

NEMESYS

Januar 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
4.3 Parameter Storage	8
5 NEMESYS Command Set	9
5.1 Groups of functions	9
5.2 Initialisation	10
5.2.1 Open Device	10
5.2.2 Close Device	10
5.3 Configuration	12
5.3.1 Get Syringe Parameters	12
5.3.2 Set Syringe Parameters	13
5.3.3 Get Flow Rate Maximum	13
5.3.4 Get Syringe Level Maximum	14
5.3.5 Get Number Of Dosing Units	15
5.3.6 Calibrate	15
5.4 Device state	16
5.4.1 Is In Fault State	16
5.4.2 Is Operational	17
5.4.3 Clear Fault	17
5.4.4 Set Operational.....	18
5.5 Volume and Flow Units.....	19
5.5.1 Get Number Of Flow Units	19
5.5.2 Get Number Of Volume Units.....	20
5.5.3 Get Flow Unit String	20
5.5.4 Get Volume Unit String.....	21
5.5.5 Get Active Flow Unit.....	22
5.5.6 Get Active Volume Unit	22
5.5.7 Set Active Flow Unit	23
5.5.8 Set Active Volume Unit.....	23
5.5.9 Flow Unit Identifier.....	24
5.5.10 Volume Unit Identifier	24

5.6	Dosing	25
5.6.1	Dose Volume.....	25
5.6.2	Set Syringe Level.....	26
5.6.3	Generate Flow.....	27
5.6.4	Empty Syringe.....	27
5.6.5	Refill Syringe.....	28
5.6.6	Stop.....	29
5.6.7	Stop All Units.....	29
5.6.8	Emergency Stop All Units	30
5.7	Gradient Control.....	31
5.7.1	Get Running Gradient State.....	31
5.7.2	Set Gradient Values.....	32
5.7.3	Get Number Of Gradient Values.....	32
5.7.4	Get Gradient Value	33
5.7.5	Start Gradient.....	33
5.8	Dosing Info	35
5.8.1	Get Syringe Level Is.....	35
5.8.2	Get Flow Rate Is	35
5.8.3	Is Calibration Finished	36
5.8.4	Is Dosing Finished	37
5.9	Valve Control.....	38
5.9.1	Switch Valve.....	38
5.9.2	Is Valve Switched To Output.....	39
5.9.3	Set Valve Automatic.....	39
5.9.4	Is Valve Automatic On	40
5.9.5	Is Valve Installed.....	40
5.10	Error Handling	42
5.10.1	Get Error String.....	42
5.10.2	Get Last Device Error	42
5.10.3	Get Device Error String.....	43
6	DLL Integration into Borland BDS2006 C++	44
7	DLL Integration into Borland BDS2006 Delphi.....	45
8	DLL Integration into LabVIEW.....	46
8.1	Introduction	46
8.2	Integration into LabVIEW	46
8.3	Groups of functions.....	49
8.4	LabVIEW Example	49

2 Version History

Date	Version	Documentation	Description
01.08.2006	Ver 1.20	August 2006	<ul style="list-style-type: none">• First Library Version
25.09.2006	Ver 1.25	September 2006	<ul style="list-style-type: none">• DLL integration into National Instruments LabVIEW• A serious memory management bug fixed – borIndmm.dll is now required for proper function of nemesys_dll.dll
10.01.2007	Ver 1.25	Januar 2007	<ul style="list-style-type: none">• DLL integration into Borland Delphi

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 neMESYS 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 'nemesys_dll.dll' is an implementation of the neMESYS command set for the communication between neMESYS 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 neMESYS 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 platform.

The library 'nemesys_dll.dll' offers the whole set of neMESYS commands. Executing neMESYS commands, managing configuration parameters, executing dosing tasks and handling low level communication with the neMESYS control units 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 6). For several high-level programming languages are further documentations and its own examples available (Chapter 6).

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

nemesys_dll.dll and borIndmm.dll
--

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

To open the library 'nemesys_dl.dll' you have to use the function [NCS_OpenDevice\(\)](#). You have to do this before you can execute any neMESYS command. At the end of your

program you have to call [NCS_CloseDevice\(\)](#). For more detailed information about the initialisation procedure, have a look at the chapter 'Initialisation' (Chapter 5.2).

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.

4.3 Parameter Storage

The neMESYS library need to store several device parameters of connected neMESYS dosing units persistent. The library will store these parameters into a `devices.ini` file. The library will create this ini file inside the directory where your executable application file (`*.exe`) is located. You have to ensure that a user that works with your application has the right to create files in this directory or the library will fail to create the file and to store parameters persistent.

5 NEMESYS Command Set

5.1 Groups of functions

The neMESYS Command Set defines following groups:

[Initialisation](#)

[Configuration](#)

[Device state](#)

[Volume and flow units](#)

[Dosing](#)

[Gradient control](#)

[Dosing info](#)

[Valve control](#)

[Error handling](#)

5.2 Initialisation

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

[Open Device](#)

[Close Device](#)

5.2.1 Open Device

Function

HANDLE **NCS_OpenDevice**(BOOL bShowStatusWnd, long *pErrCode);

Description

Function "NCS_OpenDevice" opens the connection for sending and receiving commands, initializes the device and scans the device bus for connected dosing units.

Parameters

bShowStatusWindow	BOOL	True: Display a status window during initialisation and dosing unit search.
-------------------	------	---

Return Parameters

pErrCode	long*	Returns error code
Return value	HANDLE	HANDLE for device access. Nonzero if successful; otherwise 0

Related Functions

[Close Device](#)

5.2.2 Close Device

Function

long **NCS_CloseDevice**(HANDLE hDevice);

Description

Function "NCS_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](#)

5.3 Configuration

This group defines all required functions for reading and writing configuration parameters of dosing platform or single dosing units:

[Get Syringe Parameters](#)

[Set Syringe Parameters](#)

[Get Flow Rate Maximum](#)

[Get Syringe Level Maximum](#)

[Get Number Of Dosing Units](#)

[Calibrate](#)

5.3.1 Get Syringe Parameters

Function

long **NCS_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	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to query (0 ... NumberOfDosingUnits – 1)

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 succesfull, otherwise < 0

Related Functions

[Set Syringe Parameters](#)

5.3.2 Set Syringe Parameters

Function

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

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 succesfull, otherwise < 0
--------------	------	--

Related Functions

[Set Syringe Parameters](#)

5.3.3 Get Flow Rate Maximum

Function

long **NCS_GetFlowRateMax**(HANDLE hDevice, unsigned char DosingUnit, double *pFlowRateMax)

Description

Reads the maximum realizable flow rate. 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

Return Parameters

pFlowRateMax	double*	Maximum realizable flow rate. (value depends on syringe parameters and configured flow rate unit)
Return value	long	Error code. 0 if succesfull, otherwise < 0

Related Functions

[Get Syringe Parameters](#)

[Get Active Flow Unit](#)

5.3.4 Get Syringe Level Maximum

Function

long **NCS_GetSyringeLevelMax**(HANDLE hDevice, unsigned char DosingUnit, double *pSyringeLevelMax)

Description

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

Return Parameters

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

Related Functions

[Get Syringe Parameters](#)

[Get Active Volume Unit](#)

5.3.5 Get Number Of Dosing Units

Function

long **NCS_GetNumberOfDosingUnits**(HANDLE hDevice)

Description

Returns the number of connected and properly configured dosing units.

Parameters

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

Return Parameters

Return value	long	Returns the number of detected dosing units (≥ 0) or an error code (< 0).
--------------	------	--

5.3.6 Calibrate

Function

long **NCS_Calibrate**(HANDLE hDevice, unsigned char DosingUnit)

Description

Executes reference move for single dosing unit. A reference move is necessary if a new dosing unit is connected to the dosing platform or if the real position of a dosing unit does not match the position returned by software.

Parameters

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

Return Parameters

Return value	long	Error code: 0 if succesfull, otherwise < 0
--------------	------	--

5.4 Device state

This group defines all required functions for reading and writing device state specific parameters:

[Is In Fault State](#)

[Is Operational](#)

[Clear Fault](#)

[Set Operational](#)

5.4.1 Is In Fault State

Function

long **NCS_IsInFaultState**(HANDLE hDevice, unsigned char DosingUnit)

Description

This function returns the device fault state. If dosing unit is in fault state then you can try to clear the fault state by calling [NCS_ClearFault\(\)](#).

Parameters

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

Return Parameters

Return value	long	0 – Dosing unit is not in fault state 1 – Dosing unit is in fault state < 0 – Error code
--------------	------	--

Related Functions

[Clear Fault](#)

5.4.2 Is Operational

Function

long **NCS_IsOperational**(HANDLE hDevice, unsigned char DosingUnit)

Description

Returns the device operational state. If a dosing unit is not in fault state but it is not operational then you should call the function [NCS_SetOperational\(\)](#) to set the unit into an operational state. You can not set device into operational state if it is in a fault state – you need to clear the error first.

Parameters

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

Return Parameters

Return value	long	0 – Dosing unit is not in operational state 1 – Dosing unit is in operational state < 0 – Error code
--------------	------	--

Related Functions

[Set Operational](#)

5.4.3 Clear Fault

Function

long **NCS_ClearFault**(HANDLE hDevice, unsigned char DosingUnit)

Description

Clears fault state of single dosing unit. If a device error occurred you need to call this function to set dosing unit into an error free state. If a serious error occurred then the device may be still in fault state after a call to this function.

Parameters

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

Return Parameters

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

Related Functions

[Is In Fault State](#)

5.4.4 Set Operational

Function

long **NCS_SetOperational**(HANDLE hDevice, unsigned char DosingUnit)

Description

Sets dosing unit into an operational state. Dosing is possible only in operational state. If a device error occurred you first need to call [NCS_ClearFault\(\)](#) and then you need to call this function to set a single dosing unit into an operational state.

Parameters

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

Return Parameters

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

Related Functions

[Is Operational](#)

5.5 Volume and Flow Units

This group defines a set of functions for configuration of flow rate units and volume units:

[Get Number Of Flow Units](#)

[Get Number Of Volume Units](#)

[Get Flow Unit String](#)

[Get Volume Unit String](#)

[Get Active Flow Unit](#)

[Get Active Volume Unit](#)

[Set Active Flow Unit](#)

[Set Active Volume Unit](#)

5.5.1 Get Number Of Flow Units

Function

long **NCS_GetNumberOfFlowUnits** (HANDLE hDevice)

Description

Returns the number of supported flow rate units.

Parameters

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

Return Parameters

Return value	long	Returns the number of supported flow units (> 0) or an error code (<= 0).
--------------	------	---

Related Functions

[Get Flow Unit String](#)

[Get Active Flow Unit](#)

5.5.2 Get Number Of Volume Units

Function

long **NCS_GetNumberOfVolumeUnits** (HANDLE hDevice)

Description

Returns the number of supported volume units.

Parameters

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

Return Parameters

Return value	long	Returns the number of supported volume units (> 0) or an error code (<= 0).
--------------	------	---

Related Functions

[Get Volume Unit String](#)

[Get Active Volume Unit](#)

5.5.3 Get Flow Unit String

Function

long **NCS_GetFlowUnitString**(HANDLE hDevice, unsigned char FlowUnitId, char *pUnitStringShort, unsigned char MaxStringShortSize, char *pUnitStringLong, unsigned char MaxStringLongSize);

Description

Returns a short (i.e. nl/s) and a long version (i.e. Nanolitres/Second) of a flow unit string for a certain flow unit identifier.

Parameters

hDevice	HANDLE	Handle for device access
FlowUnitId	unsigned char	Identifier of flow unit (0 ... NumberOfFlowUnits – 1) Use these symbolic names for flow rate units.
MaxStringShort	unsigned char	Size of buffer pUnitStringShort
MaxStringLong	unsigned char	Size of buffer pUnitStringLong

Return Parameters

pUnitStringShort	char*	Short unit string (nl/s, µl/s, ml/min...)
pUnitStringLong	char*	Long unit string (Nanolitres/Second...)
Return value	long	Error code. 0 if succesfull, otherwise < 0

Related Functions

[Get Number Of Flow Units](#)

[Flow Unit Identifier](#)

5.5.4 Get Volume Unit String

Function

long **NCS_GetVolumeUnitString**(HANDLE hDevice, unsigned char VolumeUnitId, char *pUnitStringShort, unsigned char MaxStringShortSize, char *pUnitStringLong, unsigned char MaxStringLongSize);

Description

Returns a short (i.e. µl) and a long version (i.e. Microlitres) of a volume unit string for a certain volume unit identifier.

Parameters

hDevice	HANDLE	Handle for device access
VolumeUnitId	unsigned char	Identifier of vol. unit (0 ... NumberOfVolumeUnits – 1) Use these symbolic names for volume units.
MaxStringShort	unsigned char	Size of buffer pUnitStringShort
MaxStringLong	unsigned char	Size of buffer pUnitStringLong

Return Parameters

pUnitStringShort	char*	Short unit string (nl, µl, ml ...)
pUnitStringLong	char*	Long unit string (Nanalitres, Microlitres ...)
Return value	long	Error code. 0 if succesfull, otherwise < 0

Related Functions

[Get Number Of Volume Units](#)

[Volume Identifier](#)

5.5.5 Get Active Flow Unit

Function

long **NCS_GetActiveFlowUnit**(HANDLE hDevice, unsigned char DosingUnit)

Description

Returns [identifier](#) of active flow unit. All functions that use flow rate values (i.e. [NCS_DoseVolume\(\)](#)) expect and return these values in active flow units.

Parameters

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

Return Parameters

Return value	long	Returns the active flow unit (≥ 0) or an error code (< 0).
--------------	------	---

Related Functions

[Get Flow Unit String](#)

[Set Active Flow Unit](#)

5.5.6 Get Active Volume Unit

Function

long **NCS_GetActiveVolumeUnit**(HANDLE hDevice, unsigned char DosingUnit)

Description

Returns [identifier](#) of active volume unit. All functions that use volume values (i.e. [NCS_DoseVolume\(\)](#)) expect and return these values in volume units.

Parameters

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

Return Parameters

Return value	long	Returns the active volume unit (≥ 0) or an error code (< 0).
--------------	------	---

Related Functions

[Get Volume Unit String](#)

[Set Active Volume Unit](#)

5.5.7 Set Active Flow Unit

Function

long **NCS_SetActiveFlowUnit**(HANDLE hDevice, unsigned char DosingUnit, unsigned char FlowUnit);

Description

Set active flow unit for single device. All functions that use flow rate values (i.e. [NCS_DoseVolume\(\)](#)) expect and return these values in flow rate units.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to change
FlowUnit	unsigned char	Identifier of flow unit (0 .. NumberOfFlowUnits – 1) Use these symbolic names for flow rate units.

Return Parameters

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

Related Functions

[Get Active Flow Unit](#)

[Flow Unit Identifiers](#)

5.5.8 Set Active Volume Unit

Function

long **NCS_SetActiveVolumeUnit**(HANDLE hDevice, unsigned char DosingUnit, unsigned char VolumeUnit);

Description

Set active volume unit for single device. All functions that use volume values (i.e. [NCS_DoseVolume\(\)](#)) expect and return these values involume units.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to change
VolumeUnit	unsigned char	Identifier of volume unit (0 .. NumberOfVolumeUnits – 1) Use these symbolic names for volume units.

Return Parameters

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

Related Functions

[Get Active Volume Unit](#)

[Volume Unit Identifiers](#)

5.5.9 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
NCS_FLOW_UNIT_MM_S	Millimetres per second (this unit is indepentend from syringe configuration)

5.5.10 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
NCS_VOL_UNIT_MM	Millimetres (this units is independent from syringe configuration)

5.6 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](#)
- [Empty Syringe](#)
- [Refill Syringe](#)
- [Stop](#)
- [Stop All Units](#)
- [Emergency Stop All Units](#)

5.6.1 Dose Volume

Function

```
long NCS_DoseVolume(HANDLE hDevice, unsigned char DosingUnit, double *pVolume, double *pFlowRate);
```

Description

Doses a defined volume with a defined flow rate. If flow rate exceeds the [maximum flow rate](#) value then dosing is performed with maximum flow rate.

Parameters

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

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

[Get Active Volume Unit](#)

[Get Active Flow Unit](#)

[Stop](#)

5.6.2 Set Syringe Level

Function

long **NCS_SetSyringeLevel**(HANDLE hDevice, unsigned char DosingUnit, double *pVolume, double *pFlowRate)

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 fill level in active volume unit format.
pFlowRate	double*	Continuous flow rate to generate in active flow rate unit format. Only positive flow rate values are valid here. The fill level defines the direction of movement.

Return Parameters

pVolume	double*	Realizable volume value.
pFlowRate	double*	Realizable flow rate.
Return value	long	Error code; 0 if succesfull, otherwise < 0

Related Functions

[Get Active Volume Unit](#)

[Get Active Flow Unit](#)

[Stop](#)

5.6.3 Generate Flow

Function

long **NCS_GenerateFlow**(HANDLE hDevice, unsigned char DosingUnit, double *pFlowRate)

Description

Generates a continuous flow with a certain flow rate. Moves the pusher until it reaches its limits or until application calls [NCS_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 active flow rate unit format. A positive flow rate indicates delivery of reagent and a negative value indicates a take up of reagent.

Return Parameters

pFlowRate	double*	Realizable flow rate.
Return value	long	Error code; 0 if succesfull, otherwise < 0

Related Functions

[Get Active Flow Unit](#)

[Stop](#)

5.6.4 Empty Syringe

Function

long **NCS_EmptySyringe**(HANDLE hDevice, unsigned char DosingUnit, double *pFlowRate)

Description

Eympties syringe with a defined flow rate by moving pusher to lower limit of dosing unit.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to empty.
pFlowRate	double*	Flow rate in active flow rate unit format. Only positive flow rate values are possible here.

Return Parameters

pFlowRate	double*	Realizable flow rate.
Return value	long	Error code; 0 if succesfull, otherwise < 0

Related Functions

[Stop](#)

[Refill Syringe](#)

5.6.5 Refill Syringe

Function

long **NCS_RefillSyringe**(HANDLE hDevice, unsigned char DosingUnit, double *pFlowRate)

Description

Refills syringe with a certain flow rate by moving pusher to upper limit of dosing unit.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to refill
pFlowRate	double*	Flow rate in active flow rate unit format. Only positive flow rate values are possible here.

Return Parameters

pFlowRate	double*	Realizable flow rate.
Return value	long	Error code; 0 if succesfull, otherwise < 0

Related Functions

[Stop](#)

[Empty Syringe](#)

5.6.6 Stop

Function

long **NCS_Stop**(HANDLE hDevice, unsigned char DosingUnit)

Description

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 succesfull, otherwise < 0
--------------	------	--

Related Functions

[Stop All Dosing Units](#)

[Emergency Stop All Dosing Units](#)

5.6.7 Stop All Units

Function

long **NCS_StopAllUnits**(HANDLE hDevice)

Description

Stops dosing of all dosing units.

Parameters

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

Return Parameters

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

5.6.8 Emergency Stop All Units

Function

long **NCS_EmergencyStopAllUnits**(HANDLE hDevice)

Description

Performs quick stop operation for all dosing units.

Parameters

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

Return Parameters

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

5.7 Gradient Control

This group defines all required functions for configuration and execution of gradient value lists:

[Get Running Gradient State](#)

[Set Gradient Values](#)

[Get Number Of Gradient Values](#)

[Get Gradient Value](#)

[Start Gradient](#)

5.7.1 Get Running Gradient State

Function

long **NCS_GetRunningGradientState**(HANDLE hDevice, unsigned char DosingUnit, unsigned long *pRunningCycle, unsigned long *pActiveValueIndex)

Description

Returns state of gradient list that is executing currently.

Parameters

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

Return Parameters

pRunningCycle	unsigned long	Returns the number of remaining gradient cycles. This value starts at the configured number of gradient cycles and counts down to zero.
pActiveValueIndex	unsigned long	Returns the actual index into the gradient value list of the flow rate that is currently active.
Return value	long	Error code; 0 if successful, otherwise < 0

5.7.2 Set Gradient Values

Function

long **NCS_SetGradientValues**(HANDLE hDevice, unsigned char DosingUnit, double *pGradientValueBuf, unsigned long NumberOfValuesInBuf)

Description

Sets up the list of gradient values. This function creates an internal copy of the values in pGradientValueBuf.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to setup
pGradientValueBuf	double*	Array of flow rate values to store in gradient list (flow rate values are in active flow rate unit format)
NumberOfValuesInBuf	unsigned long	Number of values in array pGradientValueBuf

Return Parameters

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

Related Functions

[Start Gradient](#)

5.7.3 Get Number Of Gradient Values

Function

long **NCS_GetNumberOfGradientValues**(HANDLE hDevice, unsigned char DosingUnit, unsigned long *pNumberOfGradientValues)

Description

Returns number of flow rate values in dosing unit gradient list.

Parameters

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

Return Parameters

pNumberOfGradientValues	unsigned long*	Number of values in gradient value list
Return value	long	Error code; 0 if succesfull, otherwise < 0

Related Functions

[Set Gradient Values](#)

5.7.4 Get Gradient Value

Function

long **NCS_GetGradientValue**(HANDLE hDevice, unsigned char DosingUnit, unsigned long Index, double *pValue)

Description

Reads a certain value from the list of gradient values.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to query
Index	unsigned long	Index into gradient value list

Return Parameters

pValue	double*	Flow rate in active flow rate unit format at index position
Return value	long	Error code; 0 if succesfull, otherwise < 0

5.7.5 Start Gradient

Function

long **NCS_StartGradient**(HANDLE hDevice, unsigned char DosingUnit, unsigned long Cycles, unsigned long TimeGrid100Ms)

Description

Starts processing of gradient value list for single dosing unit. To stop gradient control call [NCS_Stop\(\)](#) or [NCS_StopAllUnits\(\)](#).

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to control
Cycles	unsigned long	Number of cycles to process the gradient value list. A zero value indicates infinite processing of the list until application stops operation.
TimeGrid100Ms	unsigned long	Defines the time grid in 100 ms scale for processing the list of flow rate values (1 = 100 ms, 2 = 200 ms ...)

Return Parameters

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

Related Functions

[Set Gradient Values](#)

[Stop](#)

[Stop All Units](#)

5.8 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 Calibration Finished](#)

[Is Dosing Finished](#)

5.8.1 Get Syringe Level Is

Function

long **NCS_GetSyringeLevelIs**(HANDLE hDevice, unsigned char DosingUnit, double *pVolume)

Description

Returns actual syringe fill level.

Parameters

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

Return Parameters

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

5.8.2 Get Flow Rate Is

Function

long **NCS_GetFlowRateIs**(HANDLE hDevice, unsigned char DosingUnit, double *pFlowRateIs)

Description

Returns actual flow rate of single dosing unit.

Parameters

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

Return Parameters

pFlowRatels	double*	Actual flow rate in active flow rate unit format.
Return value	long	Error code; 0 if succesfull, otherwise < 0

5.8.3 Is Calibration Finished

Function

long **NCS_IsCalibrationFinished**(HANDLE hDevice, unsigned char DosingUnit)

Description

Returns state of calibration move. If a calibration move is started with [NCS_Calibrate\(\)](#) then this function returns 0. (1 if calibration move is finished)

Parameters

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

Return Parameters

Return value	long	0 – calibration move is active 1 – calibration move is finished and device stopped < 0 – Error code
---------------------	------	---

Related Functions

[Calibrate](#)

5.8.4 Is Dosing Finished

Function

long **NCS_IsDosingFinished**(HANDLE hDevice, unsigned char DosingUnit)

Description

Returns 1 if dosing is finished. Dosing is finished if the device is stopped.

Parameters

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

5.9 Valve Control

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

[Switch Valve](#)

[Is Valve Switched To Output](#)

[Set Valve Automatic](#)

[Is Valve Automatic On](#)

[Is Valve Installed](#)

5.9.1 Switch Valve

Function

long **NCS_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 (i.e. if [NCS_RefillSyringe\(\)](#) is called) and switching valve to output is required for delivering reagent (i.e. if [NCS_EmptySyringe\(\)](#) is called),

Parameters

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

Return Parameters

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

Related Functions

[Is Valve Switched To Output](#)

5.9.2 Is Valve Switched To Output

Function

long **NCS_IsValveSwitchedToOutput**(HANDLE hDevice, unsigned char DosingUnit)

Description

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](#)

5.9.3 Set Valve Automatic

Function

long **NCS_SetValveAutomatic**(HANDLE hDevice, unsigned char DosingUnit, unsigned char ValveAutomaticOn)

Description

Enable /disable automatic valve switching. If automatic valve switching is enabled the device switches the valve in the right direction as soon as the direction of the moving pusher changes.

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to change
ValveAutomaticOn	unsigned char	0 – Switch valve automatic off 1 – Switch valve automatic on

Return Parameters

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

Related Functions

[Is Valve Automatic On](#)

5.9.4 Is Valve Automatic On

Function

long **NCS_IsValveAutomaticOn**(HANDLE hDevice, unsigned char DosingUnit)

Description

Returns actual state of valve automatic.

Parameters

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

Return Parameters

Return value	long	0 – Valve automatic is switched off 1 – Valve automatic is switched on < 0 - Error code
--------------	------	---

Related Functions

[Set Valve Automatic](#)

5.9.5 Is Valve Installed

Function

long **NCS_IsValveInstalled**(HANDLE hDevice, unsigned char DosingUnit)

Description

Returns 1 if device is equipped with valve. (0 if no valve is installed)

Parameters

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

Return Parameters

Return value	long	0 – Dosing unit is not equipped with valve 1 – Dosing unit is quipped with valve < 0 - Error code
---------------------	------	---

5.10 Error Handling

This group defines all required functions error handling:

[Get Error String](#)

[Get Last Device Error](#)

[Get Device Error String](#)

5.10.1 Get Error String

Function

```
void NCS_GetErrorString(long ErrCode, char *pErrorString, unsigned char MaxErrStringSize)
```

Description

Returns an descriptive error message for a certain application error code.

Parameters

ErrCode	long	Handle for device access
MaxErrStringSize	unsigned char	Size of string buffer pErrorString (the maximum size is limited to 255)

Return Parameters

pErrString	char*	Zero terminated error message string
------------	-------	--------------------------------------

5.10.2 Get Last Device Error

Function

```
long NCS_GetLastDevErr(HANDLE hDevice, unsigned char DosingUnit, long *pLastErr)
```

Description

If a certain dosing unit indicates a device error then this function returns error code of the last error that occurred for a certain device.

Parameters

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

Return Parameters

pLastErr	long*	Last device error that occurred
Return value	long	Error code; 0 if succesfull, otherwise < 0

Related Functions

[Is In Fault State](#)

[Get Device Error String](#)

5.10.3 Get Device Error String

Function

long **NCS_GetDevErrorString**(HANDLE hDevice, unsigned char DosingUnit, long DevErrCode, char *pErrString, unsigned char MaxErrStringSize)

Description

Returns a descriptive error message for a certain device error code obtained with call to [NCS_GetLastDevErr\(\)](#).

Parameters

hDevice	HANDLE	Handle for device access
DosingUnit	unsigned char	Number of dosing unit to query
DevErrCode	long	Device error code
MaxErrStringSize	unsigned char	Size of character string buffer pErrString

Return Parameters

pErrString	char*	Returns zero terminated device error message string
Return value	long	Error code; 0 if succesfull, otherwise < 0

Related Functions

[Is In Fault State](#)

[Get Last Device Error](#)

6 DLL Integration into Borland BDS2006 C++

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

nemesys_api.h	Constant definitions and declarations of the library functions
nemesys_dll.dll	Dynamic link library
nemesys_dll.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 "nemesys_api.h"** to your program to include the constant definitions and the declarations of the library functions.

Third Step The file 'nemesys_dll.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 'nemesys_dll.lib'. The figure below shows the German version of this dialog.

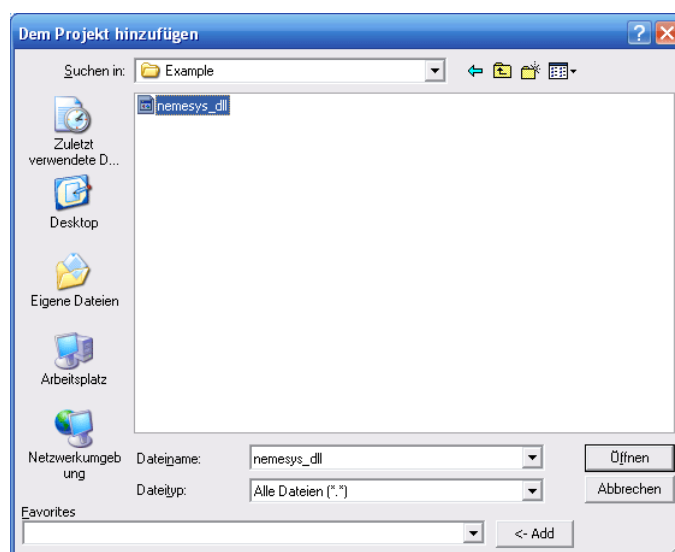


Figure 1 - Adding Library in Borland C++ Builder

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

7 DLL Integration into Borland BDS2006 Delphi

The following files are necessary to include the library functions to the programming environment of 'Borland Delphi'.

nemesys_api.pas Constant definitions and declarations of the library functions
nemesys_dll.dll Dynamic link library

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

First Step You have to copy all files to the working directory of the project.

Second Step Write the instruction '**uses nemesys_api**' to your program to include the constant definitions and the declarations of the library functions.

After this two steps you can execute all the neMESYS commands directly in your own program.

8 DLL Integration into LabVIEW

8.1 Introduction

This documentation "DLL Integration into LabVIEW" explains how the 'Windows 32-Bit DLL' is integrated into a program. This document should simplify the programming of the control software based on Windows.

It is a short overview, which important functions are used and like a program must be developed. A complete overview of all existing functions is available in the document "Windows 32-Bit DLL".

ceconi GmbH certifies that the content of this library is correct according his best knowledge.

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.

8.2 Integration into LabVIEW

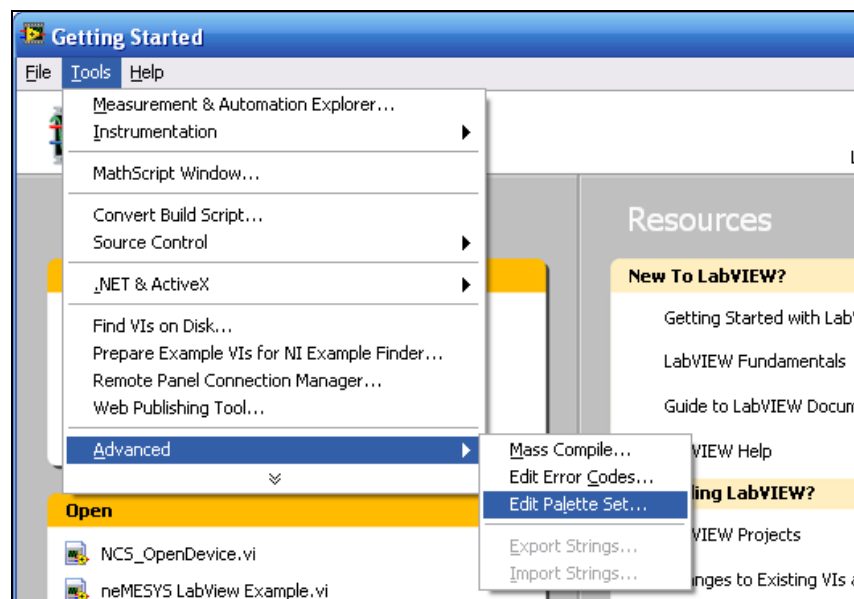
For an easy start with LabVIEW programming, all of the 'Windows 32-Bit DLL' function blocks are already configured for you (with LabVIEW 8.0). For each neMESYS command we have a VI block. In order to use the different VIs you have to perform the following steps:

First Step Copy the directory 'errors' from your installation directory into the directory '*...National Instruments\LabVIEW X.x\user.lib*'

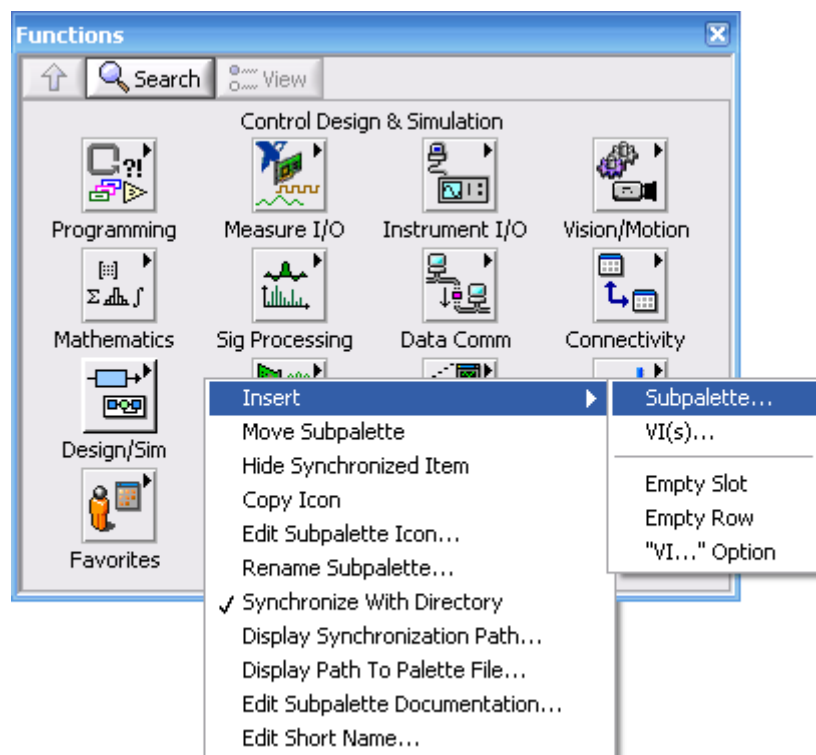
Second Step Start LabVIEW

Third Step The neMESYS VI's have to be added to the LabVIEW Palette Set. Select

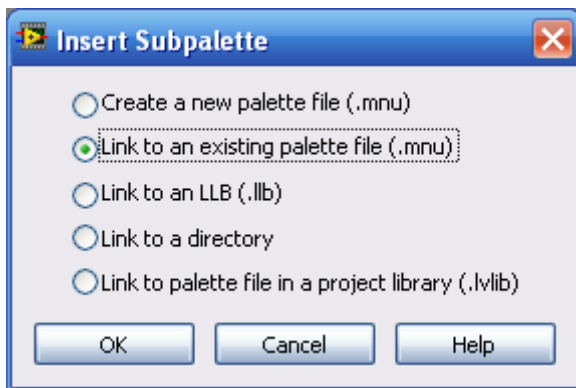
Advanced -> Edit Palette Set from the Tools menu



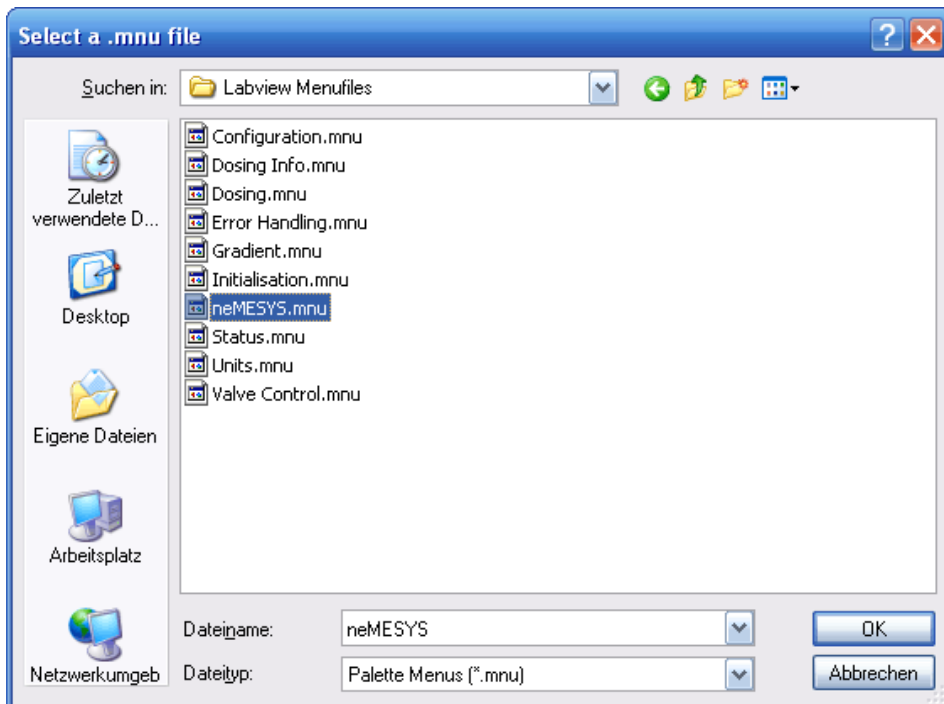
Then click with the right mouse button into the palette and select *Insert -> Subpalette* from the popup menus



Select *Link to an existing palette file (.mnu)* from the *Insert Subpalette* window and then click *OK*.



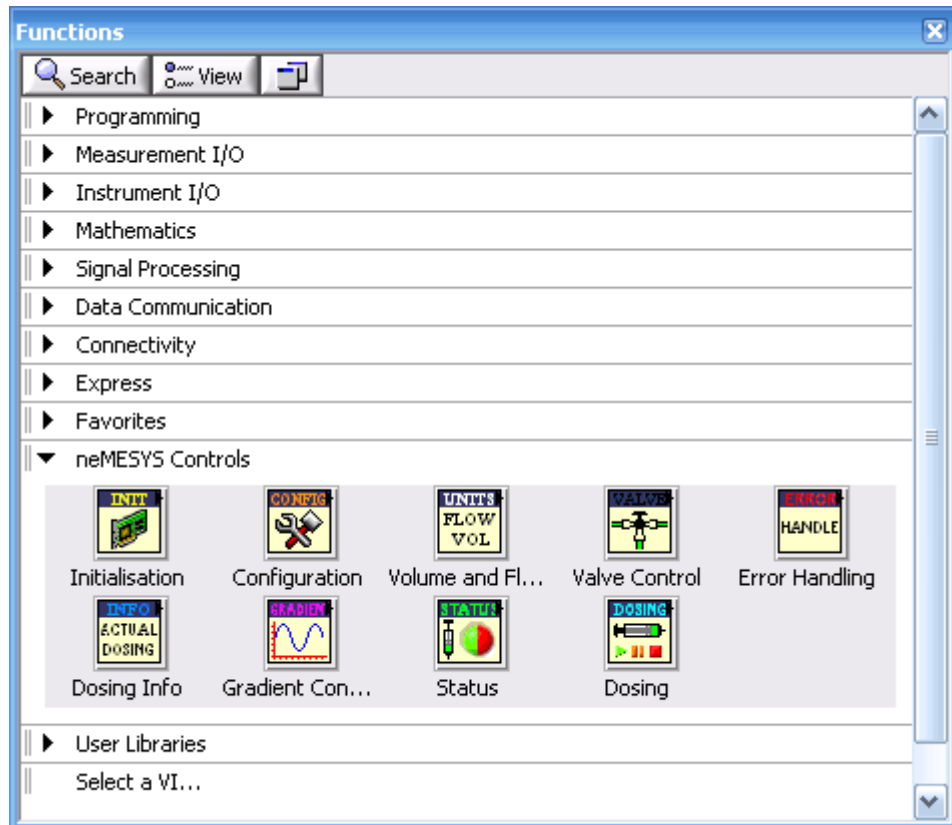
Select the file *neMESYS.mnu* from the *...neMESYS Library\Labview Menufiles* directory within your installation directory.



Finally save the changes you just made to the palette. Now the neMESYS VI's are accesible through the functions palette.

8.3 Groups of functions

The VI's of *neMESYS LabVIEW Integration* are grouped according to the [function groups](#) of '*neMESYS Windows 32-Bit DLL*'.



For detailed information about the function groups and the single VI's please refer to the [functions](#) description of the '*neMESYS Windows 32-Bit DLL*'.

8.4 LabVIEW Example

The example '*neMESYS Labview Example.vi*' in LabVIEW is a dialog based application. The application shows you, how the communication with the neMESYS devices has to be initialised and how several dosing tasks can be realized.

First opens a dialog and scans for connected neMESYS devices. Then the VI reads several configuration parameters from first connected device and sets up the user interface accordingly. Then the VI executes an calibration move and a complete refill and empty sequence for the first connected device. If this procedure is finished, a number of buttons are displayed for manual control of first dosing unit.