design guide documentation, CeUsb2dg.pdf, to understand the components which comprise the CeUsb2 software which runs in corporation with the loader driver.

1.2 Prerequisites

The CeUsb2 driver interface is accessible directly from C and C++ programming languages. The example codes given in this documentation is written in these languages. Therefore, an intermediate or upper knowledge of C/C++ and Windows programming (Win 32 SDK) is necessary to follow this documentation.

CeUsb2 boards are USB2.0 devices, so it is necessary to understand basics of the USB protocol in order to understand the USB communication and data transfer operations.

1.3 Distribution

After you install CeUsb2 software, you can find the header file for the CeUsb2 loader driver programming interface header files in CeUsb2\inc directory. These files are:

CeUsb2Ldlf.h: Main interface definition header file for CeUsb2 loader driver (IOCTL codes are in this file).

CeError.h: Contains status and error code definitions of the loader driver (This file also includes the status codes defined by the main driver and the CeUsb2 application programming interface).

These files include all IOCTL codes, structures and enumeration types to access the CeUsb2 loader driver.

CeUsb driver's executable is CeUsb2ld.sys which you can find in the driver directory.

1.4 Overview

Loader driver (CeUsb2Ld.sys) is a small kernel mode system executable which is designed with the WDM model of windows driver development kit (WIN DDK). It supports only minimum requirements of PNP (Plug & Play) protocol of Windows and doesn't support power management and WMI. If a CeUsb2 device is connected to the host, loader driver will be loaded by the system. It downloads the necessary firmware(s) on the board. If the board is unconfigured loader driver stays in the system enabling an input output control code (IOCTL) interface, with the minimum functionality to configure the board, and perform some USB vendor commands.

If the CeUsb2 board is configured previously, loader driver is unloaded and the final driver (CeUsb2.sys) is loaded by the system. In this case the user doesn't have the chance to access it. Most probably you will get a configured CeUsb2

device from Cesium GmbH; therefore, you will hardly need to understand this document and the programming interface of the loader driver.

Generally speaking, the loader driver has two main functionalities:

1 – Configuring the CeUsb2 device if it is in an unconfigured state. This process is explained very deeply in the general CeUsb2 design guide, CeUsb2dg.pdf.

2 – Downloading the firmware 8051 embedded executable(s) on the FX2 chip and run them (This process is done automatically).

Throughout this document, chapter 2 explains the usage of the loader driver's IOCTL interface with some example codes and chapter 3 is a reference for the CeUsb2 loader driver's programming interface. It explains the IOCTL codes together with the structures used with them.

2 LOADER DRIVER PROGRAMMING INTERFACE

All User mode access to the loader driver is through I/O Control calls. A user mode application first gets a handle to the device driver via a call to the Win32 function CreateFile. The user mode application then uses the Win32 function DeviceIoControl to submit an I/O control code and related input and output buffers to the driver through the handle returned by CreateFile.

2.1 Opening and closing the driver

Unlike the main CeUsb2 driver, loader driver uses the classical dos device naming technique to name its device objects and enable user to open handles to it. It can communicate with multiple unconfigured CeUsb2 devices. For each CeUsb2 device attached to the host, the driver creates a symbolic link of the form CeUsb2Ldi, where i is an instance index starting at 0. If there are 3 CeUsb2 devices attached to the host, then there will exist 3 symbolic links:

CeUsb2Ld0, CeUsb2Ld1 and CeUsb2Ld2.

CreateFile function is used together with the symbolic link to get a handle to the device object created by the device driver. After all processing is done with the driver the handle returned by the CreateFile function should be released and cleared by CloseHandle function.

Code example 2.1 shows how to open and close a CeUsb2 loader driver.

```
// Code example 2.1
```

```

// Opening and closing CeUsb2 loader driver

#include "..\inc\CeUsb2LdIf.h"
HANDLE ghDevice = NULL; // global handle to the device

// opens a loader driver with a given device index
DWORD Open(UINT DeviceIndex)
{
    char DeviceName[100]; // device name buffer

    if(!ghDevice)
    {
        // put the device name in the buffer
        sprintf(DeviceName, "\\.\%s%d", CEUSB2LD_DEVICE_NAME, DeviceIndex);

        //open driver
        ghDevice= CreateFile(
            DeviceName, // name of the device
            GENERIC_READ | GENERIC_WRITE, // access mode
            0, // share mode
            NULL, // security desc.
            OPEN_EXISTING, // how to create
            0, // file attributes
            NULL // template file
        );
        if ( ghDevice == INVALID_HANDLE_VALUE )
        {
            // unsuccessful, report the error
            return GetLastError();
        }
    }

    return 0; // return success code
}

// closes a previously opened CeUsb2 device
void Close(void)
{
    if(ghDevice) // test device handle
    {
        // close the handle
        CloseHandle(ghDevice);
        ghDevice = NULL;
    }
}

void main(void)
{
    DWORD dwStatus = Open(0); //open the device with index 0.
    if(dwStatus != 0)
    {
        //handle error
        return;
    }

    //...
}

```

```

    // do some communication with the device

    Close(); // close the device
}

```

2.2 I/O Control (IOCTL) Codes Interface

The I/O Control requests are submitted to the loader driver using the Win32 SDK function DeviceIoControl. The DeviceIoControl function is defined in SDK reference documentation as follows:

```

BOOL DeviceIoControl(
    HANDLE      hDevice,           // handle to device
    DWORD       dwIoControlCode,  // operation
    LPVOID      lpInBuffer,       // input data buffer
    DWORD       nInBufferSize,    // size of input data buffer
    LPVOID      lpOutBuffer,      // output data buffer
    DWORD       nOutBufferSize,   // size of output data buffer
    LPDWORD     lpBytesReturned,  // byte count
    LPOVERLAPPED lpOverlapped     // overlapped information
);

```

This function returns a nonzero value (TRUE) if it succeeds; otherwise it returns 0 (FALSE). Like most other Windows functions, the reason for the error can be retrieved with GetLastError function.

Following is the explanations of the DeviceIoControl function's parameters. Section 3.1, I/O Control Code (IOCTL) Reference, describes the I/O Control codes that can be passed to this function's dwIoControlCode parameter together with the usage of lpInBuffer, nInBufferSize, lpOutBuffer and nOutBufferSize arguments.

hDevice - Handle to the device on which to perform the operation. This handle is returned by CreateFile function.

dwIoControlCode – I/O control code (IOCTL) for the operation. This value identifies the specific operation to be performed and the type of device on which to perform it. (See Section 3.1).

lpInBuffer – Input data buffer (IOCTL specific).

nInBufferSize – Input data buffer size in bytes (IOCTL specific).

lpOutBuffer - Output data buffer (IOCTL specific).

nOutBufferSize - Output data buffer size in bytes (IOCTL specific).

lpBytesReturned - Pointer to a variable that receives the size, in bytes, of the data transferred by this IOCTL operation. This field is set by the CeUsb2 loader driver after a successful I/O operation.

lpOverlapped - Pointer to an OVERLAPPED structure. If hDevice was opened with the FILE_FLAG_OVERLAPPED flag, lpOverlapped must point to a valid OVERLAPPED structure. However, loader driver doesn't support asynchronous operation; therefore you can skip this field.

Code example 2.2 shows the simple usage of the CeUsb2 loader driver's IOCTL interface.

```
// Code example 2.2, cont. 2.1
// Opening and closing CeUsb2 loader driver

void main(void)
{
    DWORD dwStatus = Open(0); //open the device with index 0.
    if(dwStatus != 0)
    {
        //handle error
        return;
    }

    // Retrieve and display loader driver versioning information with
    // DeviceIoControl function

    CEUSB2LD_DRIVER_INFO DrvInfo; // driver information structure
    ULONG BytesTransferred;

    if(!DeviceIoControl(
        ghDevice, // device handle
        IOCTL_CEUSB2LD_GET_DRIVER_INFORMATION, //IOCTL code
        NULL, // no input buffer
        0, // input buffer size
        & DrvInfo, // buffer to hold the driver information
        sizeof (CEUSB2LD_DRIVER_INFO), // size of the output buffer
        & BytesTransferred, // actual bytes returned
        NULL // not overlapped
    ))
    {
        //handle error
        dwStatus = GetLastError();
        Close();
        return;
    }

    // display driver information
    printf("Driver executable %s\n, Driver version %u.%u\n, Build type string %s\n"
        DrvInfo.Name, DrvInfo.MajorVersion, DrvInfo.MinorVersion, DrvInfo.BuildTypeStr);

    Close(); // close the device
}
```

3 PROGRAMMING INTERFACE REFERENCE

3.1 I/O Control (IOCTL) Codes Reference

3.1.1 IOCTL_CEUSB2LD_GET_DRIVER_INFORMATION

Direction Input

Input buffer None

Input buffer length 0

Output buffer CEUSB2LD_DRIVER_INFO structure object.

Output buffer length Size of CEUSB2LD_DRIVER_INFO structure.

Bytes returned Size of CEUSB2LD_DRIVER_INFO structure.

Description Retrieves versioning information from the loader driver as driver name, version (major and minor) and build type string (debug or release). See also CEUSB2LD_DRIVER_INFO.

3.1.2 IOCTL_CEUSB2LD_VENDOR_REQUEST

Direction Input/Output

Input buffer CEUSB2LD_VENDOR_REQUEST_CONTROL structure object

Input buffer length Size of CEUSB2LD_VENDOR_REQUEST_CONTROL structure.

Output buffer Vendor request data buffer.

Output buffer length Vendor request data buffer size.

Bytes returned Actual amount of bytes transferred.

Description Performs a vendor specific USB request through CeUsb2 loader driver. Data buffer, data buffer length and bytes returned are vendor request specific. All vendor requests supported by the loader firmware with their buffer organization are documented in the generic firmware document.

3.1.3 IOCTL_CEUSB2LD_CONFIGURE_DEVICE

Direction Output

Input buffer Unsigned short variable (device id).

Input buffer length of unsigned short type, sizeof(USHORT).

Output buffer None

Output buffer length 0

Bytes returned 0

Description Configures a CeUsb2 device.

Notes This IOCTL code is unsupported is unsupported and should be used by Cesys only.

3.2 Interface Structures

3.2.1 CEUSB2LD_DRIVER_INFO

```
typedef struct _CEUSB2LD_DRIVER_INFO
{
    CHAR        Name[32];
    UCHAR       MajorVersion;
    UCHAR       MinorVersion;
    CHAR        BuildTypeStr[12];
}CEUSB2LD_DRIVER_INFO, *PCEUSB2_DRIVER_INFO;
```

Description : Loader river information structure.

Members :

Name – 32 byte driver executable name.

MajorVersion - Driver major version number.

MinorVersion - Driver minor version number.

BuildTypeStr – 12 byte driver build type string (i.e. "debug", "release").

3.2.2 CEUSB2LD_VENDOR_REQUEST_CONTROL

```
typedef struct _CEUSB2LD_VENDOR_REQUEST_CONTROL
{
    UCHAR       Request;
    UCHAR       Direction;
    USHORT      Value;
    USHORT      Index;
}CEUSB2LD_VENDOR_REQUEST_CONTROL, *PCEUSB2LD_VENDOR_REQUEST_CONTROL;
```

Description : Vendor request control structure for the loader driver.

Members :

Request – Vendor request code.

Direction – Vendor request direction (0 - OUT, 1 – IN).

Value – Vendor request specific value.

Index – Vendor request specific index.