

CEDOSYS

February 2007

Documentation Windows 32-Bit DLL





cetoni GmbH

Am Wiesenring 6

D- 07554 Korbußen

Tel.: +49 (0) 36602 338-0

Fax: +49 (0) 36602 338-11

E-Mail: info@cetoni.de

Internet: www.cetoni.de

1 Table of contents

1 Table of contents	3
2 Version History	5
3 Introduction	6
4 Including Library Functions	7
4.1 General Information	7
4.2 Including	7
5 ceDOSYS Command Set	9
5.1 The APIs	9
6 High level API	10
6.1 Groups of functions	10
6.2 Initialisation	10
6.2.1 Open Device	10
6.2.2 Close Device	11
6.2.3 Register Event Callback	12
6.3 Configuration	13
6.3.1 Get Syringe Parameters	13
6.3.2 Set Syringe Parameters	14
6.3.3 Get Flow Rate Maximum	14
6.3.4 Get Syringe Level Maximum	15
6.4 Dosing.....	17
6.4.1 Dose Volume	17
6.4.2 Set Syringe Level	18
6.4.3 Generate Flow	19
6.4.4 Stop	20
6.4.5 Stop All Units	20
6.5 Dosing Info.....	21
6.5.1 Get Syringe Level Is	21
6.5.2 Get Flow Rate Is.....	22
6.5.3 Is Dosing Finished	23
6.6 Valve Control	24
6.6.1 Switch Valve	24
6.6.2 Is Valve Switched To Output	25
6.7 Helper Functions	26
6.7.1 Convert Volume into Device Unit Value	26
6.7.2 Convert Device Unit Value into Volume	27

6.7.3	Convert Flow into Device Unit Value	27
6.7.4	Convert Device Unit Value into Flow	28
6.8	Flow and Volume Units	30
6.8.1	Flow Unit Identifier	30
6.8.2	Volume Unit Identifier.....	30
7	Low level API	31
7.1	Introduction	31
8	DLL Integration into Borland BDS2006 C++	32

2 Version History

Date	Version	Documentation	Description
15.02.2007	Ver 1.01	February 2007	<ul style="list-style-type: none">• First Library Version

3 Introduction

This documentation “Windows 32-Bit DLL” provides the instructions for the implemented functions. The library is arranged in groups of functions and helps to simplify the programming of the control software based on Windows. This document describes the interface between a program and the Windows DLL (Dynamic Link Library).

BECAUSE THIS LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THIS LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THIS LIBRARY IS WITH YOU. SHOULD THIS LIBRARY PROVE DEFECTIVE OR INSUFFICIENT, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

The latest edition of these “Windows 32-Bit DLL”, additional documentation and software to the ceDOSYS dosing system may also be found in the internet under <http://www.cetoni.de/service/downloads>.

4 Including Library Functions

4.1 General Information

The library *cedosys_api.dll* is an implementation of the ceDOSYS command set for the communication between ceDOSYS and a personal computer. Using this library is a simple way to develop your own application.

This library is running on each windows 32-bit operating system. You can include it into different programming environments. All the ceDOSYS commands are implemented and they can be called directly from your own program. You don't have to care about the protocol details. The only thing you have to ensure is a properly connected and configured dosing system.

The library *cedosys_api.dll* offers the whole set of ceDOSYS commands. Executing ceDOSYS commands, managing configuration parameters, executing dosing tasks and handling low level communication with the ceDOSYS device is the business of this library.

4.2 Including

The following chapter describes how to include all library functions in your own windows program. The way how you do that, depends on the compiler and on the programming language you use. Thus we are going to explain the procedure to include the library based on some examples of the most popular programming languages (Chapter 8). For several high-level programming languages are further documentations and its own examples available (Chapter 8).

In order to have a correctly working communication, you have to include the library

cedosys_api.dll

to your programming environment. You have to copy these files to the working directory of your system.

To open the library *cedosys_api.dll* you have to use the function [CCS_OpenDevice\(\)](#). You have to do this before you can execute any ceDOSYS command. At the end of your

program you have to call [CCS_CloseDevice\(\)](#). For more detailed information about the initialisation procedure, have a look at the chapter '[Initialisation](#)' (Chapter 6).

Use the calling convention `__stdcall` for this library. This convention is managing how the parameters are put on the stack and who is responsible to clean the stack after the function execution.

5 ceDOSYS Command Set

5.1 The APIs

The ceDOSYS Command Set provides two different APIs which are implemented in the *cedosys_api.dll*. The first API is a low level API that is a simple wrapper for the serial protocol. For each serial command the low level API offers a function that creates and sends an appropriate message. It handles the complete serial protocol.

The second API is a high level API that is based upon the low level API. It offers a higher degree of abstraction and uses the low level API to perform different dosing tasks. Normally an application should use the high level API and should only call into the low level API if the higher level does not offer an appropriate function for a certain task.

[High level API](#)

[Low level API](#)

6 High level API

6.1 Groups of functions

The ceDOSYS high level API defines following groups:

[Initialisation](#)

[Configuration](#)

[Dosing](#)

[Dosing info](#)

[Valve control](#)

[Helper functions](#)

6.2 Initialisation

This group defines all required functions to initialize a correct communication to the device:

[Open Device](#)

[Close Device](#)

[Register Event Callback](#)

6.2.1 Open Device

Function

```
HANDLE CCS_OpenDevice ( char SerialPort,  
                        BOOL bShowSetupDlg  
                        )
```

Description

Function "CCS_OpenDevice" opens the connection for sending and receiving commands and initializes device.

Parameters

SerialPort	char	Number of serial port to connect to device. This parameter is only valid if bShowSetupDlg is False.
bShowSetupDlg	BOOL	True: Display a setup window and let the user configure the serial communication parameters. False: Use the serial port specified in SerialPort to connect to device

Return Parameters

Return value	HANDLE	HANDLE for device access. Nonzero if successful; otherwise 0
--------------	--------	--

Related Functions

[Close Device](#)

6.2.2 Close Device

Function

```
BOOL CCS_CloseDevice( HANDLE hDevice )
```

Description

Function "CCS_ClosDevice" closes the connection to the device, frees resources and releases the interface for other applications.

Parameters

hDevice	HANDLE	Handle for device access
---------	--------	--------------------------

Return Parameters

Return value	long	Error code. 0 if successful; otherwise < 0
--------------	------	--

Related Functions

[Open Device](#)

6.2.3 Register Event Callback

Function

```
long CCS_RegisterEventCallback( HANDLE          hDevice,  
                               CCS_AppEventHandler CallbackFn  
                               )
```

Description

Register call-back function for handling of asynchronous events.

Parameters

hDevice	HANDLE	Valid handle for device access
CallbackFn	CCS_AppEventHandler	User defined call-back function (event handler)

Return Parameters

Return value	long	Error code. 0 if successful; otherwise < 0
--------------	------	--

6.3 Configuration

This group defines all required functions for reading and writing configuration parameters of the dosing system:

[Get Syringe Parameters](#)

[Set Syringe Parameters](#)

[Get Flow Rate Maximum](#)

[Get Syringe Level Maximum](#)

6.3.1 Get Syringe Parameters

Function

```
long CCS_GetSyringeParam ( HANDLE          hDevice ,
                          unsigned char   DosingUnit ,
                          double *        pLenInMillimetres ,
                          double *        pVolInMicroLitres
                          )
```

Description

Reads syringe configuration parameters. Syringe parameters are required for proper conversion from device units for speed and distance into units for volume and flow rate.

Parameters

hDevice	HANDLE	Valid handle for device access
DosingUnit	unsigned char	Number of dosing unit to query (0 - 3)

Return Parameters

pLenInMillimetres	double*	Length of syringe that contains the volume in pVolInMicroLitres
pVolInMicroLitres	double*	Volume in micro litres
Return value	long	Error code. 0 if successful; otherwise < 0

Related Functions

[Set Syringe Parameters](#)

6.3.2 Set Syringe Parameters

Function

```
long CCS_SetSyringeParam( HANDLE      hDevice,
                          unsigned char DosingUnit,
                          double       LenInMillimetres,
                          double       VolInMicrolitres
                          )
```

Description

Writes syringe configuration parameters. These parameters are required for proper value conversion and should be changed each time the syringe changes.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to change
LenInMillimetres	double	Length of syringe that contains the volume in VolInMicroLitres
VolInMicroLitres	double	Volume in micro litres

Return Parameters

Return value	long	Error code. 0 if successful, otherwise < 0
--------------	------	--

Related Functions

[Get Syringe Parameters](#)

6.3.3 Get Flow Rate Maximum

Function

```
long CCS_GetFlowRateMax ( HANDLE      hDevice,
                          unsigned char DosingUnit,
                          double *     pFlowRateMax,
                          unsigned char FlowUnit
                          )
```

Description

The function reads the maximum flow rate that should be used by all dosing commands. This flow rate depends on the mechanical configuration (i.e. gear) and it also depends on the syringe configuration.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to query
FlowUnit	unsigned char	Flow unit identifier (nl/s, nl/min, µl/s...)

Return Parameters

pFlowRateMax	double*	Stores maximum flow rate value
Return value	long	Error code. 0 if successful, otherwise < 0

Related Functions

[Get Syringe Parameters](#)

6.3.4 Get Syringe Level Maximum

Function

```
long CCS_GetSyringeLevelMax ( HANDLE      hDevice,
                              unsigned char DosingUnit,
                              double *     pSyringeLevelMax,
                              unsigned char VolUnit
                              )
```

Description

This function reads the maximum syringe level. This level defines the upper limit for dosing functions. The maximum syringe level value depends on syringe configuration and configured volume unit.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to query
VolUnit	unsigned char	Volume unit identifier (nl, µl, ml) for value in



		pSyringeLevelMax
--	--	------------------

Return Parameters

pSyringeLevelMax	double*	Maximum syringe level. The value depends on syringe configuration and configured volume unit.
Return value	long	Error code. 0 if successful, otherwise < 0

Related Functions

[Get Syringe Parameters](#)

6.4 Dosing

This group defines all required functions for execution of different dosing tasks like dosing a certain volume or generating a continuous flow:

[Dose Volume](#)

[Set Syringe Level](#)

[Generate Flow](#)

[Stop](#)

[Stop All Units](#)

6.4.1 Dose Volume

Function

```
long CCS_DoseVolume( HANDLE          hDevice,
                    unsigned char DosingUnit,
                    double *       pVolume,
                    unsigned char VolUnit,
                    double *       pFlowRate,
                    unsigned char FlowUnit
                    )
```

Description

This function doses a defined volume with a defined flow rate. Flow rate should not exceed the [maximum flow rate](#) value.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to operate
pVolume	double*	Volume to dose in volume unit format.
VolUnit	unsigned char	Volume unit identifier for volume in pVolume
pFlowRate	double*	Continuous flow rate to generate in flow rate unit format. A positive flow rate indicates delivery of reagent and a negative value indicates a take up of reagent.
FlowUnit	unsigned char	Flow unit identifier for flow value in pFlowRate

Return Parameters

pVolume	double*	The realizable volume value.
pFlowRate	double*	The realizable flow rate.
Return value	long	Error code; 0 if successful, otherwise < 0

Related Functions

[Stop](#)

6.4.2 Set Syringe Level

Function

```
long CCS_SetSyringeLevel ( HANDLE          hDevice ,
                          unsigned char    DosingUnit ,
                          double *        pVolume ,
                          unsigned char    VolUnit ,
                          double *        pFlowRate ,
                          unsigned char    FlowUnit
                          )
```

Description

Moves the pusher until syringe content reaches a certain fill level.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to operate
pVolume	double*	Syringe filling level in volume unit format.
VolUnit	unsigned char	Volume unit identifier for volume in pVolume
pFlowRate	double*	Continuous flow rate to generate in flow rate unit format. A positive flow rate indicates delivery of reagent and a negative value indicates a take up of reagent.
FlowUnit	unsigned char	Flow unit identifier for flow value in pFlowRate

Return Parameters

pVolume	double*	The realizable volume value.
pFlowRate	double*	The realizable flow rate.

Return value	long	Error code; 0 if successful, otherwise < 0
---------------------	------	--

Related Functions

[Stop](#)

6.4.3 Generate Flow

Function

```
long CCS_GenerateFlow ( HANDLE          hDevice,
                        unsigned char  DosingUnit,
                        double *       pFlowRate,
                        unsigned char  FlowUnit
                        )
```

Description

The function generates a continuous flow with a certain flow rate. The device moves the pusher until it reaches its limits or until application calls [CCS_Stop\(\)](#).

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to operate
pFlowRate	double*	Continuous flow rate to generate in flow rate unit format. A positive flow rate indicates delivery of reagent and a negative value indicates a take up of reagent.
FlowUnit	unsigned char	Flow unit identifier for flow value in pFlowRate

Return Parameters

pFlowRate	double*	The realizable flow rate.
Return value	long	Error code; 0 if successful, otherwise < 0

Related Functions

[Stop](#)

6.4.4 Stop

Function

```
long CCS_Stop ( HANDLE      hDevice ,
               unsigned char DosingUnit
               )
```

Description

This function stops dosing of one single dosing unit.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to stop

Return Parameters

Return value	long	Error code; 0 if successful, otherwise < 0
--------------	------	--

Related Functions

[Stop All Dosing Units](#)

6.4.5 Stop All Units

Function

```
long CCS_StopAllUnits ( HANDLE hDevice )
```

Description

The function stops dosing of all dosing units.

Parameters

hDevice	HANDLE	Handle for device access
---------	--------	--------------------------

Return Parameters

Return value	long	Error code; 0 if successful, otherwise < 0
--------------	------	--

6.5 Dosing Info

This group defines all required functions for query of different dosing parameters and device states:

[Get Syringe Level Is](#)

[Get Flow Rate Is](#)

[Is Dosing Finished](#)

6.5.1 Get Syringe Level Is

Function

```
long CCS_GetSyringeLevelIs ( HANDLE          hDevice,
                             unsigned char    DosingUnit,
                             double *        pVolume,
                             unsigned char    VolUnit
                             )
```

Description

Returns actual syringe fill level.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to query
VolUnit	unsigned char	Volume unit identifier for value in pVolume

Return Parameters

pVolume	double*	Syringe filling level in volume unit format.
Return value	long	Error code; 0 if successful, otherwise < 0

6.5.2 Get Flow Rate Is

Function

```
long CCS_GetFlowRateIs ( HANDLE          hDevice,  
                        unsigned char DosingUnit,  
                        double *      pFlowRateIs,  
                        unsigned char FlowUnit  
                        )
```

Description

The function returns the actual flow rate of single dosing unit.

Parameters

<code>hDevice</code>	HANDLE	Handle for device access
<code>DosingUnit</code>	unsigned char	Number of dosing unit to query
<code>FlowUnit</code>	unsigned char	Flow unit identifier for flow value in <code>pFlowRate</code>

Return Parameters

<code>pFlowRateIs</code>	double*	Actual flow rate in flow rate unit format.
Return value	long	Error code; 0 if successful, otherwise < 0

6.5.3 Is Dosing Finished

Function

```
long CCS_IsDosingFinished ( HANDLE      hDevice,  
                           unsigned char DosingUnit  
                           )
```

Description

Returns 1 if dosing is finished. Dosing is finished if the device is stopped. That does not indicate that dosing task was successfully finished.

Parameters

<i>hDevice</i>	HANDLE	Handle for device access
<i>DosingUnit</i>	unsigned char	Number of dosing unit to query

Return Parameters

Return value	long	0 – dosing is active – pusher is moving 1 – dosing is finished – dosing unit is stopped < 0 – Error code
---------------------	------	--

6.6 Valve Control

This group defines all required functions for valve control, configuration and reading of valve states:

[Switch Valve](#)

[Is Valve Switched To Output](#)

6.6.1 Switch Valve

Function

```
long CCS_SwitchValve ( HANDLE          hDevice,
                      unsigned char DosingUnit,
                      unsigned char SwitchToOutput
                      )
```

Description

Switches valve between input and output. Switching valve to input is required for taking up reagent and switching valve to output is required for delivering reagent.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to change
SwitchToOutput	unsigned char	0 – Switches valve to input 1 – Switches valve to output

Return Parameters

Return value	long	Error code; 0 if successful, otherwise < 0
--------------	------	--

Related Functions

[Is Valve Switched To Output](#)

6.6.2 Is Valve Switched To Output

Function

```
long CCS_IsValveSwitchedToOutput( HANDLE      hDevice,  
                                  unsigned char DosingUnit  
                                  )
```

Description

The function returns actual state of valve.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to query

Return Parameters

Return value	long	0 – Valve is switched to input 1 – Valve is switched to output < 0 – Error code
--------------	------	---

Related Functions

[Switch Valve](#)

6.7 Helper Functions

This group defines a set of tool and helper functions:

[Convert Volume into Device Unit Value](#)

[Convert Device Unit Value into Volume](#)

[Convert Flow into Device Unit Value](#)

[Convert Device Unit Value into Flow](#)

6.7.1 Convert Volume into Device Unit Value

Function

```
long CCS_ConvVolIntoDevUnit( HANDLE      hDevice,
                             unsigned char DosingUnit,
                             double       Volume,
                             unsigned char VolUnit,
                             long *      pDevValue
                             )
```

Description

This function converts volume unit into device specific format (increments). The conversion depends on syringe configuration parameters, which belong to a certain dosing unit. Therefore a dosing unit number is required for proper conversion.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit
Volume	double	The volume value in volume unit format.
VolUnit	unsigned char	Volume unit identifier for value in Volume.

Return Parameters

pDevValue	long*	Stores converted value. The result is a value in a device specific format (increments)
-----------	-------	--

6.7.2 Convert Device Unit Value into Volume

Function

```
long CCS_ConvDevUnitIntoVol( HANDLE      hDevice,
                             unsigned char DosingUnit,
                             long         DevValue,
                             unsigned char VolUnit,
                             long *      pVolume
                             )
```

Description

This function converts a value from internal device unit format (increments) into volume. The conversion depends on syringe configuration parameters, which belong to a certain dosing unit. Therefore a dosing unit number is required for proper conversion.

Parameters

<i>hDevice</i>	HANDLE	Handle for device access
<i>DosingUnit</i>	unsigned char	Number of dosing unit
<i>DevValue</i>	double	The value in device unit format
<i>VolUnit</i>	unsigned char	Volume unit identifier for converted value in <i>pVolume</i>

Return Parameters

<i>pVolume</i>	double*	Stores converted value. The result is a volume value in volume unit specified by <i>VolUnit</i> identifier.
----------------	---------	---

6.7.3 Convert Flow into Device Unit Value

Function

```
long CCS_ConvFlowIntoDevUnit( HANDLE      hDevice,
                              unsigned char DosingUnit,
                              double       Flow,
                              unsigned char FlowUnit,
                              long *      pDevValue
                              )
```

Description

This function converts flow value into a device specific format (increments). The conversion depends on syringe configuration parameters, which belong to a certain dosing unit. Therefore a dosing unit number is required for proper conversion.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit
Flow	double	The flow value in flow unit format
FlowUnit	unsigned char	Flow unit identifier for value in Flow parameter

Return Parameters

pDevValue	long*	Stores converted value. The result is a value in a device specific format (increments)
-----------	-------	--

6.7.4 Convert Device Unit Value into Flow

Function

```
long CCS_ConvDevUnitIntoFlow( HANDLE      hDevice,
                              unsigned char DosingUnit,
                              long         DevValue,
                              unsigned char FlowUnit,
                              long *      pFlow
                              )
```

Description

This function converts a value from internal device unit format (increments) into flow. The conversion depends on syringe configuration parameters, which belong to a certain dosing unit. Therefore a dosing unit number is required for proper conversion.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit
DevValue	double	The value in device specific format (increments)
FlowUnit	unsigned char	Flow unit identifier for converted value in pFlow

Return Parameters

pFlow	double*	Stores converted value. The result is a flow value in flow unit specified by FlowUnit identifier.
-------	---------	---

6.8 Flow and Volume Units

6.8.1 Flow Unit Identifier

Use the following symbolic names when flow rate identifiers are required:

Identifier	Description
NCS_FLOW_UNIT_NL_S	Nanolitres per second
NCS_FLOW_UNIT_UL_S	Microlitres per second
NCS_FLOW_UNIT_UL_MIN	Microlitres per minute
NCS_FLOW_UNIT_UL_H	Microlitres per hour
NCS_FLOW_UNIT_ML_MIN	Millilitres per minute
NCS_FLOW_UNIT_ML_H	Millilitres per hour

6.8.2 Volume Unit Identifier

Use the following symbolic names when volume identifiers are required:

Identifier	Description
NCS_VOL_UNIT_NL	Nanolitres
NCS_VOL_UNIT_UL	Microlitres
NCS_VOL_UNIT_ML	Millilitres

7 Low level API

7.1 Introduction

The low level API is only a wrapper for the serial communication protocol of ceDOSYS device. The API offers a function for each serial command. A complete description of the serial protocol is part of the ceDOSYS User Manual.

8 DLL Integration into Borland

BDS2006 C++

You need the following files to include the library to the programming environment of 'Borland C++ Builder'.

cedosys_api.h	Constant definitions and declarations of the library functions
cedosys_api.dll	Dynamic link library
cedosys_api.lib	Import library (OMF Format)

To include the listed files you have to do the following steps:

First Step The files have to be copied to a working directory of the project.

Second Step Write the instruction **#include "cedosys_api.h"** to your program to include the constant definitions and the declarations of the library functions.

Third Step The file 'cedosys_api.lib' has to be added to the project. For that purpose you have to open the menu point 'Add to project...' of the menu 'Project'. Add the file 'cedosys_api.lib'. The figure below shows the German version of this dialog.

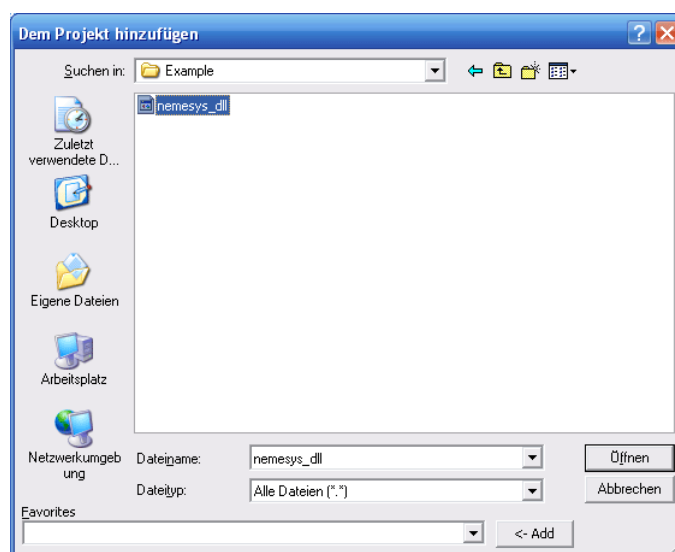


Figure 1 - Adding Library in Borland C++ Builder

After this three steps you can execute directly all library functions in your own code.