Product Specification

Product Specification ï ALS-G DLL API

For Product:	ALSG XYZ Sampler	Version ALSG_API.dll:	1.40
Document Number:	aijw10035	Document Revision:	H
Author:	Jan Willem Mol	Date:	23 Dec 2014

Document Approvals

Marketing	Engineering	Customer

Document Revision History

Revision	Date	Name	Changes With Reasons
A	4 Jan 2010	Jan Willem Mol	Initial version
B	29 Apr 2010	Jan Willem Mol	Minor Modifications
C	25 Mar 2011	Jan Willem Mol	Update to V3Rev04C
D	8 Apr 2011	Jan Willem Mol	Fixed Function Prototype typos
E	19 Apr 2011	Jan Willem Mol	Added functions for UI App (alsgGetBusy, alsgRunReference and others)
F	12 Oct 2011	Jan Willem Mol	Added Syringe Exchange specifications

G	16 Jan 2012	Jan Willem Mol	- Teaching UI functions added to DLL - alsgConnect modified to allow auto-discovery of instrument - alsgSetIO() function added to control front LEDs - Added functions to specify solvent rinse volume - alsgSetMethodSampleWashVol() added - alsgSetMethodPreInjWashVol() added - alsgSetMethodPostInjWashVol() added - alsgRunInit(void) added - alsgSetMethodInjectDepth(double dDepth) added
H	23 Dec 2014	Piet de Jager	2nd Wash Station 2 extra Sample Trays Special z-Axis motion API calls to read actual positions and sensor states

Table of Contents

1 Introduction

1.1 General

This document describes the available functions in the ALS-G DLL API that allow full control of the instrument for routine use. This API is not (yet) intended for more complex sample automation tasks such as derivitisation, complex sample preparation, mixing of samples etc.

The API is based on:

- a parameter file in which configuration and installation data is stored
- a set of API calls that allow the DLL to be intialized
- a set of API calls (as an alternative of reading to parameter file) to modify the **installation** settings
- a set of API calls (as an alternative of reading to parameter file) to modify the **configuration** settings
- a set of API calls that allow to change **method** settings
- a set of API **immediate** calls to trigger activity such as starting an analytical run
- a set of API **MicroCode** calls that allow for more detailed control of the instrument outside the context of a method
- a set of API **diagnostics** function calls

Note: None of the installation, configuration or method settings are permanent! After restart of the DLL all related internal variables will be in uninitialized or default state and will have to be refreshed by reading the parameter file and/or individually calling relevant API exported functions.

Furthermore the logic of the sampler behavior is based the reference points of the tray, wash stations etc. are stored as well as various other installation and configuration information. This information which is stored in the parameter file is usually generated with a separate application that allows for configuration and teaching of the instrument.

1.2 ALS-G hardware structure

As can be seen in diagram in Fig. 1. below the ALS-G can be controlled by one of three physical interfaces. Since both the USB and Ethernet interface use a virtual COM-Port the underlying communication protocol is the same in all three cases.

The interface between the alsg_API.dll and the software layer on top of it is described in this document.

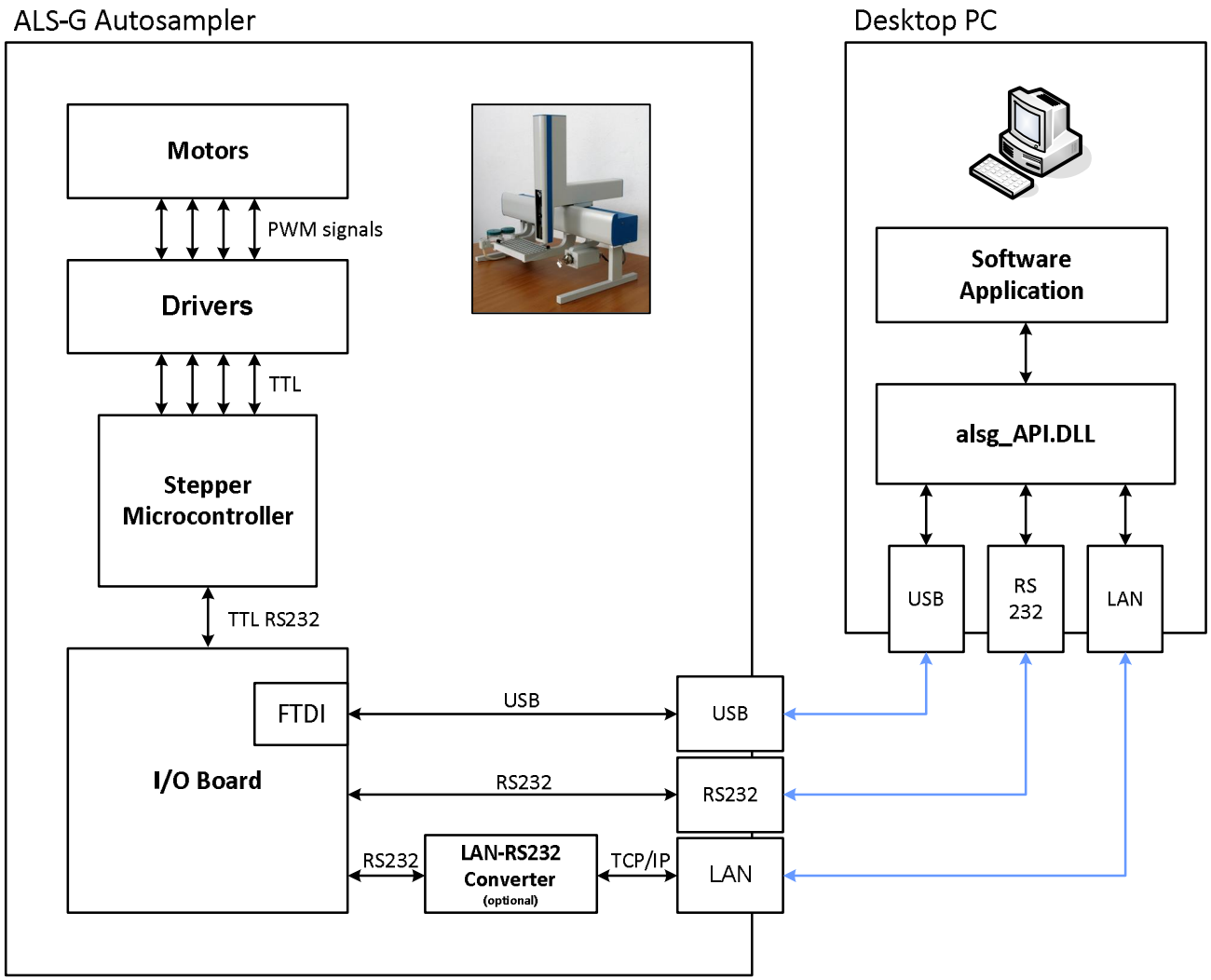

fig. 1: Overview of the ALS-G hardware structure.

2 alsg-API.dll

2.1 General

There are various categories in which the DLL interface exported functions can be subdivided. These categories are:

- Initialization — Functions that initialize the DLL
- Configuration — Functions that allow individual control on various configuration items
- Method — Functions that allow manipulation of the current instrument method
- Immediate — Functions that start immediate activity on the instrument

2.2 Initialization Exported Functions

The following functions are used for initialization of the DLL.

2.2.1 algConnect()

Prototype:

```
INT16 alsgConnect(INT16 nComPort)
```

Description:

Initializes the DLL communication threads, opens COM Port and tries to establish connection to the instrument. The COM Port and Baudrate are specified in the Parameter.TD file.

Parameters:

`INT16 nComPort`: COM Port to be used, (USB and LAN using virtual COM Port)
Note: when nComPort=0 is specified, the routine searches all present COM Ports for connected instrument and returns the COM Port to which instrument is connected

Return Value:

Used `ComPort` when successful, `0` when unsuccessful (i.e. Instrument not connected to COM port). When `0` is returned error cause can be retrieved using `alsgGetError()`.

Note:

For backward compatibility reasons the code using this function should look like this (presuming `FALSE = 0`)

```
nPort = 3;    // COM3
if (alsgConnect(nPort) == 0)
{
      MessageBox("Connection Error!");
}
```

2.2.2 alsgDisConnect()

Prototype:

```
BOOL alsgDisConnect(void)
```

Description:

Un-initializes the DLL communication threads, closes COM Port.

Parameters:

-

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.2.3 alsgReadInit()

Prototype:

```
BOOL alsgReadInit(char* szFileName)
```

Description:

Reads configuration and installation INI file and loads various configuration and calibration variables.

Parameters:

`char* szFileName` Filename (usually parameter.td) in which configuration data is stored.

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

Note:

When this function is not called previously, most other DLL functions will not operate correctly.

2.2.4 alsgSetDataPath()

Prototype:

```
BOOL alsgSetDataPath(char* szFileName);
```

Description:

Sets the file path for the ALSG_API.dll library for all read and write file operations. For example the parameter.td file, default.td, sequence.td, method.td files.

Parameters:

`char* szFileName` File path for all read and write file operations

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.2.5 alsgRunInit()

Prototype:

```
BOOL alsgReadInit(INT16 nWashStation)
```

Description:

Carries out a number of sampler initialization steps that are normally automatically carried out after DLL initialization before the first run is executed using the `RunMethod()` function. This includes dumping any possible syringe contents into the waste position of the given wash station and positioning the sampler in the home position.

Parameters:

`INT16 nWashStation`: 0 = 1st Wash Station
1 = 2nd Wash Station

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.3 Installation Exported Functions

The following functions are used to modify the installation related variables inside the DLL. These function act as an alternative of reading the Configuration/Installation file using

2.3.1 alsgSetInstallTrayPos()

Prototype:

```
BOOL alsgSetInstallTrayPos(INT16 nTray, double dxPos, double dyPos, double dzPos)
```

Description:

Set the reference point of the specified tray in x,y,z coordinates (mm). The zPos is used to specify the height of the vial for missing vial detection purposes.

Parameters:

`INT16 nTray`: 0 = 1st Tray
1 = 2nd Tray
2 = 3nd Tray
3 = 4nd Tray

`double dxPos`: x-position in mm of tray reference point (vial #1)

`double dyPos`: y-position in mm of tray reference point

`double dzPos`: z-position in mm of tray reference point (vial top)

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.3.2 alsgSetInstallTraySampleDepth()

Prototype:

```
BOOL alsgSetInstallTraySampleDepth(INT16 nTray, double dzDepth)
```

Description:

Set the depth inside the vial at which the sample is collected by the syringe needle.

Parameters:

`INT16 nTray`: 0 = 1st Tray
1 = 2nd Tray
2 = 3nd Tray
3 = 4nd Tray

`double dzDepth`: z-position in mm at which sample is collected by the syringe needle

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.3.3 alsgSetInstallWashStationPos()

Prototype:

```
BOOL alsgSetInstallWashStationPos(INT16 nWashStation, double dxPos, double dyPos)
```

Description:

Set the reference point of the given washstation in x,y coordinates (mm). The z-coordinate(s) of the bottles are set individually using other functions.

Parameters:

`INT16 nWashStation`: 0 = 1st Wash Station
1 = 2nd Wash Station

`double dxPos`: x-position in mm of tray reference point (most inner vial)

`double dyPos`: y-position in mm of tray reference point

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.3.4 alsgSetInstallWashStationSolventDepth()

Prototype:

```
BOOL alsgSetInstallWashStationSolventDepth(INT16 nWashStation, double dzDepth)
```

Description:

Set the depth of the Solvent bottle(s) of the given wash station in z coordinate (mm).

Parameters:

`INT16 nWashStation`: 0 = 1st Wash Station
1 = 2nd Wash Station

`double dzPos`: z-position in mm of solvent bottle top

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.3.5 alsgSetInstallWashStationWasteDepth()

Prototype:

```
BOOL alsgSetInstallWashStationWasteDepth(INT16 nWashStation, double dzDepth)
```

Description:

Set the depth of the Waste bottle of the given wash station in z coordinate (mm).

Parameters:

`INT16 nWashStation`: 0 = 1st Wash Station
1 = 2nd Wash Station

`double dzPos`: z-position in mm of waste bottle top

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.3.6 alsgSetInstallWashStationISTDDepth()

Prototype:

```
BOOL alsgSetInstallWashStationISTDDepth(INT16 nWashStation, double dzDepth)
```

Description:

Set the depth of the ISTD bottle of the given wash station in z coordinate (mm).

Parameters:

`INT16 nWashStation`: 0 = 1st Wash Station
1 = 2nd Wash Station

`double dzPos`: z-position in mm of ISTD bottle top

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.3.7 alsgSetInstallMotorSpeed()

Prototype:

```
BOOL alsgSetInstallMotorSpeed(INT16 nxSpeed, INT16 nySpeed)
```

Description:

Set the speed of the X and Y Motor to specific value in mm/s.

Parameters:

`INT16 nxSpeed`: X Motor speed [60é 260 mm/s]

`INT16 nySpeed`: Y Motor speed [60é 180 mm/s]

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.3.8 alsgSetInstallInjPointPos()

Prototype:

```
BOOL alsgSetInstallInjPointPos(INT16 nInjPoint, double dxPos, double dyPos, double dzPos)
```

Description:

Set the reference point for the specified injection position in mm.

Parameters:

`INT16 nInjPoint`: Injection point [0é 3]

`double dxPos`: x-position in mm of injection point reference point

`double dyPos`: y-position in mm of injection point reference point

`double dzPos`: x-position in mm of injection point reference point

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.3.9 alsgSetInstallSyringeExchangePos()

Prototype:

```
BOOL alsgSetInstallSyringeExchangePos(double dxPos, double dyPos, double dzPos)
```

Description:

Set the xyz position where the sampler head will move to as part of the syringe exchange procedure

Parameters:

`double dxPos`: x-position in mm of syringe exchange position

`double dyPos`: y-position in mm of syringe exchange position

`double dzPos`: x-position in mm of syringe exchange position

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.3.10 alsgInstallTeach()

Prototype:

```
BOOL alsgInstallTeach(HANDLE hParent, int nTeachItem, BOOL bManual, double *pdRsltX, double *pdRsltY, double * pdRsltX)
```

Description:

Displays the Teach UI and interacts with the user

Parameters:

`HANDLE hParent`: Window handle of the parent application, if any

`int nTeachItem`: Item to be taught

- 0..3: Injection Point 1 thru 4
- 4: Reserved
- 5..6: Tray Adjust 1 thru 2
- 7..8: Sample Depth for Tray 1 thru 2 (`bManual` must be `FALSE`)
- 9: Wash Station 0
- 10: Solvent Depth of Wash Station 0 (`bManual` must be `TRUE`)
- 11: Waste Depth of Wash Station 0 (`bManual` must be `TRUE`)
- 12: ISTD Depth of Wash Station 0 (`bManual` must be `TRUE`)
- 13: Wash Station 1
- 14: Solvent Depth of Wash Station 1 (`bManual` must be `TRUE`)
- 15: Waste Depth of Wash Station 1 (`bManual` must be `TRUE`)
- 16: ISTD Depth of Wash Station 1 (`bManual` must be `TRUE`)
- 17..18. Tray Adjust 3 thru 4
- 19..20. Sample Depth for Tray 3 thru 4 (`bManual` must be `FALSE`)

`BOOL bManual`: `TRUE` - Manual, `FALSE` - Teach

`double *pdRsltX`: pointer to result double value representing the x-axis position

When `nTeachItem` is 0 to 3, the X Position of Injection Point is stored.

When `nTeachItem` is 5 to 6, the X Position of Tray 0/1 is stored.

When `nTeachItem` is 7 to 8, not used

When `nTeachItem` is 9, the X Position of Wash Station 0 is stored.

When `nTeachItem` is 10-12, not used.

When `nTeachItem` is 13, the X Position of Wash Station 1 is stored.

When `nTeachItem` is 14-16, not used.

When `nTeachItem` is 17 to 18, the X Position of Tray 2/3 is stored.

When `nTeachItem` is 19-20, not used.

`double *pdRsltY`: pointer to result double value representing the y-axis position

When `nTeachItem` is 0 to 3, the Y Position of Injection Point is stored.

When `nTeachItem` is 5 to 6, the Y Position of Tray 0/1 is stored.

When `nTeachItem` is 7 to 8, not used

When `nTeachItem` is 9, the Y Position of Wash station 0 is stored.

When `nTeachItem` is 10 to 12, not used.

When `nTeachItem` is 13, the Y Position of Wash station 1 is stored.

When `nTeachItem` is 14-16, not used.

When `nTeachItem` is 17 to 18, the Y Position of Tray 2/3 is stored.

When `nTeachItem` is 19-20, not used.

`double *pdRsltZ`: pointer to result double value representing the z-axis position

When `nTeachItem` is 0 to 3, the z-Position of Injection Point is stored.

When `nTeachItem` is 5 to 6, the z-Position of Tray 0/1 is stored.

When `nTeachItem` is 7 to 8, the Sample Depth of Tray 0/1 is stored.

When `nTeachItem` is 9, not used.

When `nTeachItem` is 10, the Solvent Depth (z) of Wash Station 0 is stored

When `nTeachItem` is 11, the Waste Depth (z) of Wash Station 0 is stored

When `nTeachItem` is 12, the ISTD Depth (z) of Wash Station 0 is stored

When `nTeachItem` is 13, not used.

When `nTeachItem` is 14, the Solvent Depth (z) of Wash Station 1 is stored

When `nTeachItem` is 15, the Waste Depth (z) of Wash Station 1 is stored

When `nTeachItem` is 16, the ISTD Depth (z) of Wash Station 1 is stored

When `nTeachItem` is 17 to 18, the z-Position of Tray 2/3 is stored.

When `nTeachItem` is 19 to 20, the Sample Depth of Tray 2/3 is stored.

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.4 Configuration Exported Functions

The following functions are used to modify the configuration related variables inside the DLL.

2.4.1 alsgSetConfigMode()

Prototype:

```
BOOL alsgSetConfigMode(INT16 nMode)
```

Description:

Set the various standard operating modes of the sampler.

Parameters:

`INT16 nMode`: 0 = GC
1 = HPLC

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.4.2 alsgSetConfigStartSignal()

Prototype:

```
BOOL alsgSetConfigStartSignal(INT16 nContact)
```

Description:

Set the Start Signal behavior.

Parameters:

`INT16 nContact`: 0 = Ready Contact
1 = Remote

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful. When `FALSE` is returned error cause can be retrieved using `GetError()`.

2.4.3 alsgSetConfigReadyContact()

Prototype:

```
BOOL alsgSetConfigReadyContact(INT16 nLogic)
```

Description:

Set the Ready Contact logic.

Parameters:

`INT16 nLogic`: 0 = Normally open
1 = Normally closed

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.4.4 alsgSetConfigMomentStartOut()

Prototype:

```
BOOL alsgSetConfigMomentStartOut (INT16 aMomentStartOut)
```

Description:

Set the moment the start out pulse during injection will take place

Parameters:

`INT16 aMomentStartOut`: 0 = Start Out pulse after Plunger down
1 = Start Out pulse at start of Plunger going down

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.4.5 alsgSetConfigWaitPos()

Prototype:

```
BOOL alsgSetConfigWaitPos(INT16 nWaitPos)
```

Description:

Set the wait position of the tower of the instrument when idle.

Parameters:

`INT16 nWaitPos`: 0 = Left
1 = Right

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.4.6 alsgSetConfigTray()

Prototype:

```
BOOL alsgSetConfigTray(INT16 nTray, INT16 nTrayTyp)
```

Description:

Set the preconfigured tray type.

Parameters:

`INT16 nTray`: 0 = 1st Tray
1 = 2nd Tray
2 = 3nd Tray
3 = 4nd Tray

`INT16 nTrayTyp`: 0 = not installed
1 = 7 x15
2 = 4 x 8
3 = 10 x20

4 = 15 x 7
5 = 8 x 4

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.4.7 alsgSetConfigTrayMetrics()

Prototype:

```
BOOL alsgSetConfigTrayMetrics(INT16 nTray, INT16 nVials, INT16 nCols, INT16 nRows, double dColDist, double dRowDist)
```

Description:

Set the characteristics of user defined tray. This function is used when a predefined tray type is not used and set with `alsgSetTrayType()`.

Parameters:

`INT16 nTray`: 0 = 1st Tray
1 = 2nd Tray
2 = 3nd Tray
3 = 4nd Tray

`INT16 nVials`: Number of vials (usually same as `nCols*nRows`)

`INT16 nCols`: Number of tray columns (y-dir)

`INT16 nRows`: Number of tray rows(x-dir)

`double dColDist`: Distance in mm between columns

`double dRowDist`: Distance in mm between rows

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.4.8 alsgSetConfigWashStation()

Prototype:

```
BOOL alsgSetConfigWashStation(INT16 nWashStation, INT16 nType)
```

Description:

Set wash station type for the given Wash Station

Parameters:

`INT16 nWashStation`: 0 = 1st Wash Station
1 = 2nd Wash Station

`INT16 nType`: 0 = Solvent, Waste
1 = Solvent1, Waste1, Solvent2, Waste2, ISTD
2 = Solvent1, Solvent2, Solvent3, Waste, ISTD

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.4.9 alsgSetConfigActiveWashStation()

Prototype:

```
BOOL alsgSetConfigActiveWashStation (INT16 nWashStation)
```

Description:

Sets the active Wash Station. Although there can be two Wash Stations calibrated, there can only be one active at the time. All API functions using a Wash Station will be directed to the configured Active Wash Station unless these function have WashStation argument.

Parameters:

`INT16 nWashStation`: 0 = 1st Wash Station
1 = 2nd Wash Station

`INT16 nType`: 0 = Solvent, Waste
1 = Solvent1, Waste1, Solvent2, Waste2, ISTD
2 = Solvent1, Solvent2, Solvent3, Waste, ISTD

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.4.10 alsgSetConfigSyringeVol()

Prototype:

```
BOOL alsgSetConfigSyringeVol(INT16 nSyrVol)
```

Description:

Set the volume of the syringe used.

Parameters:

`INT16 nSyrVol`: Syringe size code [0é 4]
0 = 2µl, 1 = 5µl, 2 = 10µl, 3 = 25µl, 4 = 100µl

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.4.11 alsgSetConfigSyringeSpeed()

Prototype:

```
BOOL alsgSetConfigSyringeSpeed(INT16 nzUpSpeed, INT16 nzDownSpeed)
```

Description:

Set speed of syringe for z-Axis movements.

Parameters:

`INT16 nzUpSpeed`: z-Axis speed moving up [70é 130 mm/s]

`INT16 nzDownSpeed`: z-Axis speed moving down [70é 180 mm/s]

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.4.12 alsgSetConfigStartCycleBuzzer()

Prototype:

```
BOOL alsgSetConfigStartCycleBuzzer(INT16 nBuzzer)
```

Description:

Set buzzer alarm signal at start of run.

Parameters:

`INT16 nBuzzer`: 0 = Off
1 = On

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.4.13 alsgDetermineAutoWidth()

Prototype:

BOOL alsgDetermineAutoWidth `(INT16 nAxis)`

Description:

The auto sampler determines its x-Axis width by moving the tower over the x-axis from left sensor to right sensor and back. To read the actual measured x-axis use the alsgGetParam(0) function.

Parameters:

`INT16 nAxis`: 0 = x-Axis

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.4.14 alsgDetermineMagnetSpace()

Prototype:

```
BOOL alsgDetermineMagnetSpace (void)
```

Description:

The auto sampler determines its free space above the magnet on the z-Axis.

To read the actual measured free space use the alsgGetParam(2) function.

Parameters:

-

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.4.15 alsgGetParam()

Prototype:

double alsgGetParam `(INT16 nParameter)`

Description:

To read the specified parameter

Parameters:

```
INT16 nParameter:   0 = Determined auto width of the x-Axis
                    1 = Stored free space above the magnet of the z-Axis
                    2 = Determined free space above the magnet of the z-Axis
                    3 = Special Motion z-Axis
                    4 = Number of Errors appeared on a reference run on the z-Axis.
```

Return Value:

double format value of the requested parameter

2.4.16 alsgSetParam()

Prototype:

BOOL alsgDetermineAutoWidth `(INT16 nParameter, double dSetValue)`

Description:

Sets the value of the given parameter.

Parameters:

```
INT16 nParameter:   1 = Stored free space above the magnet of the z-Axis
                    3 = Special Motion z-Axis

double dSetValue:   When nParameter=1, free space in mm
                    When nParameter=3, 0=special motion off, 1=on
```

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.4.17 alsgGetDllVersion()

Prototype:

double alsgGetDllVersion `(void)`

Description:

The software version of the ALSG_API.dll

Parameters:

-

Return Value:

A double format value containing the software version

2.5 Method Exported Functions

The following functions are used to specify the settings under which the sampler will carry out its analytical run. Typically the method functions are related to settings that may vary from run to run.

2.5.1 alsgSetMethodInjPointNo()

Prototype:

```
BOOL alsgSetMethodInjPointNo(INT16 nInjPoint)
```

Description:

Defines in which injection point the sampler will inject. Injection positions need to be taught during installation of the instrument.

Parameters:

`INT16 nInjPoint`: Injection point reference [0é 3]

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful. When `FALSE` is returned error cause can be retrieved using `GetError()`.

2.5.2 alsgSetMethodInjDepth()

Prototype:

```
BOOL alsgSetMethodInjDepth(double dDepth)
```

Description:

Defines what additional depth the needle (z-axis) needs to move down when the sample is injected into the injection point. The default value for this depth is zero, assuming that the actual depth is determined by teaching the instrument. When a variable depth is required that can be set in the method, normally the needle is not inserted into the injector when teaching the instrument. When this is done, great care must be taken that the instrument is taught exactly above the injection point, otherwise the syringe may be damaged.

Parameters:

`double dDepth`: Injection depth

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.5.3 alsgSetMethodSimpleInjection()

Prototype:

```
BOOL alsgSetMethodSimpleInjection(double dSampleVolume, INT16 nPumps, double dAirGapVolume)
```

Description:

Defines the various settings for the GC simple injection mode.

Parameters:

`double dSampleVolume`: Sample Volume (µl)

`INT16 nPumps`: Number of Sample Pumps

`double dAirGapVolume`: Airgap Volume (µl)

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.5.4 alsgSetMethodSandwichInjection()

Prototype:

```
BOOL alsgSetMethodSandwichInjection(INT16 nSolvent, double dSolventVolume, double
dAirGapVolumePre, double dSampleVolume, double dAirGapVolumePost)
```

Description:

Defines the various settings for the sandwich injection mode.

Parameters:

`INT16 nSolvent`: Solvent used

`double dSolventVolume`: Solvent Volume (µl)

`double dAirGapVolumePre`: Airgap Volume before sample (µl)

`double dSampleVolume`: Sample Volume (µl)

`double dAirGapVolumePost`: Airgap Volume after sample (µl)

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.5.5 alsgSetMethodISTDInjection()

Prototype:

```
BOOL alsgSetMethodISTDInjection(double dAirGapVolumePre, double dISTDVolume, double dAirGapVolumePost, double dSampleVolume, double dAirGapVolume)
```

Description:

Defines the various settings for the Internal Standard injection mode. This mode is only valid when in GC Mode.

Parameters:

`double AirGapVolumePre`: Airgap Volume before sample (µl)

`double dISTDVolume`: Internal Standard Volume (µl)

`double dAirGapVolumePost`: Airgap Volume after sample (µl)

`double dSampleVolume`: Sample Volume (µl)

`double dAirGapVolume`: Airgap Volume (µl)

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.5.6 alsgSetMethodPreInjWashes()

Prototype:

```
BOOL alsgSetMethodPreInjWashes(INT16 nSolvent1, INT16 nSolvent2, INT16 nSolvent3, INT16 nSample)
```

Description:

Defines the number of clean strokes before injection.

Parameters:

`INT16 nSolvent1`: Number of Solvent1 washes

`INT16 nSolvent2`: Number of Solvent2 washes

`INT16 nSolvent3`: Number of Solvent3 washes

`INT16 nSample`: Number of Sample washes

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.5.7 alsgSetMethodPostInjWashes()

Prototype:

```
BOOL alsgSetMethodPostInjWashes(INT16 nSolvent1, INT16 nSolvent2, INT16 nSolvent3)
```

Description:

Defines the number of washes after injection.

Parameters:

`INT16 nSolvent1`: Number of Solvent1 washes

`INT16 nSolvent2`: Number of Solvent2 washes

`INT16 nSolvent3`: Number of Solvent3 washes

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.5.8 alsgSetMethodSampleWashVol()

Prototype:

```
BOOL alsgSetMethodSampleWashVol(double dWashVol)
```

Description:

Defines the injection volume.

Parameters:

`double dWashVol`: Sample wash volume

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.5.9 alsgSetMethodPreInjWashVol()

When this function is not called, default value of previous API versions is used (to safeguard upwards compatibility).

Prototype:

`BOOL` alsgSetMethodPreInjWashVol`(double dWashVolSolvent1, double dWashVolSolvent2, double dWashVolSolvent3)`

Description:

Defines the injection volume.

Parameters:

`double dWashVolSolvent1`: Solvent 1 wash volume

`double dWashVolSolvent2`: Solvent 2 wash volume

`double dWashVolSolvent3`: Solvent 3 wash volume

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.5.10 alsgSetMethodPostInjWashVol()

When this function is not called, default value of previous API versions is used (to safeguard upwards compatibility).

Prototype:

`BOOL` alsgSetMethodPostInjWashVol`(double dWashVolSolvent1, double dWashVolSolvent2, double dWashVolSolvent3)`

Description:

Defines the injection volume.

Parameters:

`double dWashVolSolvent1`: Solvent 1 wash volume

`double dWashVolSolvent2`: Solvent 2 wash volume

`double dWashVolSolvent3`: Solvent 3 wash volume

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.5.11 alsgSetMethodValveWashes()

Prototype:

`BOOL alsgSetMethodValveWashes(INT16 nSolvent1, INT16 nSolvent2)`

Description:

Defines the number of washes for cleaning the injection valve (HPLC).

Parameters:

`INT16 nSolvent1`: Number of Solvent1 washes

`INT16 nSolvent2`: Number of Solvent2 washes

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

Note:

This function is only valid in HPLC mode. As a result the number of solvents that can be specified in limited to two.

2.5.12 alsgSetMethodSpeed()

Prototype:

```
BOOL alsgSetMethodSpeed(double dSolventDraw, double dSampleDraw, double dInj, double dSolventDisp, double dSampleDisp, INT16 nSyrInsert)
```

Description:

Defines the speeds for various phases of the sample handling in the syringe.

Parameters:

`double dSolventDraw`: Solvent draw speed (µl/s)

`double dSampleDraw`: Sample draw speed (µl/s)

`double dInj`: Injection speed (µl/s)

`double dSolventDisp`: Solvent dispense speed (µl/s)

`double dSampleDisp`: Sample dispense speed (µl/s)

`INT16 nSyrInsert`: Syringe insertion speed (mm/s) (HPLC Mode only)

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.5.13 alsgSetMethodSyringeRetractionSpeed()

Prototype:

```
BOOL alsgSetMethodSyringeRetractionSpeed (INT16 nSyrSpeed)
```

Description:

Defines the speeds for the syringe when it leaves the injector

Parameters:

`INT16 nSyrSpeed`: Syringe retraction speed (mm/s)

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.5.14 alsgSetMethodDelay()

Prototype:

```
BOOL alsgSetMethodDelay(double dPreInjDelay, double dPostInjDelay, INT16 nViscDelay, INT16 nSolvDelay)
```

Description:

Defines the delay values used when taking up the sample from the vial and during injection.

Parameters:

`double dPreInjDelay`: Delay before injection (s)

`double dPostInjDelay`: Delay after injection (s)

`INT16 nViscDelay`: Viscosity delay (s)

`INT16 nSolvDelay`: Solvent delay (s)

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.6 Immediate Exported Functions

The following functions are used to issue commands to the sampler that are executed immediately.

2.6.1 alsgRunMethod()

Prototype:

```
BOOL alsgRunMethod(INT16 nTray, INT16 nVial, INT16 nReps)
```

Description:

Runs an analytical cycle based on Method values that were set previously. This function will return immediate but the execution may take considerable time during which the status of the instrument can be queried with `alsgGetStatus()`.

Parameters:

`INT16 nTray`:	0 = 1st Tray 1 = 2nd Tray
`INT16 nVial`:	Vial number to inject from
`INT16 nReps`:	Number of repetitions Note: when nReps=1 the sampler will wait for GCReady signal input

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.6.2 alsgRunSyringeWash()

Prototype:

```
BOOL alsgRunSyringeWash(INT16 nMode, INT16 nTray, INT16 nVial, INT16 nWashes)
```

Description:

Rinses the syringe with fluid based on value specified in `nMode`. This function will return immediate but the execution may take considerable time during which the status of the instrument can be queried with `alsgGetStatus()`.

Parameters:

`INT16 nMode`:	0 = Solvent1 1 = Solvent2 2 = Solvent3 (not for HPLC) 3 = Sample 4 = ISTD (not for HPLC)
`INT16 nTray`:	0 = 1st Tray (`nTray` only relevant when using `nMode` = 3) 1 = 2nd Tray
`INT16 nVial`:	Vial number to take rinse fluid (`nVial` only relevant when using `nMode` = 3)
`INT16 nWashes`:	Number of washes

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.6.3 alsgRunValveWash()

Prototype:

```
BOOL alsgRunValveWash(INT16 nMode, INT16 nWashes)
```

Description:

Washes the Injection Valve with fluid based on value specified in `nMode`. This function will return immediate but the execution may take considerable time during which the status of the instrument can be queried with `alsgGetStatus()`.

Parameters:

`INT16 nMode`:	0 = Solvent1 1 = Solvent2
`INT16 nWashes`:	Number of valve Washes

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

Note:

This command can only be run in HPLC mode.

2.6.4 alsgRunReference()

Prototype:

```
BOOL alsgRunReference(INT16 nAxis, INT16 nDirection)
```

Description:

Calibrates the specified axis by running past the specified photo interrupter in the specified direction.

Parameters:

`INT16 nAxis`:	0 = ALSG_X_AXIS	= x Axis
	1 = ALSG_Y_AXIS	= y Axis
	2 = ALSG_Z_AXIS	= z Axis
	3 = ALSG_U_AXIS	= u Axis
`INT16 nDirection`:	0 = ALSG_LEFT	= Left (x axis)
	1 = ALSG_RIGHT	= Right
	2 = ALSG_UP	= Up (z, u axes)
	4 = ALSG_IN	= In (y axis)

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.6.5 alsgRunAbort()

Prototype:

```
BOOL alsgRunAbort(void)
```

Description:

Aborts a RunMethod or Syringe or Valve Wash action.

Parameters:

-

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.6.6 alsgGetBusy()

Prototype:

```
BOOL alsgGetBusy(void)
```

Description:

Indicates whether one of the immediate commands is currently being executed.

Parameters:

-

Return Value:

`TRUE` when busy, `FALSE` when not busy.

2.6.7 alsgGetStatus()

Prototype:

```
INT16 alsgGetStatus(void)
```

Description:

Returns the current status of the instrument.

Parameters:

-

Return Value:

Status code as detailed in next section of the document.

2.6.8 alsgGetError()

Prototype:

```
INT16 alsgGetError(void)
```

Description:

Returns the current error of the instrument.

Parameters:

-

Return Value:

Error code as detailed in next section of the document.

2.6.9 alsgBeep()

Prototype:

```
INT16 alsgBeep(nReps)
```

Description:

Sounds the buzzer on the sampler `nReps`.times.

Parameters:

`INT16 nReps`: Number of times the buzzer signal is sounded with a short interval.

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.6.10 alsgSetIO()

Prototype:

```
BOOL alsgSetIO(INT16 nDev, INT16 nFunc, INT16 nVal)
```

Description:

Enables and sets the various I/O elements of the instrument. Some I/O elements have default functions. When API control is required for these elements, the element will have to be enabled before it can be cleared or set

Parameters:

`INT16 nDev`: Constant specifying I/O device
- 0 = `ALSG_FRONT_LED` = Front LEDs

`INT16 nFunc`: Constant specifying type of I/O operation
- 0 = `ALSG_IO_ENABLE` = Enable bits that are set in `nVal` for API use
- 1 = `ALSG_IO_CLR` = Clear bits that are set in `nVal`
- 2 = `ALSG_IO_SET` = Set bits that are set in `nVal`

`INT16 nVal`: Value used as parameter for operation (e.g. 0x00 in combination with `ALSG_IO_ENABLE` disables all API use of front LEDs)

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

Related constants:

```
// alsgSetIO devices
#define     ALSG_FRONT_LED    0

// alsgSetIO functions
#define     ALSG_IO_ENABLE    0
#define     ALSG_IO_CLR       1
#define     ALSG_IO_SET       2

// alsgSetIO value constants (bit pattern)
#define     ALSG_LED_RED      0x01
#define     ALSG_LED_GREEN    0x02

// alsgSetIO error constant (retrieved with alsgGetError() )
#define     ERROR_IO_NOT_ENABLED    84  // Attempt to update IO that cannot be set (should
                                        // be enabled first)
```

2.7 Diagnostic Exported Functions

The following functions are used to access diagnostics functions of the sampler. In this version of the DLL diagnostic information is maintained in a text based file.

2.7.1 alsgGetDiagPlungerStrokes()

Prototype:

```
INT32 alsgGetDiagPlungerStrokes(void)
```

Description:

Returns the number of plunger strokes since the last counter reset.

Parameters:

-

Return Value:

Number of plunger strokes. -1 signals diagnostics information cannot be retrieved.

2.7.2 alsgResetDiagPlungerStrokes()

Prototype:

`INT32` alsgResetDiagPlungerStrokes`(void)`

Description:

Resets the plunger counter.

Parameters:

-

Return Value:

Number of plunger strokes before reset. -1 signals diagnostics information cannot be retrieved.

2.7.3 alsgGetDiagInjections()

Prototype:

```
INT32 alsgGetDiagInjections(INT16 nInjPoint)
```

Description:

Returns the number of injections since last counter reset.

Parameters:

`INT16 nInjPoint`: -1 = All ports combined
Injection point [0é 3]

Return Value:

Number of injections. -1 signals diagnostics information cannot be retrieved.

2.7.4 alsgResetDiagInjections()

Prototype:

```
INT32 alsgResetDiagInjections(INT16 nInjPoint)
```

Description:

Resets the injection counter.

Parameters:

`INT16 nInjPoint`: -1 = All ports combined
Injection point [0é 3]

Return Value:

Number of injections before reset. -1 signals diagnostics information cannot be retrieved.

2.8 MicroStep Exported Functions

The following functions are used to interactively issue individual commands to control the instrument in more detail. This way actions can be programmed that go beyond the functionality of the pre-programmed method capabilities.

2.8.1 alsgStepGoToVial()

Prototype:

```
BOOL alsgStepGoToVial(INT16 nTray, INT16 nPos, double dRelDepth)
```

Description:

Moves the sampler to the specified vial position at the specified depth of the given Tray.

Parameters:

`INT16 nTray`:	0 = 1st Tray 1 = 2nd Tray 2 = 3nd Tray 3 = 4nd Tray
`INT16 nPos`:	Vial Position in Tray
`double dRelDepth`	Syringe Tip Depth in mm in relation to calibrated position, negative value indicating higher position

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

Note:

Calling this function consecutively with the same `nPos` value but difference `dRelDepth` values, the z-Axis can be moved relatively.

2.8.2 alsgStepGoToInject()

Prototype:

```
BOOL alsgStepGoToInject(INT16 nInjPoint, double dRelDepth)
```

Description:

Moves the sampler to the specified injection position at the specified depth.

Parameters:

`INT16 nInjPoint`:	Injection point [0é 3]
`double dRelDepth`	Syringe Tip Depth in mm in relation to calibrated position, negative value indicating higher position

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.8.3 alsgStepGoToSolvent()

Prototype:

```
BOOL alsgStepGoToSolvent(INT16 nWashStation, INT16 nPos, double dRelDepth)
```

Description:

Moves the sampler to the specified solvent position of the given Wash Station .

Parameters:

`INT16 nWashStation`:	0 = 1st Wash Station 1 = 2nd Wash Station
`INT16 nPos`:	0 = Solvent1 1 = Solvent2 2 = Solvent3 (not for HPLC)
`double dRelDepth`	Syringe Tip Depth in mm in relation to calibrated position, negative value indicating higher position

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.8.4 alsgStepGoToWaste()

Prototype:

```
BOOL alsgStepGoToWaste(INT16 nWashStation, INT16 nPos, double dRelDepth)
```

Description:

Moves the sampler to the specified waste position of the given Wash Station.

Parameters:

`INT16 nWashStation`:	0 = 1st Wash Station 1 = 2nd Wash Station
`INT16 nPos`:	0 = Waste1 1 = Waste2
`double dRelDepth`	Syringe Tip Depth in mm in relation to calibrated position, negative value indicating higher position

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.8.5 alsgStepGoToWait()

Prototype:

```
BOOL alsgStepGoToWait(INT16 nWashStation)
```

Description:

Move the sampler to the configured wait position of the given Wash Station.

Parameters:

`INT16 nWashStation`: 0 = 1st Wash Station
1 = 2nd Wash Station

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.8.6 alsgStepDraw()

Prototype:

```
BOOL alsgStepDraw(double dAmount, double dSpeed)
```

Description:

Draw the specified amount of fluid (or air) into the syringe at the specified speed.

Parameters:

-

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.8.7 alsgStepDispense()

Prototype:

```
BOOL alsgStepDispense(double dAmount, double dSpeed)
```

Description:

Dispense the specified amount of fluid (or air) from the syringe at the specified speed.

Parameters:

-

Return Value:

`TRUE` when successful, `FALSE` when unsuccessful.

2.8.8 alsgStepGotoSyrExchange()

Prototype:

```
double alsgStepGotoSyrExchange(INT16 nStep)
```

Description:

Move the instrument to the previously taught Syringe Exchange position. Since the Syringe Exchange procedure requires multiple steps, this phase of the procedure is specified in a parameter.

Parameters:

`INT16 nStep`:	0 = `SC_GOTO_POS`	= Goto position
	1 = `SC_U_NO_CURRENT`	= Switch off current U-axis stepper motor
	2 = `SC_GET`	= Determine syringe dead volume position
	3 = `SC_U_CURRENT`	= Switch on current U-axis stepper motor

Return Value:

-1.0 : Error

0.0 : OK for `nStep` = 0, 1, 3

> 0.0 Plunger null position for `nStep` = 2

2.9 System Functions

The DLL contains a few functions that are required for manufacturing service and other specials tasks. These functions are documented in a separate document.

2.9.1 alsgSysControl()

This API call gives access to a variety of various system calls. A complete listing of all available commands are documented separately.

Prototype:

```
Double alsgSysControl(INT16 nCmd, double dArg1, double dArg2, double dArg3, double dArg4, double dArg5, double dArg6, double dArg7, double dArg8)
```

2.9.1.1 Read Actual Axis Position

double alsgSysControl(nCmd, dArg1, dArg2, dArg3, 0, 0, 0, 0, 0)

Description:

Reads the firmware version of the instrument.

Parameters:

`INT16 nCmd`:	3 = `SYS_CTRL_CMD_GET_AXIS`
`double dArg1`:	0 = x-Axis
	1 = y-Axis
	2 = z-Axis
	3 = u-Axis, the plunger
`double dArg2`:	0 = default argument value to use, this will be documented separately.
`double dArg3`:	0 = When WAIT_POS is configured as left left
	1 = When WAIT_POS is right, the x-axis position will be mirrored

Return Value:

Firmware version of instrument.

2.9.1.2 Read Firmware Version

double alsgSysControl(nCmd, 0, 0, 0, 0, 0, 0, 0, 0)

Description:

Reads the firmware version of the instrument.

Parameters:

`INT16 nCmd`:	106 = `SYS_CTRL_CMD_GET_VERSION`

Return Value:

Firmware version of instrument.

2.9.1.3 Read Hardware Version

double alsgSysControl(nCmd, 0, 0, 0, 0, 0, 0, 0, 0)

Description:

Reads the hardware version of the instrument.

Parameters:

`INT16 nCmd`: 107 = `SYS_CTRL_CMD_GET_SUBVERSION`

Return Value:

Hardware version. Cast to a (char) value to display hardware version as a character, for instance 101=ëô

2.9.1.4 ReadSensor

double alsgSysControl(nCmd, dArg1, 0, 0, 0, 0, 0, 0, 0)

Description:

Reads the specified sensor state

Parameters:

`INT16 nCmd`: 108 = `SYS_CTRL_CMD_READ_SENSOR`

`double dArg1`: 0 = x-Axis sensor, will be left x-Axis sensor when WAIT_POS is left defined else the right x-axis sensor state is returned

1 = y-Axis sensor

2 = z-Axis sensor

3 = u-Axis sensor, the plunger sensor

4 = vial sensor

5 = x-axis left sensor

6 = x-axis right sensor

Return Value:

0: Sensor is not active, 1=sensor is active

2.10 Status Codes

0	`STATUS_NO`	No Error
1	`STATUS_ERROR`	Error occurred, call `GetError()` for details
2	`STATUS_ABORT`	Run was aborted
5	`STATUS_IDLE`	Instrument is not active
6	`STATUS_MOVE_TO_WAIT_POSITION`	Tower moves to home position
7	`STATUS_WAIT_FOR_GCREADY`	Instrument is ready to start run but waits for GC Ready signal
8	`STATUS_SEARCHING_DEVICE`	DLL is seaching for instrument on any existing COM Port
9	`STATUS_DETERMINE_AUTO_WIDTH`	Instrument is determining width of x-axis
10	`STATUS_DETERMINE_MAGNET_SPACE`	Instrument is determining free space above Magnet of z-Axis
11	`STATUS_GC_CLEAN`	Instrument is rinsing
12	`STATUS_GC_PRE_CLEAN`	Instrument is rinsing before injection
13	`STATUS_GC_PREPARE_INJECT`	Instrument prepares injection
14	`STATUS_GC_INJECT`	Instrument is injecting
15	`STATUS_GC_POST_CLEAN`	Instrument is rinsing after injection
21	`STATUS_HPLC_CLEAN`	Instrument is rinsing
22	`STATUS_HPLC_PRE_CLEAN`	Instrument is rinsing before injection
23	`STATUS_HPLC_PREPARE_INJECT`	Instrument prepares injection
24	`STATUS_HPLC_LOAD`	Instrument loads injection valve with sample
25	`STATUS_HPLC_INJECT`	Instrument activates injection valve
26	`STATUS_HPLC_EXTRACT`	Instrument extracts syringe from injection valve
27	`STATUS_HPLC_POST_CLEAN`	Instrument is rinsing after injection
28	`STATUS_HPLC_VALVE_CLEAN`	Instrument is rinsing injection valve

2.11 Error Codes

0	ERROR_NO	No Error
1	ERROR_PARAMFILE_NOT_FOUND	Parameter file not found
2	ERROR_OPEN_IFACE_FAILED	COM Port could not be opened
3	ERROR_NO_DEVICE_CONNECTED	No instrument found on any COM Port
4	ERROR_NOT_CALIBRATED_WIDTH_THIS_DLL	Parameter.td file created by another application version.
10	ERROR_MODE_RANGE	Specified Mode out of range
11	ERROR_INJ_POINT_RANGE	Specified Injection Point out of range
12	ERROR_START_SIGNAL_RANGE	Specified Start Signal out of range
13	ERROR_GCREADY_CONTACT_RANGE	Specified Contact Reference out of range
14	ERROR_WAIT_POSITION_RANGE	Specified Wait Position out of range
15	ERROR_TRAY_TYPE_RANGE	Specified Tray Type out of range
16	ERROR_WASH_STATION_TYPE_RANGE	Specified Wash Station Type out of range
17	ERROR_SYRINGE_VOLUME_RANGE	Syringe Volume out of range
18	ERROR_PLUNGER_ZEROPOS_RANGE	Plunger Zero Position out of range
19	ERROR_Z_UP_SPEED_RANGE	Z-Speed Up out of range
20	ERROR_Z_DOWN_SPEED_RANGE	Z-Speed Down out of range
21	ERROR_BUZZER_RANGE	Buzzer out of range
30	ERROR_X_SPEED_RANGE	Specified X-Speed out of range
31	ERROR_Y_SPEED_RANGE	Specified Y-Speed out of range
32	ERROR_INJ_POINT_POS_RANGE	Injection xyz Position out of range
32	ERROR_SYR_EXCH_POS_RANGE	Syringe Exchange xyz Position out of range
33	ERROR_SYRINGE_VOLUME_RANGE	Syringe Volume out of range
34	ERROR_TRAY_0_POSITION_RANGE	Tray 0 xyz Position out of range
35	ERROR_TRAY_1_POSITION_RANGE	Tray 1 xyz Position out of range
36	ERROR_TRAY_0_DEPTH_RANGE	Tray 0 depth out of range
37	ERROR_TRAY_1_DEPTH_RANGE	Tray 1 depth out of range
38	ERROR_WASH_STATION_POS_RANGE	Active Wash Station xyz Position out of range
39	ERROR_SOLVENT_DEPTH_RANGE	Solvent Depth out of range of Wash Station 0
40	ERROR_WASTE_DEPTH_RANGE	Waste Depth out of range of Wash Station 0
41	ERROR_ISTD_DEPTH_RANGE	ISTD Depth out of range of Wash Station 0

42	ERROR_TRAY_2_POS_RANGE	Tray 2 depth out of range
43	ERROR_TRAY_3_POS_RANGE	Tray 3 depth out of range
44	ERROR_SAMPLE_TRAY_2_DEPTH_RANGE	Tray 2 depth out of range
45	ERROR_SAMPLE_TRAY_2_DEPTH_RANGE	Tray 3 depth out of range
50	ERROR_INJ_MODE_RANGE	Injection Mode out of range
51	ERROR_INJ_TECHNIQUE_RANGE	Injection Technique out of range
52	ERROR_INJ_MODE_PARAM_RANGE	Injection Mode Parameter out of range
53	ERROR_INJ_VOLUME_RANGE	Injection Volume out of range
54	ERROR_WASH_SYRINGE_RANGE	Wash Syringe out of range
55	ERROR_VALVE_CLEAN_RANGE	Valve Clean out of range
56	ERROR_TRAY_RANGE	Tray out of range
57	ERROR_VIAL_RANGE	Vial number out of range
58	ERROR_SOLVENT_RANGE	Solvent out of range
59	ERROR_WASTE_RANGE	Waste out of range
60	ERROR_WASHES_RANGE	Washes out of range
61	ERROR_VOLUME_RANGE	Volume out of range
62	ERROR_DELAY_RANGE	Delay value out of range
63	ERROR_WASH_VOLUME_RANGE	Wash volume out of range
64	ERROR_INJ_DEPTH	Injection depth smaller than zero.
70	ERROR_CMD	
71	ERROR_NO_HPLC_FUNCTION	
72	ERROR_AXIS_RANGE	
73	ERROR_REF_STARTFREQ_RANGE	
74	ERROR_REF_ENDFREQ_RANGE	
75	ERROR_VEK_STARTFREQ_RANGE	
76	ERROR_VEK_ENDFREQ_RANGE	
77	ERROR_Z_SPEED_RANGE	
78	ERROR_U_SPEED_RANGE	
79	ERROR_HPLC_VALVE_POSITION_ERROR	
80	ERROR_VIAL_MISSING	No vial at sample position
81	ERROR_MISSING_STEP_GOTO_WAIT	
82	ERROR_Z_POS_RANGE	
83	ERROR_U_POS_RANGE	
84	ERROR_IO_NOT_ENABLED	IO Device bit(s)not enabled for control by SDK

References

[1] alsg_API.h

Specification Review History

Date	Reviewers
Findings	

Date	Reviewers
Findings	

Date	Reviewers
Findings	