Metrohm
Ion analysis
CH-9101 Herisau/Switzerland
Phone +41 71 353 85 85
Fax +41 71 353 89 01
E-Mail info@metrohm.ch
Internet www.metrohm.ch

Dosing Interface USB Toolbox

Version 1.0

Reference manual

8.104.0003 06.2005 / dm

Table of content

1 Overview

The Liquid Handling library for Metrohm 846 Dosing Interface provides a Dynamic Link Library (DLL) for the software development for the Microsoft Windows® 2000 and XP operating systems. Based on the Metrohm Universal Serial Bus driver (metr_770.dll) the Liquid Handling Library offers all functions necessary for free programming of liquid handling processes with the versatile Dosino drives.

The following programming languages are supported:

- JAVA
- C++
- Delphi
- Visual Basic

For these languages programming interface definitions are available, see annex.

1.1 Distributed files

This distribution includes the following files:

DosIntFace846.dll	846 Dosing Interface driver
DosIntFace846.lib	Library
DosIntFace846.def	Definition file
metr_770.dll	Metrohm USB driver
metr_770.inf	Installation file
metr_770.sys	System file
DosIntFace846.java	Java interface class
DosIntFace846Exception.java	Java interface exception class
DosIntFace846.html	JavaDoc file
DosIntFace846Exception.html	JavaDoc file
DosIntFace846.h	C++ interface header file
DosIntFace846.pas	Delphi interface
DosIntFace846.bas	Visual Basic interface
Demo_846.exe	Executable demo program
8.104.0003.pdf	Reference manual

1.2 The 846 Dosing Interface

The 846 Dosing Interface is part of the Metrohm **Titrando system** as a system component and control instrument. The instruments are controlled via USB 1.1 connections.

The four MSB connectors (MSB=Metrohm Serial Bus) of the 846 Dosing Interface enable the operation of Metrohm Dosinos (models 700 and 800) which allow complex liquid handling processes.

One or more 846 Dosing Interfaces can be controlled at the same time. Each of them can operate up to four Metrohm 700/800 Dosinos.

1.2.1 System connections

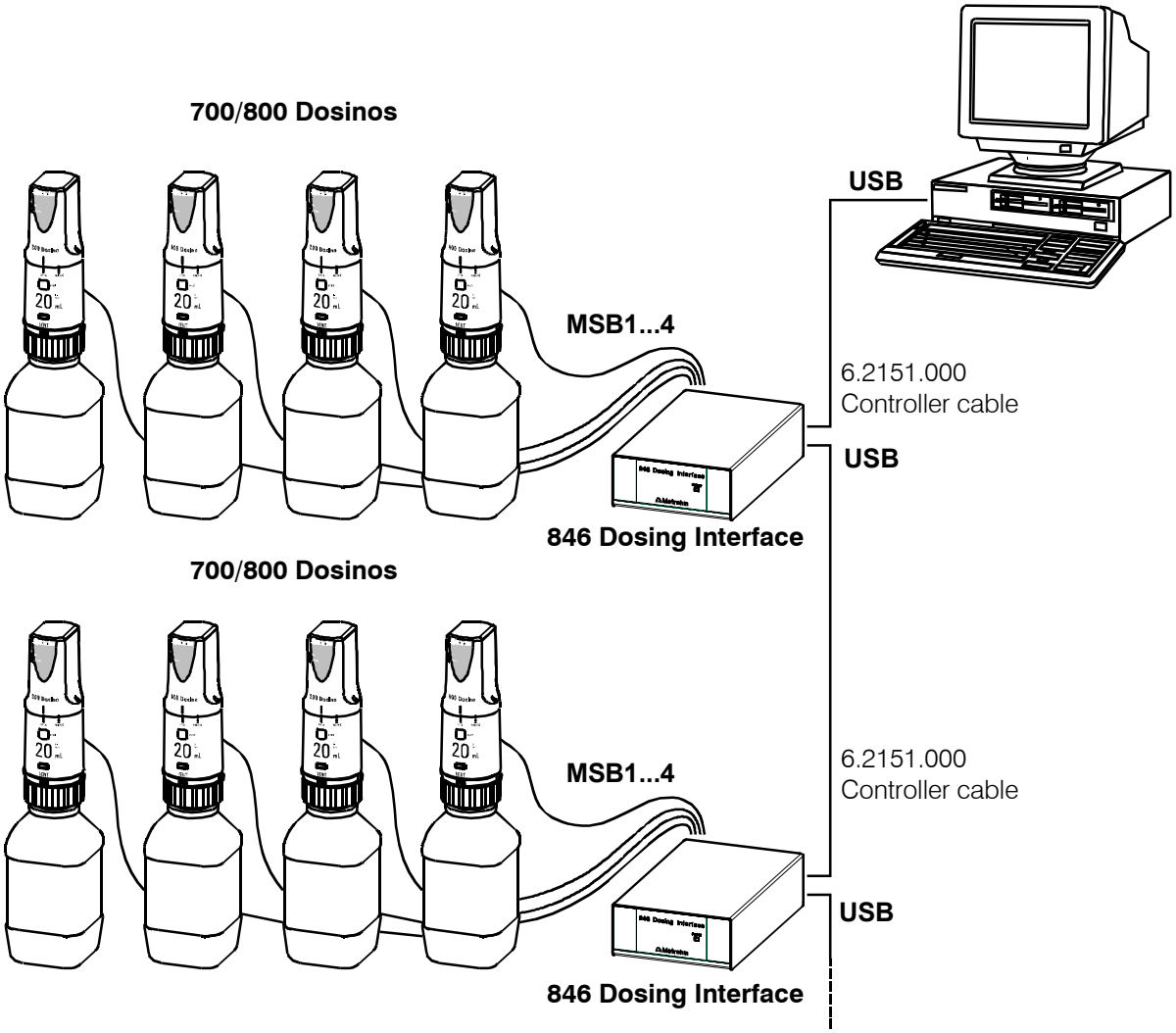


Fig. 1 System connections

Several 846 Dosing Interfaces can be connected via USB in a daisy chain. All of these instruments can be controlled by a PC software via USB. The Metrohm USB driver (metr_770.dll) controls the basic USB data transfer. The Dosing Interface USB Toolbox provides the low level functions for controlling the Dosino drives.

1.2.2 Connectors of the 846 Dosing Interface

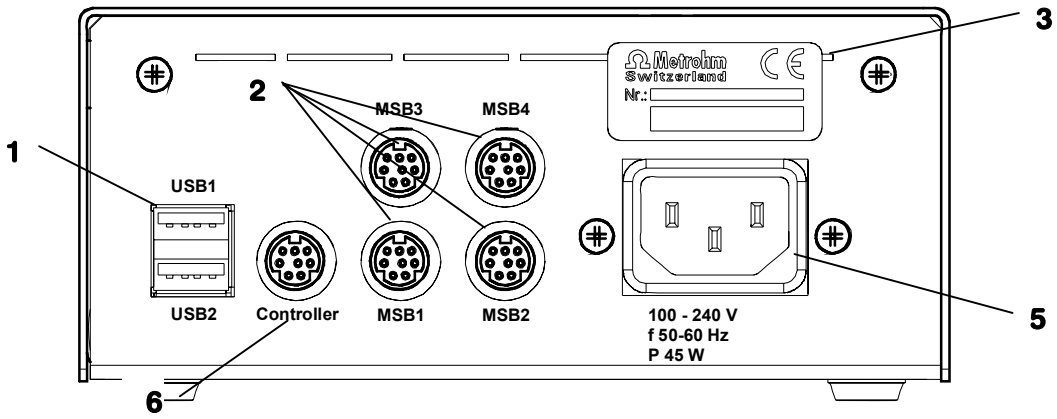


Fig. 2: Rear view of the 846 Dosing Interface

1 **USB connections USB 1 and USB 2**
USB ports (type A) for connection of peripheral devices or further Dosing Interfaces.

2 **MSB connections MSB 1 to MSB 4**
Connection for dosing drives.

3 **Instrument type and serial number**

4 **Controller connection**
Connection for PC or further Dosing interfaces. Use 6.2151.000 controller cable.

5 **Mains socket**
Mains connection

1.2.3 PC connection

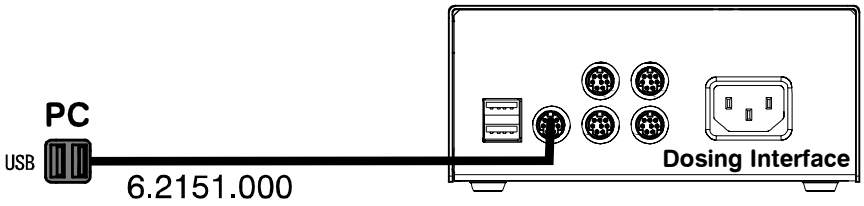


Fig. 3: Dosing Interface – Personal Computer

Always use the 6.2151.000 Controller cable to connect a 846 Dosing Interface to the USB socket of a PC or another 846 Dosing Interface.

You can extend the connection with a commercially available USB extension cable (type A/m – type A/f). The length of the connection should not exceed 5 m. If you require a longer connection then you need a USB signal amplifier. Up to five USB signal amplifiers can be connected in series; this allows a maximum extension of 25 m.

1.2.4 Dosino Connection

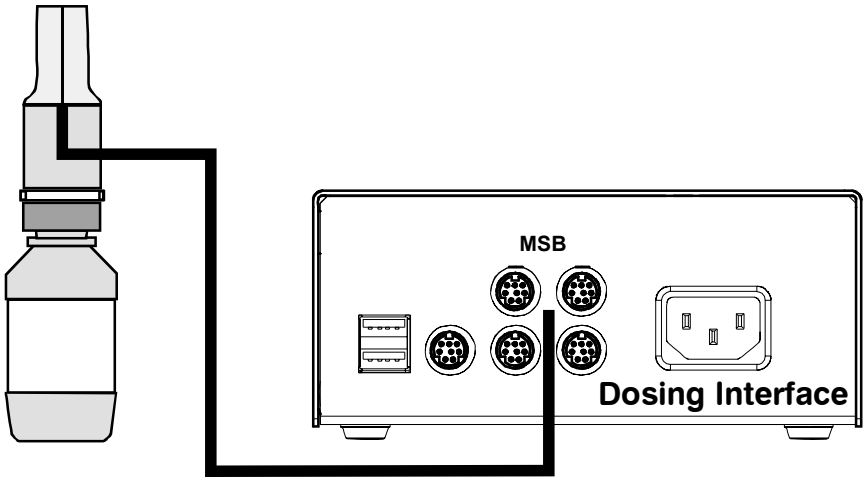
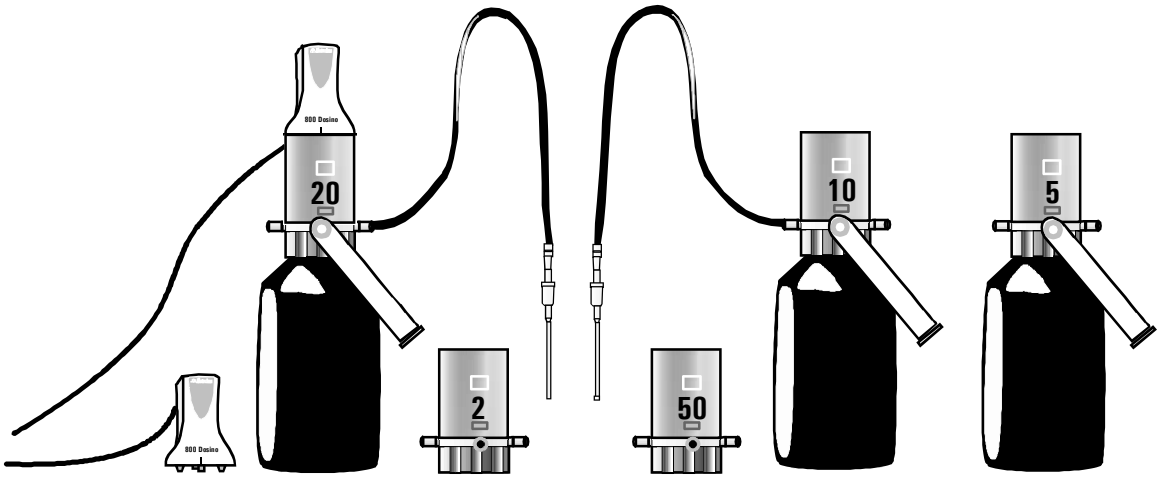


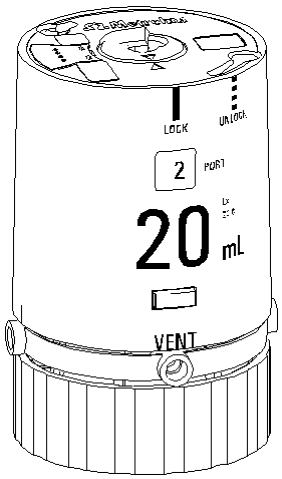
Fig. 4: Connecting a Dosino drive

700/800 Dosino drives are connected at the MSB connecters 1 to 4. A Dosino drive can be recognized and integrated in a system by use of the **`Init846()`** function. The MSB(…) connector specifies the Dosino number.

1.3 Metrohm 700/800 Dosinos and Dosing Units

The Metrohm 800 Dosino is a versatile dosing drive which can be used with various Metrohm dosing or titrating instruments (e.g. 808 or 809 Titrando). The 800 Dosino and its associated 807 Dosing unit are suitable for use as a buret, not just for simple dosing tasks and titrations, etc. but also for complex automation and liquid handling applications, such as sample transfer or pipetting.





The 800 Dosino together with an 807 Dosing unit (available with 2, 5, 10, 20 or 50 mL cylinder) forms a buret unit for simple dosing tasks or complex liquid handling applications.

The dosing units are normally mounted on reagent bottles and tubing is connected to the necessary dosing inlets and outlets. Four ports are available.

The 800 Dosino dosing drive can be simply attached to and removed from a dosing unit.

When the drive is attached the dosing piston in the dosing unit is coupled to drive rod **9** of the drive and cam **10** of the drive is guided into the recess provided for it in the centering tube of the dosing.

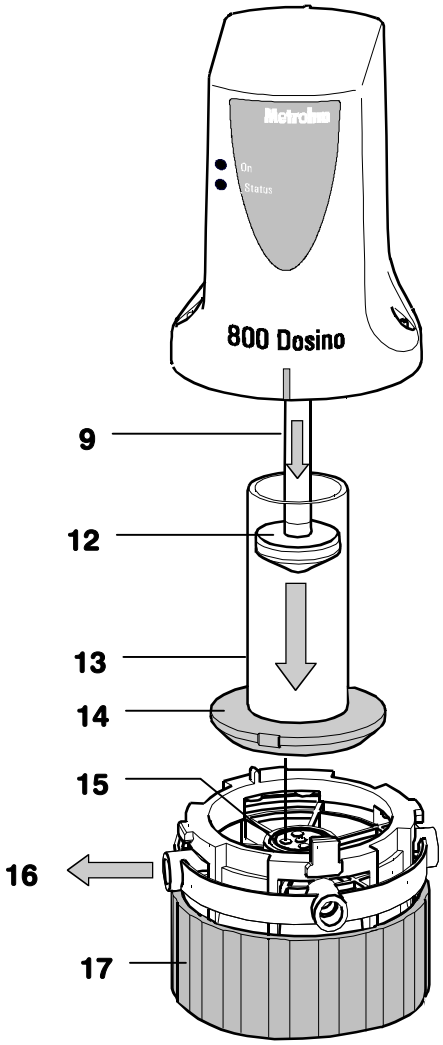


Fig. 5 Function: dosing/filling

1.3.1 Dosing/Filling the dosing cylinder

When the liquid is ejected push rod **9** of the 800 Dosino pushes dosing piston **12** in the cylinder downward. The liquid in cylinder **13** is forced through the valve disk in cylinder base **14** into one of the four openings in distributor disk **15**, depending on the valve disk position. In distributor **17** the liquid is transferred to a dosing port **16**.

When the valve disk has been switched (see below), i.e. rotated, then liquid is aspirated in the opposite direction through a different port by pulling dosing piston **12** upward with push rod **9** of the drive.

As the dosing units are exchangeable the coupling of push rod **9** has a slight mechanical tolerance that comes into effect when the direction of motion of piston **12** changes. In automated processes the drive compensates mechanically for this tolerance.

The piston movements are controlled by the accurate electronic precision engineering of the drive which, independent of the cylinder volume, has a resolution of 10 000 increments throughout the whole piston stroke.

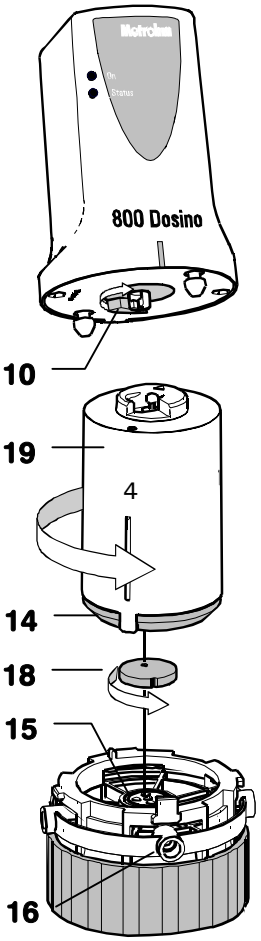


Fig. 6 Function: switching the valve disk

1.3.2 Switching the valve disk

The dosing unit has four ports. Two of them are on the outer housing and two on the bottom of the dosing unit. Depending on the position of black valve disk **18** a connection between cylinder **13** and the opening of white distributor disk **15** assigned to the port is made.

The drive attached to the dosing unit uses cam **10** to rotate centering tube **19** and therefore the whole inner cylinder unit with cylinder base **14** and valve disk **18** which it contains. After a rotation of the cylinder unit the opening of valve disk **18** points to a different opening in distributor disk **15**. This means that a different port **16** has been selected for dosing (or filling).

1.3.3 Port assignment

Distributor **17** of a dosing unit has four freely addressable inlets/outlets (ports) and an additional connection which leads directly to the lower side of distributor **17**. This VENT port, which cannot be addressed by the 4-way valve disk, is responsible for venting the storage bottle and can be fitted with a drying tube.

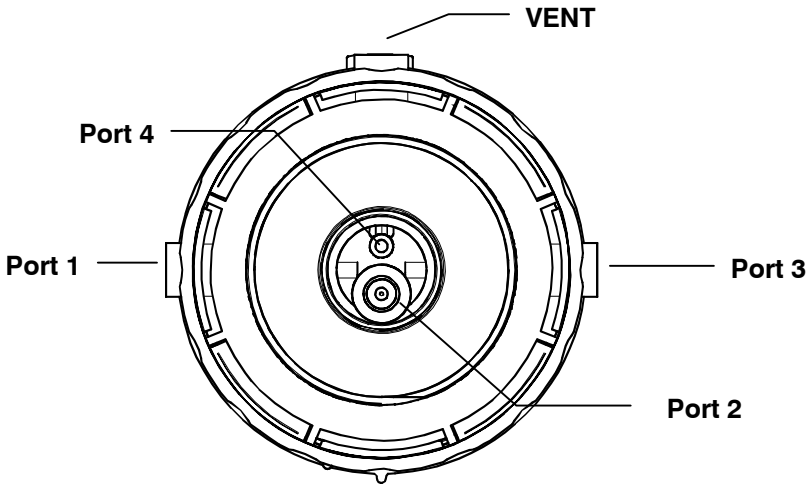


Fig. 7 Dosing unit seen from below

The inlets and outlets (ports) of the 807 Dosing unit can be used in different ways. This is an important precondition for complex liquid handling applications. Metrohm titrators (e.g. 808 and 809 Titrando) use a standard port assignment that is the most suitable one for titration applications.

1.3.4 Standard occupancy of the Dosino ports:

Port 1 Dosing outlet; M6 threaded connection on the left-hand side of the housing.

The liquid is ejected through a dosing tip or a titration tip.

Port 2 Filling inlet; M6 threaded connection on the base of the dosing unit.

The liquid is aspirated from a storage container.

Port 3 Not assigned; M6 threaded connection on the right-hand side of the housing.

Port 4 Special functions; narrow connection nipple on the base of the dosing unit. This can be used during the **'PREP'** function for ejecting the liquid. When the dosing unit is emptied port 4 is used as the air inlet.

VENT Vents the storage container; M6 threaded connection at the front. A drying tube can be connected to it, e.g. filled with molecular sieve or soda lime.

2 DLL Reference

This Reference applies to the interfaces of the programming languages C++, Delphi and Visual Basic. For descriptions of the Java interface, see 3.1.1

2.1 Constants

2.1.1 Return states

```
enum eReturnstate
{
   RET_STAT_OK                  = 0,
   RET_STAT_nvNumber            = 1,
   RET_STAT_noDosino            = 2,
   RET_STAT_commError           = 3,
   RET_STAT_argError            = 4,
   RET_STAT_nvAction            = 5
};
```

These are the return values of the Dosino functions, except **`Init846()`**.

`OK`	function could be set up correctly
`nvNumber`	not a valid 846 Dosing Interface number or Dosino number
`noDosino`	not a valid Dosino
`commError`	communication error
`argError`	function arguments out of specified range
`nvAction`	not a valid action

2.1.2 Dosino states

```
enum eDosinoState
{
   DOS_STAT_IDLE                = 0,
   DOS_STAT_FILL                = 1,
   DOS_STAT_EXCHANGE            = 2,
   DOS_STAT_POSITION            = 3,
   DOS_STAT_ZEROADJUST          = 4,
   DOS_STAT_ADJUST              = 5,
   DOS_STAT_COCK                = 6,
   DOS_STAT_TOENDDOS            = 7,
   DOS_STAT_DOS                 = 8,
   DOS_STAT_PREPAR              = 9,
   DOS_STAT_EMPTY               = 10,
   DOS_STAT_BUSY                = 11,
   DOS_STAT_HOLD_FILL           = 12,
   DOS_STAT_HOLD_EXCHANGE       = 13,
   DOS_STAT_HOLD_POSITION       = 14,
   DOS_STAT_HOLD_ZEROADJUST     = 15,
   DOS_STAT_HOLD_ADJUST         = 16,
   DOS_STAT_HOLD_TOENDDOS       = 17,
   DOS_STAT_HOLD_DOS            = 18,
   DOS_STAT_HOLD_PREPAR         = 19,
   DOS_STAT_HOLD_EMPTY          = 20,
   DOS_STAT_TIMEOUT             = 21,
   DOS_STAT_UNDEFINED           = 22
};
```

Status messages of a Dosino which are returned by the **`Status()`** function.

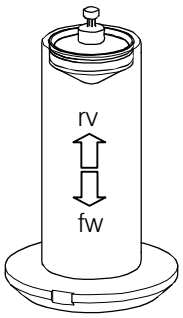
`IDLE`	Dosino is ready to execute a function
`FILL`	Cylinder is being filled
`EXCHANGE`	Dosino is executing Exchange function
`POSITION`	Dosino is executing GoPos function
`ZEROADJUST`	Dosino is executing ZeroAdjust function
`ADJUST`	Dosino is executing Adjust function
`COCK`	Dosino is executing Cock function
`TOENDDOS`	Dosino is executing ToEnd function
`DOS`	Dosino is executing MakeStep function
`PREPAR`	Dosino is executing Prep function
`EMPTY`	Dosino is executing Empty function
`BUSY`	Dosino is busy
`HOLD_FILL`	Dosino is halted while filling
`HOLD_EXCHANGE`	Dosino is halted in Exchange function
`HOLD_POSITION`	Dosino is halted in GoPos function
`HOLD_ZEROADJUST`	Dosino is halted in ZeroAdjust function
`HOLD_ADJUST`	Dosino is halted in Adjust function
`HOLD_TOENDDOS`	Dosino is halted in ToEnd function
`HOLD_DOS`	Dosino is halted in MakeStep function
`HOLD_PREPAR`	Dosino is halted in Prep function
`HOLD_EMPTY`	Dosino is halted in Empty function
`TIMEOUT`	Dosino is in timeout
`UNDEFINED`	Dosino is in undefined status

2.1.3 Piston movement

```
enum eDirection
{
   DIR_fwd                    =0,
   DIR_rev                    =1
};
```

Moving directions of a Dosino piston.

`fwd`	Forward, dosing (0 → 10000 pulses)
`rev`	Reverse, filling (10000 → 0 pulses)

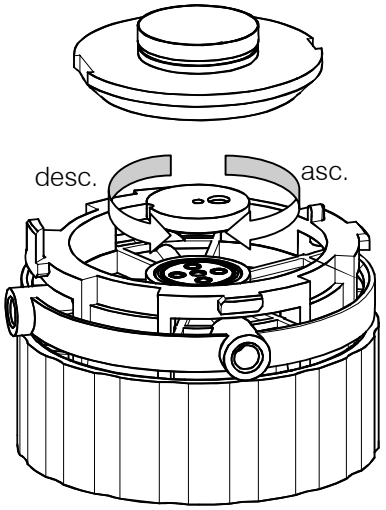


2.1.4 Valve disk

```
enum eCockMove
{
   CK_MV_Asc                =0,
   CK_MV_Desc               =1,
   CK_MV_Auto               =2,
   CK_MV_NotOver            =3
};
```

Turning mode of a Dosing Unit's valve disk. Port 1 to 4 of a Dosing Unit can be accessed in different orders.

`Asc`	Ascending order (Port 1 → 4)
`Desc`	Descending order (Port 4 → 1)
`Auto`	automatic mode, shortest path
`NotOver`	protected mode, a specified port will not be crossed



2.2 Functions

2.2.1 Init846

bool Init846()

Initializes all the connected Metrohm 846 Dosing Interfaces found in the USB chain. All Dosino drives are recognized and the Dosing Units are reset. Running processes are stopped.

This function returns true if everything is initialized correctly.

Input parameters

none

Output parameters

bool

2.2.2 GetProgramVersion

GetProgramVersion(long IfNo, char* ProgramVersion, long ProgVerBufSize)

Reads the program version of the Dosing Interface firmware.

Input parameters

IfNo	*long*	Dosing Interface [1…]
ProgVerBufSize	*long*	Number of characters

Output parameters

eReturnstate		return value
ProgramVersion	*char**	Text string of program version

2.2.3 GetInterfaceId

GetInterfaceId(long IfNo, long& InterfaceId)

Reads the serial number of a Dosing Interface. The serial number can be used as a unique identifier of a Dosing Interface.

Input parameters

IfNo	*long*	Dosing Interface [1…]

Output parameters

eReturnstate		return value
InterfaceId	*long*	serial number of Dosing Interface

2.2.4 Status

Status(long IfNo, long MsbNo, eDosinoState &DosinoState)

Reads the current status of a Dosino drive.

Input parameters

IfNo	*long*	Dosing Interface [1…]
MsbNo	*long*	Dosino at MSB[1…4]

Output parameters

eReturnstate		return value
DosinoState	*eDosinoState*	status of the Dosino drive

2.2.5 GetCylVolume

GetCylVolume(long IfNo, long MsbNo, long &Volume)

Returns the cylinder volume of the Dosing Unit. This function can be used for the detection of a mounted Dosing Unit.

Input parameters

IfNo	*long*	Dosing Interface [1…]
MsbNo	*long*	Dosino at MSB[1…4]

Output parameters

eReturnstate		return value
&Volume	*long*	0, 2; 5; 10; 20; 50 mL cylinder volume =0 means: no Dosing Unit mounted

2.2.6 DU_Cock

DU_Cock(long IfNo, long MsbNo, long Port, eCockMove Move, long NotOver)

Turns the valve disk to the selected port. The turning direction is chosen according to "Move". If Notover is set, the "NotOverPort" will not be crossed. Otherwise this parameter is ignored.

Input parameters

IfNo	*long*	Dosing Interface [1…]
MsbNo	*long*	Dosino at MSB[1…4]
Port	*long*	target Dosino port [1…4]
Move	*eCockMove*	turning direction of valve disk
NotOver	*long*	protected Dosino port

Output parameters

eReturnstate		return value

2.2.7 ZeroAdjust

`ZeroAdjust(long IfNo, long MsbNo, float RevRate)`

Runs the piston of the Dosino to zero position and initializes the drive.

Input parameters

`IfNo`	*long*	Dosing Interface [1…]
`MsbNo`	*long*	Dosino at MSB[1…4]
`RevRate`	*float*	Filling speed of piston [0.01...166mL/min]

Output parameters

`eReturnstate`	return value

2.2.8 Adjust

`Adjust(long IfNo, long MsbNo, eDirection Direction)`

Eliminate slack of piston coupling. This function can be used, if the direction of the piston movement is about to be changed.

Input parameters

`IfNo`	*long*	Dosing Interface [1…]
`MsbNo`	*long*	Dosino at MSB[1…4]
`Direction`	*eDirection*	Direction of the desired piston movement.

Output parameters

`eReturnstate`	return value

2.2.9 GoPos

`GoPos(long IfNo, long MsbNo, long Position, float Rate)`

Move the piston to an absolute position. The stroke path of the piston is divided into 10000 positions.

Input parameters

`IfNo`	*long*	Dosing Interface [1…]
`MsbNo`	*long*	Dosino at MSB[1…4]
`Position`	*long*	absolute piston position [0…10000]
`Rate`	*float*	Dosing rate [0.01…166 mL/min]

Output parameters

`eReturnstate`	return value

2.2.10 DU_ToEnd

DU_ToEnd(long IfNo, long MsbNo, float FwdRate)

Move piston to the mechanical end position. This function can be used to remove any air bubbles in the cylinder.

Input parameters

IfNo	*long*	Dosing Interface [1…]
MsbNo	*long*	Dosino at MSB[1…4]
FwdRate	*float*	Dosing rate [0.01…166 mL/min]

Output parameters

eReturnstate	return value

2.2.11 DU_MakeStep

DU_MakeStep(long IfNo, long MsbNo, long FillPort, float Volume, eDirection Direction, float FwdRate, float RevRate)

Dose a specified volume via the actual port. Dosing rate, filling rate, filling port and piston direction of the valve disk can be chosen. With reverse piston direction it is possible to aspirate a desired volume. In this case the filling port is used as outlet port. The dosing can be hold, continued and stopped.

Input parameters

IfNo	*long*	Dosing Interface [1…]
MsbNo	*long*	Dosino at MSB[1…4]
FillPort	*long*	Filling port
Volume	*float*	Dosing volume [0…99999.99 mL]
Direction	*eDirection*	Direction of piston movement
FwdRate	*float*	Dosing rate [0.01…166 mL/min]
RevRate	*float*	Filling rate [0.01…166 mL/min]

Output parameters

eReturnstate	return value

2.2.12 DU_Fill

DU_Fill(long IfNo, long MsbNo, long Port, float RevRate)

Fill the cylinder from a specified port. Filling can be hold and continued, but not be stopped.

Input parameters

IfNo	*long*	Dosing Interface [1…]
MsbNo	*long*	Dosino at MSB[1…4]
Port	*long*	Dosino port to fill from [1…4]
RevRate	*float*	Filling rate [0.01…166 mL/min]

Output parameters

eReturnstate	return value

2.2.13 DU_Exchange

```
DU_Exchange(long IfNo, long MsbNo, long Port, float RevRate)
```

Prepares the Dosing Unit for exchange. The cylinder is filled from the specified port and after that the valve disk is turned to port 2. This action can be hold and continued, but not be stopped.

Input parameters

`IfNo`	*long*	Dosing Interface [1…]
`MsbNo`	*long*	Dosino at MSB[1…4]
`Port`	*long*	Dosino port to fill from [1…4]
`RevRate`	*float*	Filling rate [0.01…166 mL/min]

Output parameters

eReturnstate	return value

2.2.14 DosStop

```
DosStop(long IfNo, long MsbNo)
```

Stops the current Dosino action.

Input parameters

`IfNo`	*long*	Dosing Interface [1…]
`MsbNo`	*long*	Dosino at MSB[1…4]

Output parameters

eReturnstate	return value

2.2.15 DosHold

```
DosHold(long IfNo, long MsbNo)
```

Holds the current Dosino action. The halted action can be continued or finally be stopped.

Input parameters

`IfNo`	*long*	Dosing Interface [1…]
`MsbNo`	*long*	Dosino at MSB[1…4]

Output parameters

eReturnstate	return value

2.2.16 DosContinue

DosContinue(long IfNo, long MsbNo)

Continue a halted Dosino action.

Input parameters

IfNo	*long*	Dosing Interface [1…]
MsbNo	*long*	Dosino at MSB[1…4]

Output parameters

eReturnstate	return value

2.2.17 DU_Prep

DU_Prep(long IfNo, long MsbNo, long InPort, float InVolume, float InRate, long OutPort, float OutVolume, float OutRate, long SpecPort, float SpecVolume, float SpecRate)

Prepares a Dosing Unit for further use. A "Prep" cycle includes emptying the dosing cylinder and rinsing and filling the tubings in one automated process.

Input parameters

IfNo	*long*	Dosing Interface [1…]
MsbNo	*long*	Dosino at MSB[1…4]
InPort	*long*	Filling port [1…4]
InVolume	*float*	Fill tube volume [0…20000 µL]
InRate	*float*	Filling rate [0.01…166 mL/min]
OutPort	*long*	Output port 1 [1…4]
OutVolume	*float*	Dosing volume 1 [0…20000 µL]
OutRate	*float*	Dosing rate 1 [0.01…166 mL/min]
SpecPort	*long*	Output Port 2 [1…4]
Specvolume	*float*	Dosing volume 2 [0…20000 µL]
SpecRate	float	Dosing rate 2 [0.01…166 mL/min]

Output parameters

eReturnstate	return value

2.2.18 Empty

Empty(long IfNo, long MsbNo, long InPort, float InVolume, float InRate, long OutPort, float OutVolume, float OutRate, long SpecPort, float SpecVolume, float SpecRate)

Empties the Buret Unit. Dosing cylinder and tubings are emptied in one automated process.

Input parameters

IfNo	*long*	Dosing Interface [1…]
MsbNo	*long*	Dosino at MSB[1…4]
InPort	*long*	Filling port [1…4]
InVolume	*float*	Fill tube volume [0…20000 µL]

`InRate`	*float*	Filling rate [0.01…166 mL/min]
`OutPort`	*long*	Output port 1 [1…4]
`OutVolume`	*float*	Dosing volume 1 [0…20000 µL]
`OutRate`	*float*	Dosing rate 1 [0.01…166 mL/min]
`SpecPort`	*long*	Output Port 2 [1…4]
`Specvolume`	*float*	Dosing volume 2 [0…20000 µL]
SpecRate	float	Dosing rate 2 [0.01…166 mL/min]

Output parameters

`eReturnstate`	return value

2.2.19 GetInterfaceError

```
GetInterfaceError(long IfNo, long MsbNo, long ErrorNumber,
char* ErrorCode, long ErrorBufSize)
```

Reads casual error messages which occur during Dosino actions.

Input parameters

`IfNo`	*long*	Dosing Interface [1…]
`MsbNo`	*long*	Dosino at MSB[1…4]
`Number`	*long*	Error number [0…9]
`ErrorBufSize`	*long*	Number of characters

Output parameters

`eReturnstate`		return value
`ErrorCode`	*char**	Error code, format [GGG-CCC-K-III], G = group, C = code, K = class, I = index (1...4 or 255), see 2.3

2.3 Error Messages

Error messages are presented in an array (Error number0…9), in which error number 0 is always the most recent one. If more errors occur in a process, more error entries are made. New errors always overwrite older ones, beginning with index 0.

Error messages of the function **`GetInterfaceError`**:

Error message **XXX-NNN-A-I**	*XXX=error group, NNN=Code, A=class, I=Index (mostly MSB number)*
000-001-1-255	Process already exists
000-002-1-255	Process not found
000-003-1-255	Too many processes
000-004-1-255	Wrong node
000-005-1-255	Trigger not allowed
000-006-1-255	Wrong value
005-001-1-X	Dosing device not found
005-002-1-X	Dosing device used
005-003-1-X	Check exchange/dosing unit
005-004-1-X	Empty not possible
005-005-1-X	Piston blocked
005-006-1-X	Cock blocked
005-007-2-X	Dosing rate was corrected
005-008-2-X	Filling rate was corrected
005-009-2-X	Dosing increment was corrected
005-010-3-X	Dosing unit was plugged in
005-011-3-X	Dosing unit was plugged out
005-012-2-X	Dosing unit out during reading
005-013-2-X	Dosing unit out during writing
005-015-1-X	Dosing deving connected incorr.
007-001-1-X	Stirrer not found
010-001-1-X	Remote device not found
010-003-1-X	Other devices at MSB
012-001-3-255	Manual stop of mode

3 Annex

3.1 Programming interface definitions

3.1.1 Java interface class

```
/*
 * Copyright(c) 2004 Metrohm AG
 *
 * $Archive: /tools/846_usb_dll/src/java_wrapper_846/src/metr/DosIntFace846.java $
 * $Date: 13.06.05 9:16 $
 * $Author: Km $
 * $Revision: 5 $
 */

package metr;

/**
 *
 * @author Michael Keller
 * @version 1.2.5.0, 09/06/05
 */

/**
 * DosIntFace846 class which contains all static methods to communicate with 846 Dosing
Interfaces.<br>
 * Befor communicating, the init method init846 has to be called.<br>
 * 846 Dosing Interfaces and Dosinos can't be hot pluged.<br>
 * <br>
 * If the cock move int parameter is described as eCockMove, the following description will be
used:<br>
 *      0: ascending order (Port 1 --> 4) <br>
 *      1: descending order (Port 4 --> 1) <br>
 *      2: automatic mode, shortest path <br>
 *      3: protected mode, cock will not move over specified port <br>
 * <br>
 * If the direction int parameter is described as eDirection, the following description will
be used:<br>
 *      0: forward, dosing   (0 --> 10000 pulses) <br>
 *      1: reverse, filling  (10000 --> 0 pulses) <br>
 *
 */
public class DosIntFace846 {

    // load the 846 DLL for native conaction to dosing interfaces
    static{
        System.loadLibrary("DosIntFace846");
    }


    public DosIntFace846() {}

    /**
     * Initializes all Metrohm 846 Dosing Interfaces found in the USB chain and sorts them
according to the serial number.
     * All Dosino drives are recognized and the Dosing Units are reset.
     * Running processes are stopped.
     *
     * @return boolean, method returns true if successful
     */
    public static final native boolean init846();

    /**
     * Returns the cylinder volume of the Dosing Unit.
     * This function can be used for the detection of a mounted Dosing Unit.
```

```
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @return Volume 0; 2; 5; 10; 20; 50 mL, Cylinder volume = 0 means: no Dosing Unit
mounted!
     * @throws DosIntFace846Exception
     */
    public static final int getCylVolume(int ifNo, int msbNo) throws DosIntFace846Exception{
        dosinoVolume = 0;
        int nReturnstate = DosIntFace846.callCylVolume(ifNo, msbNo);
        if (nReturnstate == 0)
            return dosinoVolume;
        else
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }

    /**
     * Turns the valve disk to the selected port. The turning direction is chosen
     * according to "Move". If "Move" is set to protected mode, the Port specified
     * in "NotOver" will not be crossed. Otherwise this parameter is ignored.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @param port Target Dosino port [1... 4]
     * @param move Turning direction of the valve disk [eCockMove]
     * @param notOver protected Dosino port [1... 4]
     * @throws DosIntFace846Exception
     */
    public static final void duCock(int ifNo, int msbNo, int port, int move, int notOver)
throws DosIntFace846Exception{
        int nReturnstate = DosIntFace846.callCock(ifNo, msbNo, port, move, notOver);
        if (nReturnstate != 0)
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }

    /**
     * Moves the piston of the Dosino to zero position and initializes the drive.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @param revRate Filling rate of piston [0.01...166mL/min]
     * @throws DosIntFace846Exception
     */
    public static final void zeroAdjust(int ifNo, int msbNo, float revRate) throws DosInt-
Face846Exception{
        int nReturnstate = DosIntFace846.callZeroAdjust(ifNo, msbNo, revRate);
        if (nReturnstate != 0)
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }

    /**
     * Eliminate slack of piston coupling.
     * This function can be used, when the direction of the piston movement is about to be
changed.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @param direction Direction of the desired piston movement [eDirection]
     * @throws DosIntFace846Exception
     */
    public static final void adjust (int ifNo, int msbNo, int direction) throws DosInt-
Face846Exception{
        int nReturnstate = DosIntFace846.callAdjust (ifNo, msbNo, direction);
        if (nReturnstate != 0)
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }
    /**
```

```
     * Move the piston to an absolute position. The full stroke of the piston is subdivided in
10000 positions.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @param position absolut Position [0...10000]
     * @param rate Dosing rate [0.01... 166 mL/min]
     * @throws DosIntFace846Exception
     */
    public static final void goPos(int ifNo, int msbNo, int position, float rate) throws
DosIntFace846Exception{
        int nReturnstate = DosIntFace846.callGoPos(ifNo, msbNo, position, rate);
        if (nReturnstate != 0)
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }
    /**
     * Move piston to the mechanical end position.
     * This function can be used to remove any air bubbles from the cylinder.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @param fwdRate Dosing rate [0.01... 166 mL/min]
     * @throws DosIntFace846Exception
     */
    public static final void duToEnd(int ifNo, int msbNo, float fwdRate) throws DosInt-
Face846Exception{
        int nReturnstate = DosIntFace846.callToEnd(ifNo, msbNo, fwdRate);
        if (nReturnstate != 0)
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }
    /**
     * Dose a specified volume via the actual port. Dosing rate, filling rate, filling port
and piston direction can be chosen.
     * With reverse piston direction it is possible to aspirate a desired volume. In this case
the filling port is used as outlet port.
     * The dosing can be hold, continued and stopped.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @param fillPort Filling port  [1...4]
     * @param volume Dosing volume [0... 99999.99 mL]
     * @param direction Direction of piston movement [eDirection]
     * @param fwdRate Dosing rate [0.01... 166 mL/min]
     * @param revRate Filling rate [0.01... 166 mL/min]
     * @throws DosIntFace846Exception
     */
    public static final void duMakeStep(int ifNo, int msbNo, int fillPort, float volume, int
direction, float fwdRate, float revRate) throws DosIntFace846Exception{
        int nReturnstate = DosIntFace846.callMakeStep(ifNo, msbNo, fillPort, volume, direc-
tion, fwdRate, revRate);
        if (nReturnstate != 0)
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }
    /**
     * Fill the cylinder from a specified port.
     * Filling can be held and continued, but not stopped.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @param port Fill port [1..4]
     * @param revRate Filling rate [0.01... 166 mL/min]
     * @throws DosIntFace846Exception
     */
    public static final void duFill(int ifNo, int msbNo, int port, float revRate) throws
DosIntFace846Exception{
        int nReturnstate = DosIntFace846.callFill(ifNo, msbNo, port, revRate);
        if (nReturnstate != 0)
```

```
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }
    /**
     * Prepares the Dosing Unit for exchange. The cylinder is filled from the specified port
and then the valve disk is
     * turned to port 2.
     * This action can be held and continued, but not stopped.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @param port Fill port [1..4]
     * @param revRate Filling rate [0.01... 166 mL/min]
     * @throws DosIntFace846Exception
     */
    public static final void duExchange(int ifNo, int msbNo, int port, float revRate)  throws
DosIntFace846Exception{
        int nReturnstate = DosIntFace846.callExchange(ifNo, msbNo, port, revRate);
        if (nReturnstate != 0)
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }


    /**
     * Reads the state of a Dosino drive.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @return  0  = Dosino is ready to execute a function, <br>
     *          1  = Cylinder is being filled, <br>
     *          2  = Dosino is executing Exchange function, <br>
     *          3  = Dosino is executing GoPos function, <br>
     *          4  = Dosino is executing ZeroAdjust function, <br>
     *          5  = Dosino is executing Adjust function, <br>
     *          6  = Dosino is executing Cock function, <br>
     *          7  = Dosino is executing ToEnd function, <br>
     *          8  = Dosino is executing MakeStep function, <br>
     *          9  = Dosino is executing Prep function, <br>
     *          10 = Dosino is executing Empty function, <br>
     *          11 = Dosino is busy, <br>
     *          12 = Dosino is halted while filling, <br>
     *          13 = Dosino is halted in Exchange function, <br>
     *          14 = Dosino is halted in GoPos function, <br>
     *          15 = Dosino is halted in ZeroAdjust function, <br>
     *          16 = Dosino is halted in Adjust function, <br>
     *          17 = Dosino is halted in ToEnd function, <br>
     *          18 = Dosino is halted in MakeStep function, <br>
     *          19 = Dosino is halted in Prep function, <br>
     *          20 = Dosino is halted in Empty function, <br>
     *          21 = Dosino has timed out, <br>
         *          22 = Dosino status is undefined<br>
     * @throws DosIntFace846Exception
     */
    public static final int status(int ifNo, int msbNo) throws DosIntFace846Exception{
        dosinoState = 22;
        int nReturnstate = DosIntFace846.callStatus(ifNo, msbNo);
        if (nReturnstate == 0)
            return dosinoState;
        else
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }

    /**
     * Stops the current Dosino action.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @throws DosIntFace846Exception
     */
```

```
    public static final void dosStop(int ifNo, int msbNo) throws DosIntFace846Exception{
        int nReturnstate = DosIntFace846.callStop(ifNo, msbNo);
        if (nReturnstate != 0)
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }

    /**
     * Holds the current Dosino action. The held action can be continued or finally stopped.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @throws DosIntFace846Exception
     */
    public static final void dosHold(int ifNo, int msbNo) throws DosIntFace846Exception{
        int nReturnstate = DosIntFace846.callHold(ifNo, msbNo);
        if (nReturnstate != 0)
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }

    /**
     * Continue a held Dosino action.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @throws DosIntFace846Exception
     */
    public static final void dosCont(int ifNo, int msbNo) throws DosIntFace846Exception{
        int nReturnstate = DosIntFace846.callCont(ifNo, msbNo);
        if (nReturnstate != 0)
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }

    /**
     * Prepares a Dosing Unit for further use. A "Prep" cycle includes emptying the dosing
cylinder and rinsing and
     * filling the tubings in one automated process.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @param inPort Filling port [1... 4]
     * @param inVolume Fill tube volume [0... 20000 mm^3]
     * @param inRate Filling rate [0.01... 166 mL/min]
     * @param outPort Output port 1 [1... 4]
     * @param outVolume Dosing tube volume on Output port 1 [0... 20000 mm^3]
     * @param outRate Dosing rate on Output port 1 [0.01... 166 mL/min]
     * @param specPort Output port 2 [1... 4]
     * @param specVolume Dosing tube volume on Output port 2 [0... 20000 mm^3]
     * @param specRate Dosing rate on Output port 1 [0.01... 166 mL/min]
     * @throws DosIntFace846Exception
     */
    public static final void duPrep(int ifNo, int msbNo, int inPort, float inVolume, float
inRate,
        int outPort, float outVolume, float outRate,
        int specPort, float specVolume, float specRate) throws DosIntFace846Exception{
        int nReturnstate = DosIntFace846.callPrep(ifNo, msbNo, inPort, inVolume, inRate,
        outPort, outVolume, outRate,
        specPort, specVolume, specRate);
        if (nReturnstate != 0)
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }

    /**
     * Empties the Buret Unit. Dosing cylinder and tubings are emptied in one automated
process.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
```

```
     * @param inPort Filling port [1... 4]
     * @param inVolume Fill tube volume [0... 20000 mm^3]
     * @param inRate Filling rate [0.01... 166 mL/min]
     * @param outPort Output port 1 [1... 4]
     * @param outVolume Dosing tube volume on Output port 1 [0... 20000 mm^3]
     * @param outRate Dosing rate on Output port 1 [0.01... 166 mL/min]
     * @param specPort Output port 2 [1... 4]
     * @param specVolume Dosing tube volume on Output port 2 [0... 20000 mm^3]
     * @param specRate Dosing rate on Output port 1 [0.01... 166 mL/min]
     * @throws DosIntFace846Exception
     */
    public static final void duEmpty(int ifNo, int msbNo, int inPort, float inVolume, float
inRate,
        int outPort, float outVolume, float outRate,
        int specPort, float specVolume, float specRate) throws DosIntFace846Exception{
        int nReturnstate = DosIntFace846.callEmpty(ifNo, msbNo, inPort, inVolume, inRate,
        outPort, outVolume, outRate,
        specPort, specVolume, specRate);
        if (nReturnstate != 0)
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }



    /**
     * Reads casual error messages which occur during Dosino actions.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @param errorNumber Error number [0... 9]
     * @return error code as return value [GGG-CCC-K-III], G = group, C = code, K = class, I =
index (1...4 or 255)
     * @throws DosIntFace846Exception
     */
    public static final String getInterfaceError(int ifNo, int msbNo, int errorNumber) throws
DosIntFace846Exception{
        errorCode = "";
        int nReturnstate = DosIntFace846.callInterfaceError(ifNo, msbNo, errorNumber);
        if (nReturnstate == 0)
            return errorCode;
        else
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }

    /**
     * Reads the stop condition of a Dosino action.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @param msbNo Dosino at MSB [1... 4]
     * @return Stop type of the last action
     * @throws DosIntFace846Exception
     */
    public static final int getStopType(int ifNo, int msbNo) throws DosIntFace846Exception{
        stopType = 0;
        int nReturnstate = DosIntFace846.callStopType(ifNo, msbNo);
        if (nReturnstate == 0)
            return stopType;
        else
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }

    /**
     * Reads the serial number of a Dosing Interface. The serial number can be used as a
unique identifier of an instrument.
     *
     * @param ifNo Dosing Interface [1... ?]
     * @return Serial number of Dosing Interface
     * @throws DosIntFace846Exception
```

```
     */
    public static final int getInterfaceId(int ifNo) throws DosIntFace846Exception{
        interfaceId = 0;
        int nReturnstate = DosIntFace846.callInterfaceId(ifNo);
        if (nReturnstate == 0)
            return interfaceId;
        else
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }

    /**
     * Get the program version of the 846 Dosing Interface
     *
     * @param ifNo Dosing Interface [1... ?]
     * @return string with the program version
     * @throws DosIntFace846Exception
     */
    public static final String getProgramVersion(int ifNo) throws DosIntFace846Exception{
        programVersion = "";
        int nReturnstate = DosIntFace846.callProgramVersion(ifNo);
         if (nReturnstate == 0)
            return programVersion;
        else
            throw new DosInt-
Face846Exception(DosIntFace846Exception.getErrorString(nReturnstate), nReturnstate);
    }



    //-------------------------------------------------------------------------------------
    //-------------------------  only for JNI, don't use it  ------------------------------

    /** only for JNI, don't use it */
    public static final native int callCylVolume(int ifNo, int msbNo);
    /** only for JNI, don't use it */
    public static final void setCylVolume(int iDosinoVolume) {dosinoVolume = iDosinoVolume;}
    /** only for JNI, don't use it */
    public static final native int callCock(int ifNo, int msbNo, int port, int move, int
notOver);
    /** only for JNI, don't use it */
    public static final native int callZeroAdjust(int ifNo, int msbNo, float revRate);
    /** only for JNI, don't use it */
    public static final native int callAdjust (int ifNo, int msbNo, int direction);
    /** only for JNI, don't use it */
    public static final native int callGoPos(int ifNo, int msbNo, int position, float rate);
    /** only for JNI, don't use it */
    public static final native int callToEnd(int ifNo, int msbNo, float fwdRate);
    /** only for JNI, don't use it */
    public static final native int callMakeStep(int ifNo, int msbNo, int fillPort, float
volume, int direction, float fwdRate, float revRate);
    /** only for JNI, don't use it */
    public static final native int callFill(int ifNo, int msbNo, int port, float revRate);
    /** only for JNI, don't use it */
    public static final native int callExchange(int ifNo, int msbNo, int port, float revRate);
    /** only for JNI, don't use it */
    public static final native int callStop(int ifNo, int msbNo);
    /** only for JNI, don't use it */
    public static final native int callHold(int ifNo, int msbNo);
    /** only for JNI, don't use it */
    public static final native int callCont(int ifNo, int msbNo);
    /** only for JNI, don't use it */
    public static final native int callPrep(int ifNo, int msbNo, int inPort, float inVolume,
float inRate,
    int outPort, float outVolume, float outRate,
    int specPort, float specVolume, float specRate);
    /** only for JNI, don't use it */
    public static final native int callEmpty(int ifNo, int msbNo, int inPort, float inVolume,
float inRate,
    int outPort, float outVolume, float outRate,
    int specPort, float specVolume, float specRate);
```

```
    /** only for JNI, don't use it */
    public static final native int callStatus(int ifNo, int msbNo);
    /** only for JNI, don't use it */
    public static final void setDosinoState(int iDosinoState) {dosinoState = iDosinoState;}
    /** only for JNI, don't use it */
    public static final native int callInterfaceError(int ifNo, int msbNo, int errorNumber);
    /** only for JNI, don't use it */
    public static final void setErrorCode(String sErrorCode) {errorCode = sErrorCode;}
    /** only for JNI, don't use it */
    public static final native int callStopType(int ifNo, int msbNo);
    /** only for JNI, don't use it */
    public static final void setStopType(int iStopType) {stopType = iStopType;}
    /** only for JNI, don't use it */
    public static final native int callInterfaceId(int ifNo);
    /** only for JNI, don't use it */
    public static final void setInterfaceId(int lInterfaceId) {interfaceId = lInterfaceId;}
    /** only for JNI, don't use it */
    public static final native int callProgramVersion(int ifNo);
    /** only for JNI, don't use it */
    public static final void setProgramVersion(String sProgramVersion) {programVersion =
sProgramVersion;}

    // private data members
    private static int dosinoVolume = 0;
    private static int dosinoState = 0;
    private static String errorCode = "";
    private static int stopType = 0;
    private static int interfaceId = 0;
    private static String programVersion = "";

}
```

3.1.2 Java interface exception class

```
/*
 * Copyright(c) 2005 Metrohm AG
 * created date: 09.06.2005
 * $Archive:
/tools/846_usb_dll/src/java_wrapper_846/src/metr/DosIntFace846Exception.jav
a $
 * $Date: 9.06.05 15:46 $
 * $Author: Km $
 * $Revision: 2 $
 */

package metr;

/**
 *
 * @author Michael Keller
 * @version 1.2.5.0, 09/06/05
 */
/**
 * Exception class for 846 Dosing Interface communication
 * <br>
 * Specification of the error code number:
 *      1:  not a valid 846 Dosing Interface number or Dosino number <br>
 *      2:  not a valid Dosino <br>
 *      3:  communication error <br>
 *      4:  function arguments out of specified range <br>
 *      5:  not a valid action <br>
 */
public class DosIntFace846Exception extends Exception{

    private int nErrorCode;

    public DosIntFace846Exception(String s,int nErrorCode) {
        super(s);
        this.nErrorCode = nErrorCode;
    }

    public int getErrorCode() {
        return nErrorCode;
    }

    public static String getErrorString(int nErrorCode) {
        switch (nErrorCode) {
            case 0:  return "method could be set up correctly";
            case 1:  return "not a valid 846 Dosing Interface number or
Dosino number";
            case 2:  return "not a valid Dosino";
            case 3:  return "communication error";
            case 4:  return "function arguments out of specified range";
            case 5:  return "not a valid action";
            default:  return "unknown error code";
        }

    }


}

/*
```

3.1.3 C++ interface definition header file

```
/*****************************************************************************

   Metrohm AG Switzerland. All rights reserved.

   $Header:
/Tools/846_USB_DLL/src/846_Dosing_Interface/846DosingInterfaceDefinition.h        7
27.05.05 16:39 Km $

 ----------------------------------------------------------------------------
   $Log:  /Tools/846_USB_DLL/src/846_Dosing_Interface/846DosingInterfaceDefinition.h
$
 *
 * 7     27.05.05 16:39 Km
 *
 * 6     27.05.05 11:54 Km
 * Change in Interface
 *
 * 6     12.05.05 15:30 bc, rw --> parameter names adapted
 *
 * 5     06.01.05 13:52 Km
 *
 * 4     05.01.05 10:45 Km
 *
 * 3     04.01.05 14:32 Km
 *
 * 2     03.01.05 15:29 Km
 *
 * 1     15.12.04 15:36 Km

   $NoKeywords: $
*****************************************************************************/

#ifndef Dosinginterface846_h
#define Dosinginterface846_h


#ifdef _USRDLL
#define DllDirection __declspec( dllexport )
#else
#define DllDirection __declspec( dllimport )
#endif

enum eReturnstate
{
    // function could be set up correctly
    RET_STAT_OK             =  0,
    // not a valid 846 Dosing Interface number or Dosino number
    RET_STAT_nvNumber       =  1,
    // not a valid Dosino
    RET_STAT_noDosino       =  2,
    // communication error
    RET_STAT_commError      =  3,
    // function arguments out of specified range
    RET_STAT_argError       =  4,
    // not a valid action
    RET_STAT_nvAction       =  5
};
```

```
enum eDosinoState
{
    // Dosino is ready to execute a function
    DOS_STAT_IDLE            =  0,
    // Cylinder is being filled
    DOS_STAT_FILL            =  1,
    // Dosino is executing Exchange function
    DOS_STAT_EXCHANGE        =  2,
    // Dosino is executing GoPos function
    DOS_STAT_POSITION        =  3,
    // Dosino is executing ZeroAdjust function
    DOS_STAT_ZEROADJUST      =  4,
    // Dosino is executing Adjust function
    DOS_STAT_ADJUST          =  5,
    // Dosino is executing Cock function
    DOS_STAT_COCK            =  6,
    // Dosino is executing ToEnd function
    DOS_STAT_TOENDDOS        =  7,
    // Dosino is executing MakeStep function
    DOS_STAT_DOS             =  8,
    // Dosino is executing Prep function
    DOS_STAT_PREPAR          =  9,
    // Dosino is executing Empty function
    DOS_STAT_EMPTY           = 10,
    // Dosino is busy
    DOS_STAT_BUSY            = 11,
    // Dosino is halted while filling
    DOS_STAT_HOLD_FILL       = 12,
    // Dosino is halted in Exchange function
    DOS_STAT_HOLD_EXCHANGE   = 13,
    // Dosino is halted in GoPos function
    DOS_STAT_HOLD_POSITION   = 14,
    // Dosino is halted in ZeroAdjust function
    DOS_STAT_HOLD_ZEROADJUST = 15,
    // Dosino is halted in Adjust function
    DOS_STAT_HOLD_ADJUST     = 16,
    // Dosino is halted in ToEnd function
    DOS_STAT_HOLD_TOENDDOS   = 17,
    // Dosino is halted in MakeStep function
    DOS_STAT_HOLD_DOS        = 18,
    // Dosino is halted in Prep function
    DOS_STAT_HOLD_PREPAR     = 19,
    // Dosino is halted in Empty function
    DOS_STAT_HOLD_EMPTY      = 20,
    // Dosino has timed out
    DOS_STAT_TIMEOUT         = 21,
    // Dosino status is undefined
    DOS_STAT_UNDEFINED       = 22
};

enum eDirection
{
    // Forward, dosing   (0 --> 10000 pulses)
    DIR_fwd                  =  0,
    // Reverse, filling  (10000 --> 0 pulses)
    DIR_rev                  =  1
};

enum eCockMove
{
    // Ascending order (Port 1 --> 4)
    CK_MV_Asc                =  0,
    // Descending order (Port 4 --> 1)
```

```
    CK_MV_Desc              =  1,
    // Automatic mode, shortest path
    CK_MV_Auto              =  2,
    // Protected mode, cock will not move over specified port
    CK_MV_NotOver           =  3
};

#ifdef __cplusplus
// C-Declarations for C++
extern "C" {
#endif

// Initializes all Metrohm 846 Dosing Interfaces found in the USB chain.
// All Dosino drives are recognized and the Dosing Units are reset.
// Running processes are stopped.
//
// return       bool                            Function returns true if
//                                              successful
DllDirection bool __stdcall Init846();


// Returns the cylinder volume of the Dosing Unit. This function can be used for
// the detection of a mounted Dosing Unit.
//
// parameter    long            IfNo            Dosing Interface [1... ?]
//              long            MsbNo           Dosino at MSB [1... 4]
//              long&           Volume          0; 2; 5; 10; 20; 50 mL
//                                              Cylinder volume = 0
//                                              means: no Dosing Unit mounted!
// return       eReturnstate
DllDirection eReturnstate __stdcall GetCylVolume(long IfNo, long MsbNo,
                                                 long &Volume);

// Turns the valve disk to the selected port. The turning direction is chosen
// according to "Move". If "Move" is set to CK_MV_NotOver, the Port specified
// in "NotOver" will not be crossed. Otherwise this parameter is ignored.
//
// parameter    long            IfNo            Dosing Interface [1... ?]
//              long            MsbNo           Dosino at MSB [1... 4]
//              long            Port            Target Dosino port [1... 4]
//              eCockMove       Move            Turning direction of the
//                                              valve disk
//              long            NotOver         protected Dosino port [1... 4]
// return       eReturnstate
DllDirection eReturnstate __stdcall DU_Cock(long IfNo, long MsbNo, long Port,
                                            eCockMove Move, long NotOver);


// Moves the piston of the Dosino to zero position and initializes the drive.
//
// parameter    long            IfNo            Dosing Interface [1... ?]
//              long            MsbNo           Dosino at MSB [1... 4]
//              float           RevRate         Filling rate of piston
//                                              [0.01...166mL/min]
// return       eReturnstate
DllDirection eReturnstate __stdcall ZeroAdjust(long IfNo, long MsbNo,
                                               float RevRate);


// Eliminate slack of piston coupling. This function can be used, when the
// direction of the piston movement is about to be changed.
//
// parameter    long            IfNo            Dosing Interface [1... ?]
```

```
//              long            MsbNo             Dosino at MSB [1... 4]
//              eDirection      Direction         Direction of the desired
//                                                piston movement.
// return       eReturnstate
DllDirection eReturnstate __stdcall Adjust(long IfNo, long MsbNo,
                                           eDirection Direction);


// Move the piston to an absolute position. The full stroke of the piston is
// subdivided in 10000 positions.
//
// parameter    long            IfNo              Dosing Interface [1... ?]
//              long            MsbNo             Dosino at MSB [1... 4]
//              long            Position          Absolute piston position
//                                                [0... 10000]
//              float           Rate              Dosing rate
//                                                [0.01... 166 mL/min]
// return       eReturnstate
DllDirection eReturnstate __stdcall GoPos(long IfNo, long MsbNo, long Position,
                                          float Rate);


// Move piston to the mechanical end position. This function can be used to
// remove any air bubbles from the cylinder.
//
// parameter    long            IfNo              Dosing Interface [1... ?]
//              long            MsbNo             Dosino at MSB [1... 4]
//              float           FwdRate           Dosing rate
//                                                [0.01... 166 mL/min]
// return       eReturnstate
DllDirection eReturnstate __stdcall DU_ToEnd(long IfNo, long MsbNo,
                                             float FwdRate);


// Dose a specified volume via the actual port. Dosing rate, filling rate,
// filling port and piston direction can be chosen.
// With reverse piston direction it is possible to aspirate a desired volume.
// In this case the filling port is used as outlet port.
// The dosing can be hold, continued and stopped.
//
// parameter    long            IfNo              Dosing Interface [1... ?]
//              long            MsbNo             Dosino at MSB [1... 4]
//              long            FillPort          Filling port
//              float           Volume            Dosing volume
//                                                [0... 99999.99 mL]
//              eDirection      Direction         Direction of piston movement
//              float           FwdRate           Dosing rate
//                                                [0.01... 166 mL/min]
//              float           RevRate           Filling rate
//                                                [0.01... 166 mL/min]
// return       eReturnstate
DllDirection eReturnstate __stdcall DU_MakeStep(long IfNo, long MsbNo,
                                                long Fillport, float Volume,
                                                eDirection Direction,
                                                float FwdRate, float RevRate);


// Fill the cylinder from a specified port.
// Filling can be held and continued, but not stopped.
//
// parameter    long            IfNo              Dosing Interface [1... ?]
//              long            MsbNo             Dosino at MSB [1... 4]
//              long            Port              Fill port [1..4]
```

```
//              float           RevRate           Filling rate
//                                                [0.01... 166 mL/min]
// return       eReturnstate
DllDirection eReturnstate __stdcall DU_Fill(long IfNo, long MsbNo, long Port,
                                            float RevRate);


// Prepares the Dosing Unit for exchange. The cylinder is filled from the
// specified port and then the valve disk is turned to port 2.
// This action can be held and continued, but not stopped.
//
// parameter    long            IfNo              Dosing Interface [1... ?]
//              long            MsbNo             Dosino at MSB [1... 4]
//              long            Port              Fill port [1..4]
//              float           RevRate           Filling rate
//                                                [0.01... 166 mL/min]
// return       eReturnstate
DllDirection eReturnstate __stdcall DU_Exchange(long IfNo, long MsbNo, long Port,
                                                float RevRate);


// Reads the status of a Dosino drive.
//
// parameter    long            IfNo              Dosing Interface [1... ?]
//              long            MsbNo             Dosino at MSB [1... 4]
//              eDosinoState&   DosinoState       status of the Dosino drive
// return       eReturnstate
DllDirection eReturnstate __stdcall Status(long IfNo, long MsbNo,
                                           eDosinoState &DosinoState);


// Stops the current Dosino action.
//
// parameter    long            IfNo              Dosing Interface [1... ?]
//              long            MsbNo             Dosino at MSB [1... 4]
// return       eReturnstate
DllDirection eReturnstate __stdcall DosStop(long IfNo, long MsbNo);


// Holds the current Dosino action. The held action can be continued or
// finally stopped.
//
// parameter    long            IfNo              Dosing Interface [1... ?]
//              long            MsbNo             Dosino at MSB [1... 4]
// return       eReturnstate
DllDirection eReturnstate __stdcall DosHold(long IfNo, long MsbNo);


// Continue a held Dosino action.
//
// parameter    long            IfNo              Dosing Interface [1... ?]
//              long            MsbNo             Dosino at MSB [1... 4]
// return       eReturnstate
DllDirection eReturnstate __stdcall DosContinue(long IfNo, long MsbNo);


// Prepares a Dosing Unit for further use. A "Prep" cycle includes emptying the
// dosing cylinder and rinsing and filling the tubings in one automated process.
//
// parameter    long            IfNo              Dosing Interface [1... ?]
//              long            MsbNo             Dosino at MSB [1... 4]
//              long            InPort            Filling port [1... 4]
//              float           InVolume          Fill tube volume
```

```
//                                                  [0... 20000 mm^3]
//              float           InRate              Filling rate
//                                                  [0.01... 166 mL/min]
//              long            OutPort             Output port 1 [1... 4]
//              float           OutVolume           Dosing tube volume on Output
//                                                  port 1 [0... 20000 mm^3]
//              float           OutRate             Dosing rate on Output port 1
//                                                  [0.01... 166 mL/min]
//              long            SpecPort            Output port 2 [1... 4]
//              float           SpecVolume          Dosing tube volume on Output
//                                                  port 2 [0... 20000 mm^3]
//              float           SpecRate            Dosing rate on Output port 1
//                                                  [0.01... 166 mL/min]
// return       eReturnstate
DllDirection eReturnstate __stdcall DU_Prep(long IfNo, long MsbNo,
                                            long InPort, float InVolume,
                                            float InRate,
                                            long OutPort, float OutVolume,
                                            float OutRate,
                                            long SpecPort, float SpecVolume,
                                            float SpecRate);


// Empties the Buret Unit. Dosing cylinder and tubings are emptied in one
// automated process.
//
// parameter    long            IfNo                Dosing Interface [1... ?]
//              long            MsbNo               Dosino at MSB [1... 4]
//              long            InPort              Filling port [1... 4]
//              float           InVolume            Fill tube volume
//                                                  [0... 20000 mm^3]
//              float           InRate              Filling rate
//                                                  [0.01... 166 mL/min]
//              long            OutPort             Output port 1 [1... 4]
//              float           OutVolume           Dosing tube volume on Output
//                                                  port 1 [0... 20000 mm^3]
//              float           OutRate             Dosing rate on Output port 1
//                                                  [0.01... 166 mL/min]
//              long            SpecPort            Output port 2 [1... 4]
//              float           SpecVolume          Dosing tube volume on Output
//                                                  port 2 [0... 20000 mm^3]
//              float           SpecRate            Dosing rate on Output port 1
//                                                  [0.01... 166 mL/min]
// return       eReturnstate
DllDirection eReturnstate __stdcall DU_Empty(long IfNo, long MsbNo,
                                            long InPort, float InVolume,
                                            float InRate,
                                            long OutPort, float OutVolume,
                                            float OutRate,
                                            long SpecPort, float SpecVolume,
                                            float SpecRate);


// Reads casual error messages which occur during Dosino actions.
//
// parameter    long            IfNo                Dosing Interface [1... ?]
//              long            MsbNo               Dosino at MSB [1... 4]
//              long            ErrorNumber         Error number [0... 9]
//              char*           ErrorCode           Error code, format
//                                                  [GGG-CCC-K-III],
//                                                  G = group, C = code,
//                                                  K = class, I = index
//                                                  (1...4 or 255)
```

```
//              long            ErrorBufSize      Number of characters of
//                                                ErrorCode including a
//                                                terminating null character
//                                                (at least 14)
// return       eReturnstate
DllDirection eReturnstate __stdcall GetInterfaceError(long IfNo, long MsbNo,
                                                      long ErrorNumber,
                                                      char* ErrorCode,
                                                      long ErrorBufSize);


// Reads the stop condition of a Dosino action.
//
// parameter    long            IfNo              Dosing Interface [1... ?]
//              long            MsbNo             Dosino at MSB [1... 4]
//              long&           StopType          Stop type of the last action
// return       eReturnstate
DllDirection eReturnstate __stdcall GetStopType(long IfNo, long MsbNo,
                                                long &StopType);


// Reads the serial number of a Dosing Interface. The serial number can be used
// as a unique identifier of an instrument.
//
// parameter    long            IfNo              Dosing Interface [1... ?]
//              long&           InterfaceId       Serial number of Dosing
//                                                Interface
// return       eReturnstate
DllDirection eReturnstate __stdcall GetInterfaceId(long IfNo,
                                                   long &InterfaceId);


// Get the program version of the 846 Dosing Interface
//
// parameter    long            IfNo              Dosing Interface [1... ?]
//              char*           ProgramVersion    Text string of program version
//              long            ProgVerBufSize    Number of characters of
//                                                ProgramVersion including a
//                                                 terminating null character
// return       eReturnstate
DllDirection eReturnstate __stdcall GetProgramVersion(long IfNo,
                                                      char* ProgramVersion,
                                                      long ProgVerBufSize);


#ifdef __cplusplus
}
#endif

#endif // Dosinginterface846_h
```

3.1.4 Delphi Interface

```
(*****************************************************************************)
(*                                                                           *)
(*  Metrohm AG Switzerland. All rights reserved.                             *)
(*                                                                           *)
(*---------------------------------------------------------------------------*)
(*                                                                           *)
(* 1     30.05.05 14:28 rw                                                   *)
(*                                                                           *)
(*****************************************************************************)

UNIT DosIntFace846;
{$MINENUMSIZE 4 }

INTERFACE

  CONST
    dllName = '846_Dosing_Interface.dll';

  TYPE
    LONG  = LongInt;
    FLOAT = Single;
    BOOL  = LongBool;

    eReturnState = (
      (* function could be set up correctly *)
      RET_STAT_OK,
      (* not a valid 846 Dosing Interface number or Dosino number *)
      RET_STAT_nvNumber,
      (* not a valid Dosino *)
      RET_STAT_noDosino,
      (* communication error *)
      RET_STAT_commError,
      (* function arguments out of specified range *)
      RET_STAT_argError,
      (* not a valid action *)
      RET_STAT_nvAction);

    eDosinoState = (
      (* Dosino is ready to execute a function *)
      DOS_STAT_IDLE,
      (* Cylinder is being filled *)
      DOS_STAT_FILL,
      (* Dosino is executing Exchange function *)
      DOS_STAT_EXCHANGE,
      (* Dosino is executing GoPos function *)
      DOS_STAT_POSITION,
      (* Dosino is executing ZeroAdjust function *)
      DOS_STAT_ZEROADJUST,
      (* Dosino is executing Adjust function *)
      DOS_STAT_ADJUST,
      (* Dosino is executing Cock function *)
      DOS_STAT_COCK,
      (* Dosino is executing ToEnd function *)
      DOS_STAT_TOENDDOS,
      (* Dosino is executing MakeStep function *)
      DOS_STAT_DOS,
      (* Dosino is executing Prep function *)
      DOS_STAT_PREPAR,
      (* Dosino is executing Empty function *)
      DOS_STAT_EMPTY,
```

```
      (* Dosino is busy *)
      DOS_STAT_BUSY,
      (* Dosino is halted while filling *)
      DOS_STAT_HOLD_FILL,
      (* Dosino is halted in Exchange function *)
      DOS_STAT_HOLD_EXCHANGE,
      (* Dosino is halted in GoPos function *)
      DOS_STAT_HOLD_POSITION,
      (* Dosino is halted in ZeroAdjust function *)
      DOS_STAT_HOLD_ZEROADJUST,
      (* Dosino is halted in Adjust function *)
      DOS_STAT_HOLD_ADJUST,
      (* Dosino is halted in ToEnd function *)
      DOS_STAT_HOLD_TOENDDOS,
      (* Dosino is halted in MakeStep function *)
      DOS_STAT_HOLD_DOS,
      (* Dosino is halted in Prep function *)
      DOS_STAT_HOLD_PREPAR,
      (* Dosino is halted in Empty function *)
      DOS_STAT_HOLD_EMPTY,
      (* Dosino has timed out *)
      DOS_STAT_TIMEOUT,
      (* Dosino status is undefined *)
      DOS_STAT_UNDEFINED);

    eDirection = (
      (* Forward, dosing   (0 --> 10000 pulses) *)
      DIR_fwd,
      (* Reverse, filling  (10000 --> 0 pulses) *)
      DIR_rev);

    eCockMove  = (
      (* Ascending order (Port 1 --> 4) *)
      CK_MV_Asc,
      (* Descending order (Port 4 --> 1) *)
      CK_MV_Desc,
      (* Automatic mode, shortest path *)
      CK_MV_Auto,
      (* Protected mode, cock will not move over specified port *)
      CK_MV_NotOver);


(* Initializes all Metrohm 846 Dosing Interfaces found in the USB chain.      *)
(* All Dosino drives are recognized and the Dosing Units are reset.           *)
(* Running processes are stopped.                                             *)
(*                                                                            *)
(* return       bool                              Function returns true if    *)
(*                                                successful                  *)
    FUNCTION Init846 : BOOL; STDCALL; EXTERNAL dllName;


(* Returns the cylinder volume of the Dosing Unit. This function can be       *)
(* used for the detection of a mounted Dosing Unit.                           *)
(*                                                                            *)
(* parameter    LONG            IfNo        Dosing Interface [1... ?]         *)
(*              LONG            MsbNo       Dosino at MSB [1... 4]            *)
(*              LONG        VAR Volume      0; 2; 5; 10; 20; 50 mL            *)
(*                                          Cylinder volume = 0               *)
(*                                          means: no Dosing Unit mounted!    *)
(* return       eReturnstate                                                  *)
    FUNCTION GetCylVolume(IfNo,MsbNo : LONG; VAR Volume : LONG)
     : eReturnState; STDCALL; EXTERNAL dllName;
```

```
(* Turns the valve disk to the selected port. The turning direction is        *)
(* chosen according to "Move". If "Move" is set to CK_MV_NotOver, the Port    *)
(* specified in "NotOver" will not be crossed. Otherwise this parameter is    *)
(* ignored.                                                                   *)
(*                                                                            *)
(* parameter    LONG            IfNo        Dosing Interface [1... ?]         *)
(*              LONG            MsbNo       Dosino at MSB [1... 4]            *)
(*              LONG            Port        Target Dosino port [1... 4]       *)
(*              eCockMove       Move        Turning direction of the          *)
(*                                          valve disk                        *)
(*              LONG            NotOver     protected Dosino port [1... 4]    *)
(* return       eReturnstate                                                  *)
    FUNCTION DU_Cock(IfNo,MsbNo,Port : LONG; Move : eCockMove; NotOver : LONG)
     : eReturnState; STDCALL; EXTERNAL dllName;


(* Moves the piston of the Dosino to zero position and initializes the drive. *)
(*                                                                            *)
(* parameter    LONG            IfNo              Dosing Interface [1... ?]   *)
(*              LONG            MsbNo             Dosino at MSB [1... 4]      *)
(*              FLOAT           RevRate           Filling rate of piston      *)
(*                                                [0.01...166mL/min]          *)
(* return       eReturnstate                                                  *)
    FUNCTION ZeroAdjust(IfNo,MsbNo : LONG; RevRate : FLOAT) : eReturnState;
     STDCALL; EXTERNAL dllName;


(* Eliminate slack of piston coupling. This function can be used, when the    *)
(* direction of the piston movement is about to be changed.                   *)
(*                                                                            *)
(* parameter    LONG            IfNo              Dosing Interface [1... ?]   *)
(*              LONG            MsbNo             Dosino at MSB [1... 4]      *)
(*              eDirection      Direction         Direction of the desired    *)
(*                                                piston movement.            *)
(* return       eReturnstate                                                  *)
    FUNCTION Adjust(IfNo,MsbNo : LONG; Direction : eDirection) : eReturnState;
     STDCALL; EXTERNAL dllName;


(* Move the piston to an absolute position. The full stroke of the piston is  *)
(* subdivided in 10000 positions.                                             *)
(*                                                                            *)
(* parameter    LONG            IfNo              Dosing Interface [1... ?]   *)
(*              LONG            MsbNo             Dosino at MSB [1... 4]      *)
(*              LONG            Position          Absolute piston position    *)
(*                                                [0... 10000]                *)
(*              FLOAT           Rate              Dosing rate                 *)
(*                                                [0.01... 166 mL/min]        *)
(* return       eReturnstate                                                  *)
    FUNCTION GoPos(IfNo,MsbNo,Position : LONG; Rate : FLOAT) : eReturnState;
     STDCALL; EXTERNAL dllName;


(* Move piston to the mechanical end position. This function can be used to   *)
(* remove any air bubbles from the cylinder.                                  *)
(*                                                                            *)
(* parameter    LONG            IfNo              Dosing Interface [1... ?]   *)
(*              LONG            MsbNo             Dosino at MSB [1... 4]      *)
(*              FLOAT           FwdRate           Dosing rate                 *)
(*                                                [0.01... 166 mL/min]        *)
(* return       eReturnstate                                                  *)
    FUNCTION DU_ToEnd(IfNo,MsbNo : LONG; FwdRate : FLOAT) : eReturnState;
```

```
     STDCALL; EXTERNAL dllName;


(* Dose a specified volume via the actual port. Dosing rate, filling rate,    *)
(* filling port and piston direction can be chosen.                           *)
(* With reverse piston direction it is possible to aspirate a desired volume. *)
(* In this case the filling port is used as outlet port.                      *)
(* The dosing can be hold, continued and stopped.                             *)
(*                                                                            *)
(* parameter    LONG            IfNo             Dosing Interface [1... ?]    *)
(*              LONG            MsbNo            Dosino at MSB [1... 4]       *)
(*              LONG            FillPort         Filling port                 *)
(*              FLOAT           Volume           Dosing volume                *)
(*                                               [0... 99999.99 mL]           *)
(*              eDirection      Direction        Direction of piston movement *)
(*              FLOAT           FwdRate          Dosing rate                  *)
(*                                               [0.01... 166 mL/min]         *)
(*              FLOAT           RevRate          Filling rate                 *)
(*                                               [0.01... 166 mL/min]         *)
(* return       eReturnstate                                                  *)
    FUNCTION DU_MakeStep(IfNo,MsbNo : LONG; FillPort : LONG;
                         Volume : FLOAT; Direction : eDirection;
                         FwdRate,RevRate : FLOAT) : eReturnState;
     STDCALL; EXTERNAL dllName;


(* Fill the cylinder from a specified port.                                   *)
(* Filling can be held and continued, but not stopped.                        *)
(*                                                                            *)
(* parameter    LONG            IfNo              Dosing Interface [1... ?]   *)
(*              LONG            MsbNo             Dosino at MSB [1... 4]      *)
(*              LONG            Port              Fill port [1..4]            *)
(*              FLOAT           RevRate           Filling rate                *)
(*                                                [0.01... 166 mL/min]        *)
(* return       eReturnstate                                                  *)
    FUNCTION DU_Fill(IfNo,MsbNo,port : LONG; RevRate : FLOAT) : eReturnState;
     STDCALL; EXTERNAL dllName;


(* Prepares the Dosing Unit for exchange. The cylinder is filled from the     *)
(* specified port and then the valve disk is turned to port 2.                *)
(* This action can be held and continued, but not stopped.                    *)
(*                                                                            *)
(* parameter    LONG            IfNo              Dosing Interface [1... ?]   *)
(*              LONG            MsbNo             Dosino at MSB [1... 4]      *)
(*              LONG            Port              Fill port [1..4]            *)
(*              FLOAT           RevRate           Filling rate                *)
(*                                                [0.01... 166 mL/min]        *)
(* return       eReturnstate                                                  *)
    FUNCTION DU_Exchange(IfNo,MsbNo,Port : LONG; RevRate : FLOAT)
     : eReturnState; STDCALL; EXTERNAL dllName;


(* Reads the status of a Dosino drive.                                        *)
(*                                                                            *)
(* parameter    LONG             IfNo              Dosing Interface [1... ?]  *)
(*              LONG             MsbNo             Dosino at MSB [1... 4]     *)
(*              eDosinoState VAR DosinoState       status of the Dosino drive *)
(* return       eReturnstate                                                  *)
    FUNCTION Status(IfNo,MsbNo : LONG; VAR DosinoState : eDosinoState)
     : eReturnState; STDCALL; EXTERNAL dllName;
```

```
(* Stops the current Dosino action.                                          *)
(*                                                                           *)
(* parameter    LONG            IfNo             Dosing Interface [1... ?]   *)
(*              LONG            MsbNo            Dosino at MSB [1... 4]      *)
(* return       eReturnstate                                                 *)
    FUNCTION DosStop(IfNo,MsbNo : LONG) : eReturnState;
     STDCALL; EXTERNAL dllName;


(* Holds the current Dosino action. The held action can be continued or      *)
(* finally stopped.                                                          *)
(*                                                                           *)
(* parameter    LONG            IfNo             Dosing Interface [1... ?]   *)
(*              LONG            MsbNo            Dosino at MSB [1... 4]      *)
(* return       eReturnstate                                                 *)
    FUNCTION DosHold(IfNo,MsbNo : LONG) : eReturnState;
     STDCALL; EXTERNAL dllName;


(* Continues a held Dosino action.                                           *)
(*                                                                           *)
(* parameter    LONG            IfNo             Dosing Interface [1... ?]   *)
(*              LONG            MsbNo            Dosino at MSB [1... 4]      *)
(* return       eReturnstate                                                 *)
    FUNCTION DosContinue(IfNo,MsbNo : LONG) : eReturnState;
     STDCALL; EXTERNAL dllName;


(* Prepares a Dosing Unit for further use. A "Prep" cycle includes emptying  *)
(* the dosing cylinder and rinsing and filling the tubings in one automated  *)
(* process.                                                                  *)
(*                                                                           *)
(* parameter    LONG            IfNo            Dosing Interface [1... ?]    *)
(*              LONG            MsbNo           Dosino at MSB [1... 4]       *)
(*              LONG            InPort          Filling port [1... 4]        *)
(*              FLOAT           InVolume        Fill tube volume             *)
(*                                              [0... 20000 mm^3]            *)
(*              FLOAT           InRate          Filling rate                 *)
(*                                              [0.01... 166 mL/min]         *)
(*              LONG            OutPort         Output port 1 [1... 4]       *)
(*              FLOAT           OutVolume       Dosing tube volume on Output *)
(*                                              port 1 [0... 20000 mm^3]     *)
(*              FLOAT           OutRate         Dosing rate on Output port 1 *)
(*                                              [0.01... 166 mL/min]         *)
(*              LONG            SpecPort        Output port 2 [1... 4]       *)
(*              FLOAT           SpecVolume      Dosing tube volume on Output *)
(*                                              port 2 [0... 20000 mm^3]     *)
(*              FLOAT           SpecRate        Dosing rate on Output port 1 *)
(*                                              [0.01... 166 mL/min]         *)
(* return       eReturnstate                                                 *)
    FUNCTION DU_Prep(IfNo,MsbNo,InPort : LONG; InVolume,InRate : FLOAT;
                     OutPort : LONG; OutVolume,OutRate : FLOAT;
                     SpecPort : LONG; SpecVolume,SpecRate : FLOAT)
     : eReturnState; STDCALL; EXTERNAL dllName;


(* Empties the Buret Unit. Dosing cylinder and tubings are emptied in one    *)
(* automated process.                                                        *)
(*                                                                           *)
(* parameter    LONG            IfNo            Dosing Interface [1... ?]    *)
(*              LONG            MsbNo           Dosino at MSB [1... 4]       *)
(*              LONG            InPort          Filling port [1... 4]        *)
(*              FLOAT           InVolume        Fill tube volume             *)
```

```
(*                                              [0... 20000 mm^3]           *)
(*             FLOAT           InRate           Filling rate                *)
(*                                              [0.01... 166 mL/min]        *)
(*             LONG            OutPort          Output port 1 [1... 4]      *)
(*             FLOAT           OutVolume        Dosing tube volume on Output *)
(*                                              port 1 [0... 20000 mm^3]    *)
(*             FLOAT           OutRate          Dosing rate on Output port 1 *)
(*                                              [0.01... 166 mL/min]        *)
(*             LONG            SpecPort         Output port 2 [1... 4]      *)
(*             FLOAT           SpecVolume       Dosing tube volume on Output *)
(*                                              port 2 [0... 20000 mm^3]    *)
(*             FLOAT           SpecRate         Dosing rate on Output port 1 *)
(*                                              [0.01... 166 mL/min]        *)
(* return      eReturnstate                                                 *)
    FUNCTION DU_Empty(IfNo,MsbNo,InPort : LONG; InVolume,InRate : FLOAT;
                      OutPort : LONG; OutVolume,OutRate : FLOAT;
                      SpecPort : LONG; SpecVolume,SpecRate : FLOAT)
     : eReturnState; STDCALL; EXTERNAL dllName;


(* Reads casual error messages which occur during Dosino actions.           *)
(*                                                                          *)
(* parameter   LONG            IfNo             Dosing Interface [1... ?]  *)
(*             LONG            MsbNo            Dosino at MSB [1... 4]     *)
(*             LONG            ErrorNumber      Error number [0... 9]      *)
(*             AnsiString  VAR ErrorCode        Error code, format         *)
(*                                              [GGG-CCC-K-III],           *)
(*                                              G = group, C = code,       *)
(*                                              K = class, I = index       *)
(*                                              (1...4 or 255)             *)
(* return      eReturnstate                                                 *)
    FUNCTION GetInterfaceError(IfNo,MsbNo,ErrorNumber : LONG;
                               VAR ErrorCode : AnsiString) : eReturnState;


(* Reads the stop condition of a Dosino action.
(*
(* parameter   LONG             IfNo            Dosing Interface [1... ?]   *)
(*             LONG             MsbNo           Dosino at MSB [1... 4]      *)
(*             LONG         VAR StopType        Stop type of the last action *)
(* return      eReturnstate                                                 *)
    FUNCTION GetStopType(IfNo,MsbNo : LONG; VAR StopType : LONG)
     : eReturnState; STDCALL; EXTERNAL dllName;


(* Reads the serial number of a Dosing Interface. The serial number can be  *)
(* used as a unique identifier of an instrument.                            *)
(*                                                                          *)
(* parameter   LONG            IfNo             Dosing Interface [1... ?]  *)
(*             LONG        VAR InterfaceId      Serial number of Dosing    *)
(*                                              Interface                  *)
(* return      eReturnstate                                                 *)
    FUNCTION GetInterfaceId(IfNo : LONG; VAR InterfaceId : LONG)
     : eReturnState; STDCALL; EXTERNAL dllName;


(* Get the program version of the 846 Dosing Interface                      *)
(*                                                                          *)
(* parameter   long            IfNo          Dosing Interface [1... ?]     *)
(*             AnsiString  VAR ProgramVersion Text string of program version *)
(* return      eReturnstate                                                 *)
    FUNCTION GetProgramVersion(IfNo : LONG; VAR ProgramVersion : AnsiString)
     : eReturnState;
```

```
IMPLEMENTATION

    FUNCTION GetErr(IfNo,MsbNo,Number : LONG; VAR Buf; BufCapacity : LONG)
     : eReturnState; STDCALL; EXTERNAL dllName NAME 'GetError';

    FUNCTION GetInterfaceError(IfNo,MsbNo,ErrorNumber : LONG;
                               VAR ErrorCode : AnsiString) : eReturnState;
    CONST
      maxErrCodeLen = 14;
    VAR
      eCodeArr : ARRAY[0..maxErrCodeLen - 1] OF Char;
    BEGIN
      result := GetErr(IfNo,MsbNo,ErrorNumber,eCodeArr,maxErrCodeLen);
      errorCode := eCodeArr;
    END;

    FUNCTION GetProgVer(IfNo : LONG; VAR buf; bufCapacity : LONG)
     : eReturnState; STDCALL; EXTERNAL dllName NAME 'GetProgramVersion';

    FUNCTION GetProgramVersion(IfNo : LONG; VAR ProgramVersion : AnsiString)
     : eReturnState;
    CONST
      maxProgVerLen = 32;
    VAR
      progVerArr : ARRAY[0..maxProgVerLen] OF Char;
    BEGIN
      result := GetProgVer(IfNo,progVerArr,maxProgVerLen);
      ProgramVersion := progVerArr
    END;

INITIALIZATION

FINALIZATION

END.
```

3.1.5 Visual Basic Interface

```
Attribute VB_Name = "Module1"
'*****************************************************************************
'*                                                                           *
'*  Metrohm AG Switzerland. All rights reserved.                             *
'*                                                                           *
'*---------------------------------------------------------------------------*
'*                                                                           *
'* 1     31.05.05 14:28 rw                                                   *
'*                                                                           *
'*****************************************************************************

Option Explicit

Public Enum eReturnState
  ' function could be set up correctly
  RET_STAT_OK
  ' not a valid 846 Dosing Interface number or Dosino number
  RET_STAT_nvNumber
  ' not a valid Dosino
  RET_STAT_noDosino
  ' communication error
  RET_STAT_commError
  ' function arguments out of specified range
  RET_STAT_argError
  ' not a valid action
  RET_STAT_nvAction
End Enum

Public Enum eDosinoState
  ' Dosino is ready to execute a function
  DOS_STAT_IDLE
  ' Cylinder is being filled
  DOS_STAT_FILL
  ' Dosino is executing Exchange function
  DOS_STAT_EXCHANGE
  ' Dosino is executing GoPos function
  DOS_STAT_POSITION
  ' Dosino is executing ZeroAdjust function
  DOS_STAT_ZEROADJUST
  ' Dosino is executing Adjust function
  DOS_STAT_ADJUST
  ' Dosino is executing Cock function
  DOS_STAT_COCK
  ' Dosino is executing ToEnd function
  DOS_STAT_TOENDDOS
  ' Dosino is executing MakeStep function
  DOS_STAT_DOS
  ' Dosino is executing Prep function
  DOS_STAT_PREPAR
  ' Dosino is executing Empty function
  DOS_STAT_EMPTY
  ' Dosino is busy
  DOS_STAT_BUSY
  ' Dosino is halted while filling
  DOS_STAT_HOLD_FILL
  ' Dosino is halted in Exchange function
  DOS_STAT_HOLD_EXCHANGE
  ' Dosino is halted in GoPos function
  DOS_STAT_HOLD_POSITION
  ' Dosino is halted in ZeroAdjust function
```

```
  DOS_STAT_HOLD_ZEROADJUST
  ' Dosino is halted in Adjust function
  DOS_STAT_HOLD_ADJUST
  ' Dosino is halted in ToEnd function
  DOS_STAT_HOLD_TOENDDOS
  ' Dosino is halted in MakeStep function
  DOS_STAT_HOLD_DOS
  ' Dosino is halted in Prep function
  DOS_STAT_HOLD_PREPAR
  ' Dosino is halted in Empty function
  DOS_STAT_HOLD_EMPTY
  ' Dosino has timed out
  DOS_STAT_TIMEOUT
  ' Dosino status is undefined
  DOS_STAT_UNDEFINED
End Enum

Public Enum eDirection
  ' Forward, dosing   (0 --> 10000 pulses)
  DIR_fwd
  ' Reverse, filling  (10000 --> 0 pulses)
  DIR_rev
End Enum

Public Enum eCockMove
  ' Ascending order (Port 1 --> 4)
  CK_MV_Asc
  ' Descending order (Port 4 --> 1)
  CK_MV_Desc
  ' Automatic mode, shortest path
  CK_MV_Auto
  ' Protected mode, cock will not move over specified port
  CK_MV_NotOver
End Enum


' Initializes all Metrohm 846 Dosing Interfaces found in the USB chain.
' All Dosino drives are recognized and the Dosing Units are reset.
' Running processes are stopped.
'
' return       Boolean                          Function returns true if
'                                               successful
Declare Function Init846 Lib "846_Dosing_Interface.dll" () As Boolean


' Returns the cylinder volume of the Dosing Unit. This function can be
' used for the detection of a mounted Dosing Unit.
'
' parameter    Long            IfNo        Dosing Interface [1... ?]
'              Long            MsbNo       Dosino at MSB [1... 4]
'              Long      ByRef Volume      0; 2; 5; 10; 20; 50 mL
'                                          Cylinder volume = 0
'                                          means: no Dosing Unit mounted!
' return       eReturnstate
Declare Function GetCylVolume Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByVal MsbNo As Long, ByRef Volume As Long) _
  As eReturnState


' Turns the valve disk to the selected port. The turning direction is
' chosen according to "Move". If "Move" is set to CK_MV_NotOver, the Port
' specified in "NotOver" will not be crossed. Otherwise this parameter is
' ignored.
```

```
'
' parameter    Long            IfNo        Dosing Interface [1... ?]
'              Long            MsbNo       Dosino at MSB [1... 4]
'              Long            Port        Target Dosino port [1... 4]
'              eCockMove       Move        Turning direction of the
'                                          valve disk
'              Long            NotOver     protected Dosino port [1... 4]
' return       eReturnstate
Declare Function DU_Cock Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByVal MsbNo As Long, ByVal Port As Long, _
   ByVal Move As eCockMove, ByVal NotOver As Long) As eReturnState


' Moves the piston of the Dosino to zero position and initializes the drive.
'
' parameter    Long            IfNo              Dosing Interface [1... ?]
'              Long            MsbNo             Dosino at MSB [1... 4]
'              Single          RevRate           Filling rate of piston
'                                                [0.01...166mL/min]
' return       eReturnstate
Declare Function ZeroAdjust Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByVal MsbNo As Long, ByVal RevRate As Single) _
  As eReturnState


' Eliminate slack of piston coupling. This function can be used, when the
' direction of the piston movement is about to be changed.
'
' parameter    Long            IfNo              Dosing Interface [1... ?]
'              Long            MsbNo             Dosino at MSB [1... 4]
'              eDirection      Direction         Direction of the desired
'                                                piston movement.
' return       eReturnstate
Declare Function Adjust Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByVal MsbNo As Long, ByVal Direction As eDirection) _
  As eReturnState


' Move the piston to an absolute position. The full stroke of the piston is
' subdivided in 10000 positions.
'
' parameter    Long            IfNo              Dosing Interface [1... ?]
'              Long            MsbNo             Dosino at MSB [1... 4]
'              Long            Position          Absolute piston position
'                                                [0... 10000]
'              Single          Rate              Dosing rate
'                                                [0.01... 166 mL/min]
' return       eReturnstate
Declare Function GoPos Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByVal MsbNo As Long, ByVal Position As Long, _
   ByVal Rate As Single) As eReturnState


' Move piston to the mechanical end position. This function can be used to
' remove any air bubbles from the cylinder.
'
' parameter    Long            IfNo              Dosing Interface [1... ?]
'              Long            MsbNo             Dosino at MSB [1... 4]
'              Single          FwdRate           Dosing rate
'                                                [0.01... 166 mL/min]
' return       eReturnstate
 Declare Function DU_ToEnd Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByVal MsbNo As Long, ByVal FwdRate As Single) _
```

```
  As eReturnState


' Dose a specified volume via the actual port. Dosing rate, filling rate,
' filling port and piston direction can be chosen.
' With reverse piston direction it is possible to aspirate a desired volume.
' In this case the filling port is used as outlet port.
' The dosing can be hold, continued and stopped.
'
' parameter    Long            IfNo             Dosing Interface [1... ?]
'              Long            MsbNo            Dosino at MSB [1... 4]
'              Long            FillPort         Filling port
'              Single          Volume           Dosing volume
'                                               [0... 99999.99 mL]
'              eDirection      Direction        Direction of piston movement
'              Single          FwdRate          Dosing rate
'                                               [0.01... 166 mL/min]
'              Single          RevRate          Filling rate
'                                               [0.01... 166 mL/min]
' return       eReturnstate
Declare Function DU_MakeStep Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByVal MsbNo As Long, ByVal FillPort As Long, _
   ByVal Volume As Single, ByVal Direction As eDirection, _
   ByVal FwdRate As Single, ByVal RevRate As Single) _
  As eReturnState


' Fill the cylinder from a specified port.
' Filling can be held and continued, but not stopped.
'
' parameter    Long            IfNo              Dosing Interface [1... ?]
'              Long            MsbNo             Dosino at MSB [1... 4]
'              Long            Port              Fill port [1..4]
'              Single          RevRate           Filling rate
'                                                [0.01... 166 mL/min]
' return       eReturnstate
Declare Function DU_Fill Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByVal MsbNo As Long, ByVal Port As Long, _
   ByVal RevRate As Single) As eReturnState


' Prepares the Dosing Unit for exchange. The cylinder is filled from the
' specified port and then the valve disk is turned to port 2.
' This action can be held and continued, but not stopped.
'
' parameter    Long            IfNo              Dosing Interface [1... ?]
'              Long            MsbNo             Dosino at MSB [1... 4]
'              Long            Port              Fill port [1..4]
'              Single          RevRate           Filling rate
'                                                [0.01... 166 mL/min]
' return       eReturnstate
Declare Function DU_Exchange Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByVal MsbNo As Long, ByVal Port As Long, _
   ByVal RevRate As Single) As eReturnState


' Reads the status of a Dosino drive.
'
' parameter    Long               IfNo           Dosing Interface [1... ?]
'              Long               MsbNo          Dosino at MSB [1... 4]
'              eDosinoState ByRef DosinoState    status of the Dosino drive
' return       eReturnstate
Declare Function Status Lib "846_Dosing_Interface.dll" _
```

```
  (ByVal IfNo As Long, ByVal MsbNo As Long, _
   ByRef DosinoState As eDosinoState) As eReturnState


' Stops the current Dosino action.
'
' parameter    Long            IfNo              Dosing Interface [1... ?]
'              Long            MsbNo             Dosino at MSB [1... 4]
' return       eReturnstate
Declare Function DosStop Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByVal MsbNo As Long) As eReturnState


' Holds the current Dosino action. The held action can be continued or
' finally stopped.
'
' parameter    Long            IfNo              Dosing Interface [1... ?]
'              Long            MsbNo             Dosino at MSB [1... 4]
' return       eReturnstate
Declare Function DosHold Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByVal MsbNo As Long) As eReturnState


' Continues a held Dosino action.
'
' parameter    Long            IfNo              Dosing Interface [1... ?]
'              Long            MsbNo             Dosino at MSB [1... 4]
' return       eReturnstate
Declare Function DosContinue Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByVal MsbNo As Long) As eReturnState


' Prepares a Dosing Unit for further use. A "Prep" cycle includes emptying
' the dosing cylinder and rinsing and filling the tubings in one automated
' process.
'
' parameter    Long            IfNo              Dosing Interface [1... ?]
'              Long            MsbNo             Dosino at MSB [1... 4]
'              Long            InPort            Filling port [1... 4]
'              Single          InVolume          Fill tube volume
'                                                [0... 20000 mm^3]
'              Single          InRate            Filling rate
'                                                [0.01... 166 mL/min]
'              Long            OutPort           Output port 1 [1... 4]
'              Single          OutVolume         Dosing tube volume on Output
'                                                port 1 [0... 20000 mm^3]
'              Single          OutRate           Dosing rate on Output port 1
'                                                [0.01... 166 mL/min]
'              Long            SpecPort          Output port 2 [1... 4]
'              Single          SpecVolume        Dosing tube volume on Output
'                                                port 2 [0... 20000 mm^3]
'              Single          SpecRate          Dosing rate on Output port 1
'                                                [0.01... 166 mL/min]
' return       eReturnstate
Declare Function DU_Prep Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByVal MsbNo As Long, ByVal InPort As Long, _
   ByVal InVolume As Single, ByVal InRate As Single, _
   ByVal OutPort As Long, ByVal OutVolume As Single, _
   ByVal OutRate As Single, _
   ByVal SpecPort As Long, ByVal SpecVolume As Single, _
   ByVal SpecRate As Single) _
  As eReturnState
```

```
' Empties the Buret Unit. Dosing cylinder and tubings are emptied in one
' automated process.
'
' parameter    Long            IfNo            Dosing Interface [1... ?]
'              Long            MsbNo           Dosino at MSB [1... 4]
'              Long            InPort          Filling port [1... 4]
'              Single          InVolume        Fill tube volume
'                                              [0... 20000 mm^3]
'              Single          InRate          Filling rate
'                                              [0.01... 166 mL/min]
'              Long            OutPort         Output port 1 [1... 4]
'              Single          OutVolume       Dosing tube volume on Output
'                                              port 1 [0... 20000 mm^3]
'              Single          OutRate         Dosing rate on Output port 1
'                                              [0.01... 166 mL/min]
'              Long            SpecPort        Output port 2 [1... 4]
'              Single          SpecVolume      Dosing tube volume on Output
'                                              port 2 [0... 20000 mm^3]
'              Single          SpecRate        Dosing rate on Output port 1
'                                              [0.01... 166 mL/min]
' return       eReturnstate
Declare Function DU_Empty Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByVal MsbNo As Long, ByVal InPort As Long, _
   ByVal InVolume As Single, ByVal InRate As Single, ByVal OutPort As Long, _
   ByVal OutVolume As Single, ByVal OutRate As Single, _
   ByVal SpecPort As Long, ByVal SpecVolume As Single, _
   ByVal SpecRate As Single) As eReturnState


' Reads the stop condition of a Dosino action.
'
' parameter    Long         IfNo               Dosing Interface [1... ?]
'              Long         MsbNo              Dosino at MSB [1... 4]
'              Long         ByRef StopType     Stop type of the last action
' return       eReturnstate
Declare Function GetStopType Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByVal MsbNo As Long, ByRef StopType As Long) _
  As eReturnState


' Reads the serial number of a Dosing Interface. The serial number can be
' used as a unique identifier of an instrument.
'
' parameter    Long               IfNo          Dosing Interface [1... ?]
'              Long         ByRef InterfaceId   Serial number of Dosing
'                                               Interface
' return       eReturnstate
Declare Function GetInterfaceId Lib "846_Dosing_Interface.dll" _
  (ByVal IfNo As Long, ByRef InterfaceId As Long) As eReturnState


Declare Function GetErr Lib "846_Dosing_Interface.dll" Alias "GetInterfaceError" _
  (ByVal IfNo As Long, ByVal MsbNo As Long, ByVal number As Long, _
   ByVal buf As String, ByVal bufCapacity As Long) As eReturnState


Declare Function GetProgVer Lib "846_Dosing_Interface.dll" _
          Alias "GetProgramVersion" _
  (ByVal IfNo As Long, ByVal buf As String, ByVal bufCapacity As Long) _
  As eReturnState
```

```
Private Function TruncString(ByVal s As String) As String
  Dim l As Long
  l = InStr(s, Chr(0))
  If (l > 0) Then
    TruncString = Left(s, l - 1)
  Else
    TruncString = s
  End If
End Function


' Reads casual error messages which occur during Dosino actions.
'
' parameter    Long            IfNo              Dosing Interface [1... ?]
'              Long            MsbNo             Dosino at MSB [1... 4]
'              Long            ErrorNumber       Error number [0... 9]
'              String    ByRef ErrorCode         Error code, format
'                                                [GGG-CCC-K-III],
'                                                G = group, C = code,
'                                                K = class, I = index
'                                                (1...4 or 255)
' return       eReturnstate
Public Function GetInterfaceError(ByVal IfNo As Long, ByVal MsbNo As Long, _
  ByVal ErrorNumber As Long, ByRef ErrorCode As String) As eReturnState

  Dim temp As String

  temp = String(255, vbNullChar)
  GetInterfaceError = GetErr(IfNo, MsbNo, ErrorNumber, temp, 255)
  ErrorCode = TruncString(temp)
End Function


' Get the program version of the 846 Dosing Interface
'
' parameter    Long         IfNo                 Dosing Interface [1... ?]
'              String       ByRef ProgramVersion Text string of program version
' return       eReturnstate
Public Function GetProgramVersion(ByVal IfNo As Long, _
  ByRef ProgramVersion As String) As eReturnState

  Dim temp As String

  temp = String(255, vbNullChar)
  GetProgramVersion = GetProgVer(IfNo, temp, 255)
  ProgramVersion = TruncString(temp)
End Function
```