



## **CONTENS**

<b>CONTENS.....</b>	<b>1</b>
<b>1 INTRODUCTION.....</b>	<b>2</b>
1.1 About This Documentation.....	2
1.2 Prerequisites.....	2
1.3 Distribution.....	2
1.4 Overview.....	3
<b>2 USING CEUSB2 DIAG.....</b>	<b>3</b>
2.1 CeUsb2 Diag Main Dialog.....	3
2.2 CeUsb2 Diag Command Dialog.....	6
<b>3 CEUSB2 DIAG REFERENCE.....</b>	<b>7</b>
3.1 CeUsb2 Diag Command Language.....	7
3.1.1 if, else and end_if.....	8
3.1.2 loop and end_loop.....	8
3.1.3 label and goto.....	9
3.2 CeUsb2 Diag Command Reference.....	9
3.2.1 h.....	9
3.2.2 e.....	9
3.2.3 edlg.....	10
3.2.4 v.....	10
3.2.5 drv.....	10
3.2.6 fw.....	10
3.2.7 api.....	10
3.2.8 prop.....	11
3.2.9 desc.....	11
3.2.10 feature.....	11
3.2.11 cpu.....	11
3.2.12 gpif.....	11
3.2.13 serial.....	11
3.2.14 fpga.....	11
3.2.15 cf.....	11
3.2.16 clf.....	12
3.2.17 rf.....	12
3.2.18 ven.....	12
3.2.19 gc.....	13
3.2.20 sc.....	14
3.2.21 gi.....	14

3.2.22	si .....	14
3.2.23	frn.....	14
3.2.24	gps.....	15
3.2.25	sps .....	15
3.2.26	br.....	15
3.2.27	bw .....	15
3.2.28	sr.....	15
3.2.29	sw .....	16
3.2.30	print.....	16
3.2.31	mb.....	16
3.2.32	clrscr .....	16
3.2.33	out_mode.....	16
3.2.34	sleep.....	17

## 1 INTRODUCTION

### 1.1 About This Documentation

This documentation explains the usage of the diagnostic program CeUsb2Diag.exe which is designed to test and control the CeUsb2 types of boards.

### 1.2 Prerequisites

Before reading this documentation it would be helpful to check general CeUsb2 design guide documentation, CeUsb2dg.pdf, to understand the components which comprise the complete CeUsb2 software.

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 executable file of the CeUsb2 diagnostic program, CeUsb2Diag.exe, in the bin directory. Windows dynamic link libraries (DLLs) CeUsb2Api.dll and CeUsb2Mfc.dll are required to be in the same directory.

## **1.4 Overview**

CeUsb2 diagnostic program is designed with the MFC (Microsoft Foundation Classes) using MS Visual C++ 6.0. It uses the CeUsb2 application programming interface (API) and graphical user programming interface (GUI) of the CeUsb2 software.

Chapter 2 explains the usage of the diagnostic program with its controls and buttons.

Chapter 3 explains the simple command language of the diagnostic program. It also gives a brief explanation of the commands which is defined by the program.

## **2 USING CEUSB2 DIAG**

### **2.1 CeUsb2 Diag Main Dialog**

CeUsb2Diag consists of a command line edit box and several buttons. Command prompt covers all the functionalities of the buttons and extends them with some more CeUsb2 commands.

Figure 2.1 is a screenshot of the CeUsb2Diag program just after it is started.

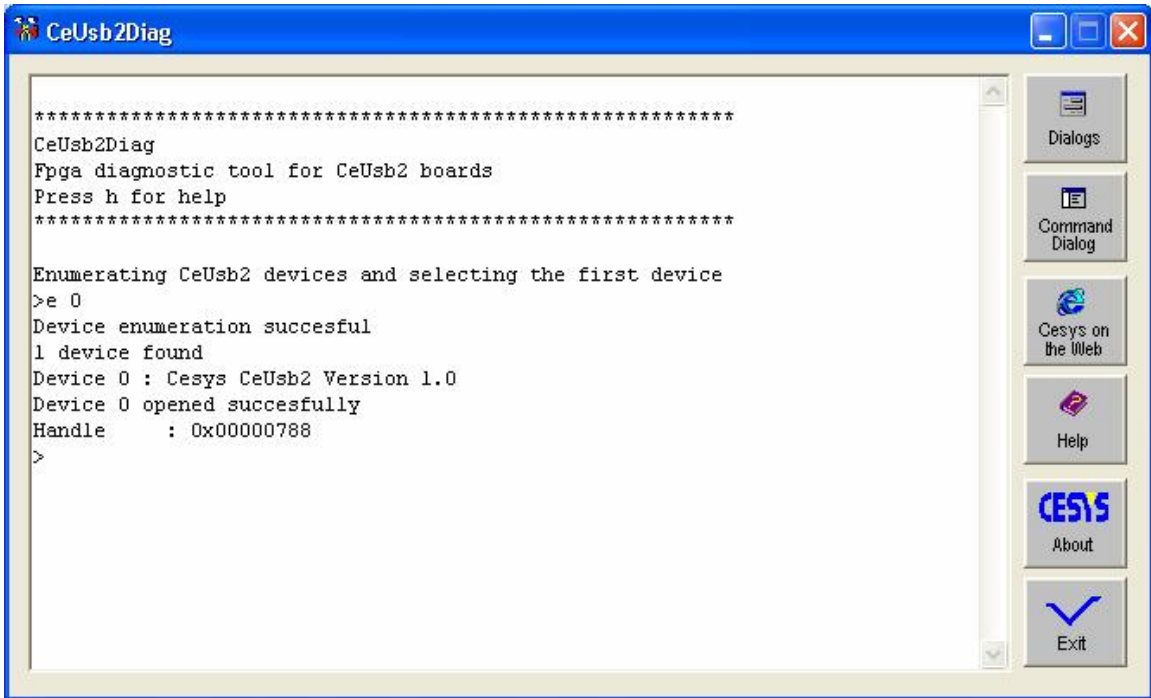


Figure 2.1 – CeUsb2Diag main dialog

Users can write their commands in command line edit box and execute them by pressing the ENTER key. It also has command history feature. Previously stored commands can be retrieved with arrow up and down keys.

Exit button closes the CeUsb2 diagnostic program.

About button displays the About Box of the program (see figure 2.2).

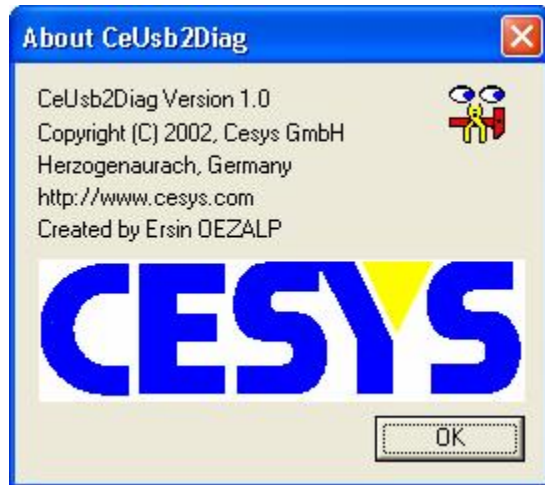
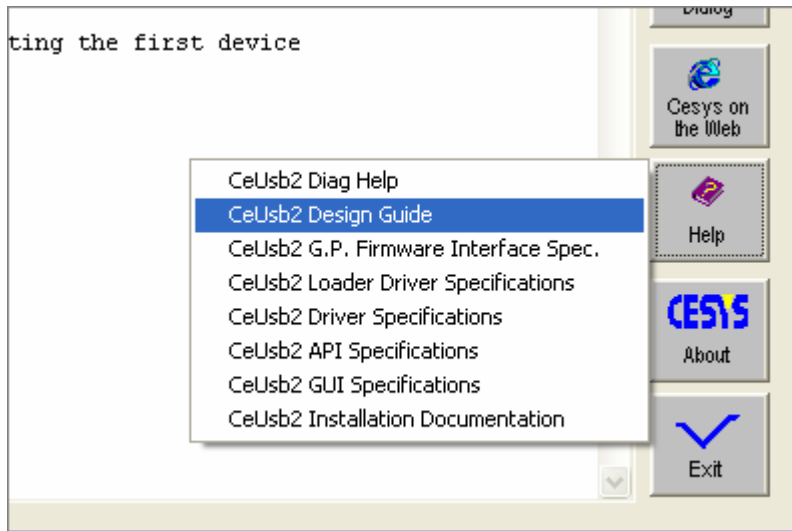


Figure 2.2 – CeUsb2Diag About Box

Help button opens a drop down menu (see figure 2.3) on which the user can select CeUsb2 help documents and open them. Notice that all help documents has pdf extensions, therefore you need a pdf viewer program installed in your system (you can download Adobe's free Acrobat Reader program from <http://www.adobe.com> web-site).



*Figure 2.3 – Drop-down Help Menu*

Cesys on the web button opens your current internet browser with the web-site of Cesys GmbH (<http://www.cesys.com>).

Command Dialog button opens the command interpreter dialog (see section 2.2).

Dialogs button opens a drop down menu (see figure 2.4) on which the user can open some CeUsb2 control dialogs. These dialogs are implemented in CeUsb2 graphical user programming interface (GUI), and explained in GUI document, CeUsb2gui.pdf.

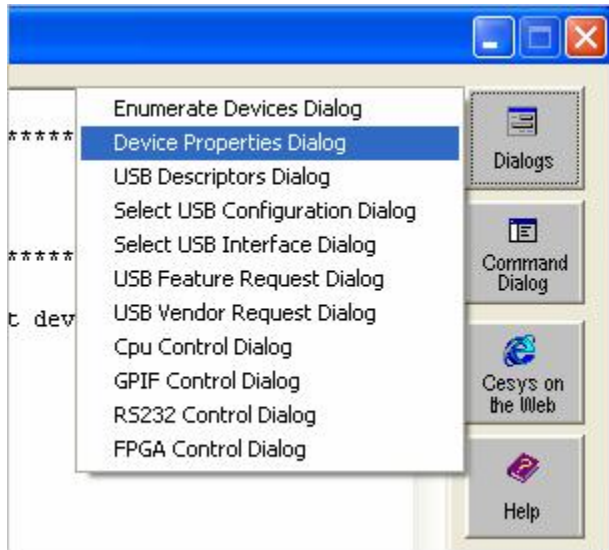
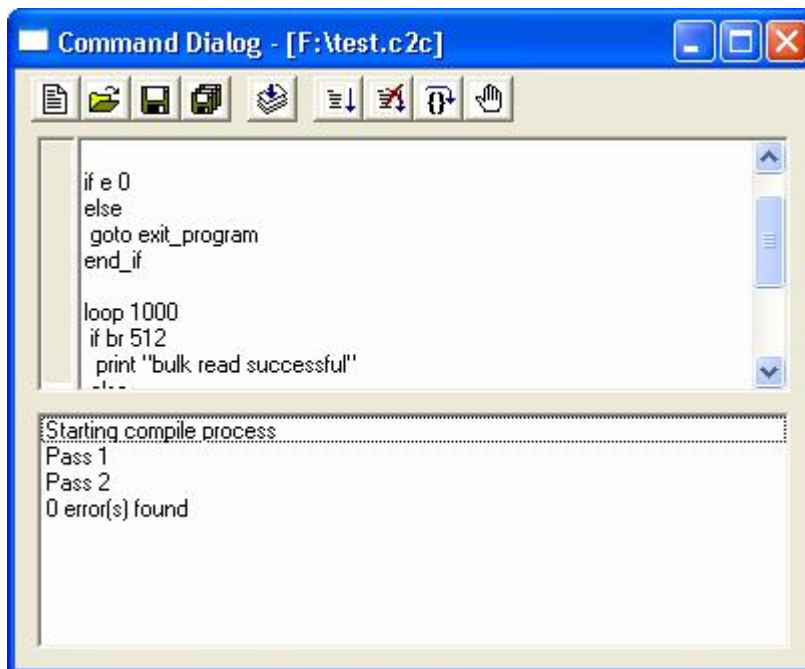


Figure 2.4 – Dialogs drop-down menu

## 2.2 CeUsb2 Diag Command Dialog

CeUsb2Diag command dialog can be opened by clicking the Command Dialog button on the main dialog of the program. It can open, save, compile and run ASCII based CeUsb2 command files with extensions txt and c2c.

Figure 2.5 is a screenshot of the command dialog.



*Figure 2.5 – CeUsb2 command dialog*

Upper part of the dialog is a rich edit box which displays the most currently opened command file. The user can change the command files with this simple editor. It also has copy and paste features.

Bottom part of the dialog is a list box which displays the compile and run time messages. “clrscr” command clears this list box.

Command dialog is controlled with its toolbar buttons. If the mouse pointer waits for a while on a button of the toolbar, a short tool-tip explanation will be seen on the screen. Following are the short descriptions of this buttons’ functionalities:

New Command File button closes the currently opened command file and creates a new unnamed and empty one.

Open Command File button opens a previously stored command file. It displays the open file dialog of the system on which the user can select the command files with extensions txt and c2c.

Save Command File button saves the currently opened command file. If the file doesn’t have a name, it displays the save file dialog of the system on which the user can select a file name.

Save Command File As button saves the currently opened command file with a different name. It displays the save file dialog of the system on which the user can select a file name with extensions txt and c2c.

Compile button parses and compiles the command file. If there are some compile errors, they are displayed in the output messages list box.

Run button executes the command file.

Stop button stops the execution of the command line if it is running currently.

Single step and Breakpoint buttons don’t have any functionality in this version of the diagnostic program. They will be implemented in the near future.

## **3 CEUSB2 DIAG REFERENCE**

### **3.1 CeUsb2 Diag Command Language**

CeUsb2 command dialog has a text base simple command language in which the users can write a series of commands with some control statements. Following is the explanation of these control statements.

### 3.1.1 if, else and end\_if

#### Usage

```
if command0
    command1
    command2
    ....
else
    command3
    command4
    ....
end_if
```

**Description** if statement controls conditional branching. If command0 succeeds the body commands (command1, command2 ...) are executed; if not commands in else body (command3, command4 are executed.

If statements can be used nested also.

#### Example

```
if e 0
    o
    print "device enumeration successfull"
else
    print "device enumeration failed"
    goto exit_program
end_if
```

### 3.1.2 loop and end\_loop

#### Usage

```
loop [loop count]
    command1
    command2
    ....
end_loop
```

**Description** loop statement repeats a statement or compound statements specified number of times or forever. [loop count] argument determines the number of loop rotations. If no [loop count] is not specified, then the loop is infinite.

#### Example

```
loop 1000
    if br -p1 512
```

```
        else
            print "unable to read from bulk pipe 1"
            goto exit_program
        end_if
end_if
```

### 3.1.3 label and goto

#### Usage

```
goto label0
```

```
command
```

```
....
```

```
label label0
```

```
command
```

```
.....
```

**Description** The goto statement transfers the execution to a label.

#### Example

```
loop 1000
    if br -p1 512
    else
        print "unable to read from bulk pipe 1"
        goto exit_program
    end_if
end_if

.....

label exit_program
program "abnormal termination"
```

## 3.2 CeUsb2 Diag Command Reference

### 3.2.1 h

**Usage** h

**Description** Displays the help screen with the usage of the commands.

### 3.2.2 e

**Usage** e  
e [device index]

**Description** Enumerates the CeUsb2 devices in the system. The second version selects the device with the index [device index] for further operation, and tries to open it.

**Arguments**  
[device index] – Index of the device to select.

### 3.2.3 edlg

**Usage** edlg

**Description** Opens the CeUsb2 device enumeration dialog.

### 3.2.4 v

**Usage** v

**Description** Displays versioning information for CeUsb2Diag program.

### 3.2.5 drv

**Usage** drv

**Description** Displays versioning information for currently installed CeUsb2 driver.

### 3.2.6 fw

**Usage** fw

**Description** Displays versioning information for currently running CeUsb2 firmware application.

### 3.2.7 api

**Usage** api

**Description** Displays versioning information for the current CeUsb2 application programming interface (API).

### **3.2.8 prop**

**Usage** prop

**Description** Opens CeUsb2 device properties dialog.

### **3.2.9 desc**

**Usage** desc

**Description** Opens USB descriptors dialog.

### **3.2.10 feature**

**Usage** feature

**Description** Opens USB feature request dialog.

### **3.2.11 cpu**

**Usage** cpu

**Description** Opens CeUsb2 CPU control dialog.

### **3.2.12 gpif**

**Usage** gpif

**Description** Opens CeUsb2 GPIF control dialog.

### **3.2.13 serial**

**Usage** serial

**Description** Opens RS232 – Serial interface control dialog.

### **3.2.14 fpga**

**Usage** fpga

**Description** Opens FPGA control dialog.

### **3.2.15 cf**

**Usage** cf "file name" -c[control pipe] -p[pipe number] -g[gpif dynamic]

**Description** Configures the FPGA with a given FPGA configuration file.

**Arguments**

"file name" – (Optional, string) FPGA configuration file path. If this file could not be found, or if this argument is not used, the file open dialog of Windows system is displayed, in which the user can determine the FPGA configuration file.

[control pipe] – (Optional, unsigned integer, default 0) Determines the USB pipe which will be used for FPGA configuration. If this argument is other than 0, the default control pipe will be used. Otherwise another bulk or isochronous pipe will be searched.

[pipe number] – (Optional, signed integer, default -1) Determines the pipe number on which FPGA configuration data will be send to the firmware. If this member is not used or is smaller than 0, then the best suitable USB communication pipe will be searched automatically. If control pipe argument is not 0, this argument doesn't have any effects.

[gpif dynamic] - (Optional, unsigned integer, default 1) If this argument is other than 0, then internally stored GPIF Initialization data and GPIF Waveform data will be used dynamically for FPGA configuration. All settings will be restored after the configuration process is finished. If control pipe argument is not 0, this argument doesn't have any effects.

**3.2.16**      **clf**

**Usage**      clf

**Description** Clears the current FPGA configuration.

**3.2.17**      **rf**

**Usage**      rf

**Description** Sends a reset signal to the currently running FPGA design.

**3.2.18**      **ven**

**Usage**      ven

ven -t[request type] -r[request recipient]  
          -c[request code] -d[direction] -v[value] -i[index]  
          -l[data length] [data bytes ...]

**Description** The first version of this command opens the USB vendor or class request dialog.

The second version performs a vendor or class specific USB request through CeUsb2 devices with the given parameters.

### Arguments

[request type] – (Optional, unsigned integer) Type of the request. If this argument is not specified then internal vendor request function is used. Otherwise vendor or class request is used with the type value. Possible type values are:

- 0 – Vendor request
- 1 or others – Class request

[request recipient] – (Optional, unsigned integer) Request recipient. Valid only if the request type argument is specified. Possible values are:

- 0 – Recipient is device.
- 1 – Recipient is interface.
- 2 – Recipient is endpoint.
- 3 or others – Recipient is another device defined target.

[request code] – ( Hexadecimal ) 1 byte request code.

[direction] – (Optional, unsigned integer, default 0) Request direction. Possible values are:

- 0 – OUT, data is transferred from host to the device.
- 1 or others – IN, data is transferred from device to the host.

[value] – (Optional, hexadecimal, default 0) 2 byte vendor or class request specific value.

[index] – (Optional, hexadecimal, default 0) 2 byte vendor or class request specific index.

[data length] – (Unsigned integer between 0 and 64, default 0) Total number of bytes that will be transferred with the request. This value ignored if the transfer direction is OUT.

[data bytes.....] – Hexadecimal 1 byte data bytes.

### 3.2.19 gc

**Usage** gc

**Description** Gets the current USB configuration number which is active on the CeUsb2 board.

### 3.2.20      **sc**

**Usage**        `sc`  
                 `sc [configuration number]`

**Description** The first version of this command opens the Select USB configuration dialog.  
The second version selects an USB configuration on the CeUsb2 device.

#### **Arguments**

[configuration number] – Configuration number to select.

### 3.2.21      **gi**

**Usage**        `gi [interface number]`

**Description** Gets the current alternate setting for a given interface.

#### **Arguments**

[interface number] – Interface number.

### 3.2.22      **si**

**Usage**        `si`  
                 `si [interface number] [alternate setting] [maximum transfer sizes ...]`

#### **Description**

The first version of this command opens the Select USB interface dialog.  
The second version selects an USB interface and alternate setting for the current USB configuration on CeUsb2 board.

#### **Arguments**

[interface number] – (Unsigned integer) Interface number to select.  
[alternate setting] – (Unsigned integer) Alternate setting number to select.  
[maximum transfer sizes ...] (Optional, hexadecimal, unsigned long)  
Maximum transfer sizes for the USB pipes.

### 3.2.23      **frn**

**Usage**        `frn`

**Description** Retrieves the current frame number on the USB bus.

### 3.2.24 **gps**

**Usage** `gps`

**Description** Retrieves the current power state of the device. Possible power states are:

- 0 – Power state unspecified.
- 1 – Power state D0.
- 2 – Power state D1.
- 3 – Power state D2.
- 4 – Power state D3.
- 5 – Power state maximum.

### 3.2.25 **sps**

**Usage** `sps [power state]`

**Description** Sets the current power state of the device.

**Arguments**

[power state] – New power state of the device. See `gps` for possible power states.

### 3.2.26 **br**

**Usage** `br -p[pipe number] [length]`

**Description** Reads specified amount of data from a bulk pipe.

**Arguments**

- [pipe number] – (Optional, unsigned integer, default 1) Bulk pipe number.
- [length] – Number of bytes to read.

### 3.2.27 **bw**

**Usage** `bw -p[pipe number] [data bytes ...]`

**Description** Writes specified amount of data to a bulk pipe.

**Arguments**

- [pipe number] – (Optional, unsigned integer, default 0) Bulk pipe number.
- [data bytes ...] – (1 byte, hexadecimal) Data bytes to write.

### 3.2.28 **sr**

**Usage** `sr [address] [length]`

**Description** Reads specified amount of data over control pipe with GPIF single transitions.

## Arguments

[address] – (2 bytes hexadecimal) Address (offset) to read.  
[length] – Number of bytes to read.

### 3.2.29 sw

**Usage** sw [address] [data bytes ...]

**Description** Writes specified amount of data over control pipe with GPIF single transitions.

#### Arguments

[address] – (2 bytes hexadecimal) Address (offset) to write.  
[data bytes ...] – (1 byte, hexadecimal) Data bytes to write.

### 3.2.30 print

**Usage** print "text"

**Description** Displays a text message in the current output window.

#### Arguments

"text" –Text message.

### 3.2.31 mb

**Usage** mb "text"

**Description** Displays a Windows message box with a given text.

#### Arguments

"text" – Message box text.

### 3.2.32 clrscr

**Usage** clrscr

**Description** Clears the current output window's contents.

### 3.2.33 out\_mode

**Usage** out\_mode  
ot\_mode [mode]

**Description** The first version of this function shows the output mode of the current output window.

The second version sets the output mode of the current output window. Possible output modes are:

0 – Dump mode. All internal buffer dump requests will be processed.

1 or others – Silent mode. All internal buffer dump requests will be skipped.

### **3.2.34      sleep**

**Usage**          sleep [milliseconds]

**Description** Delays the execution of the commands.

**Arguments**

[milliseconds] – Amount of time to delay in milliseconds.