Product Specification – ALS-G DLL API

For Product:	ALSG XYZ Sampler	For Revision:	V3Rev04C
Document Number:	aijw10035	Document Revision:	C
Author:	Jan Willem Mol	Date:	25 Mar 2011

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

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 **diagnostics** function calls (not yet documented in this version of the document)

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.

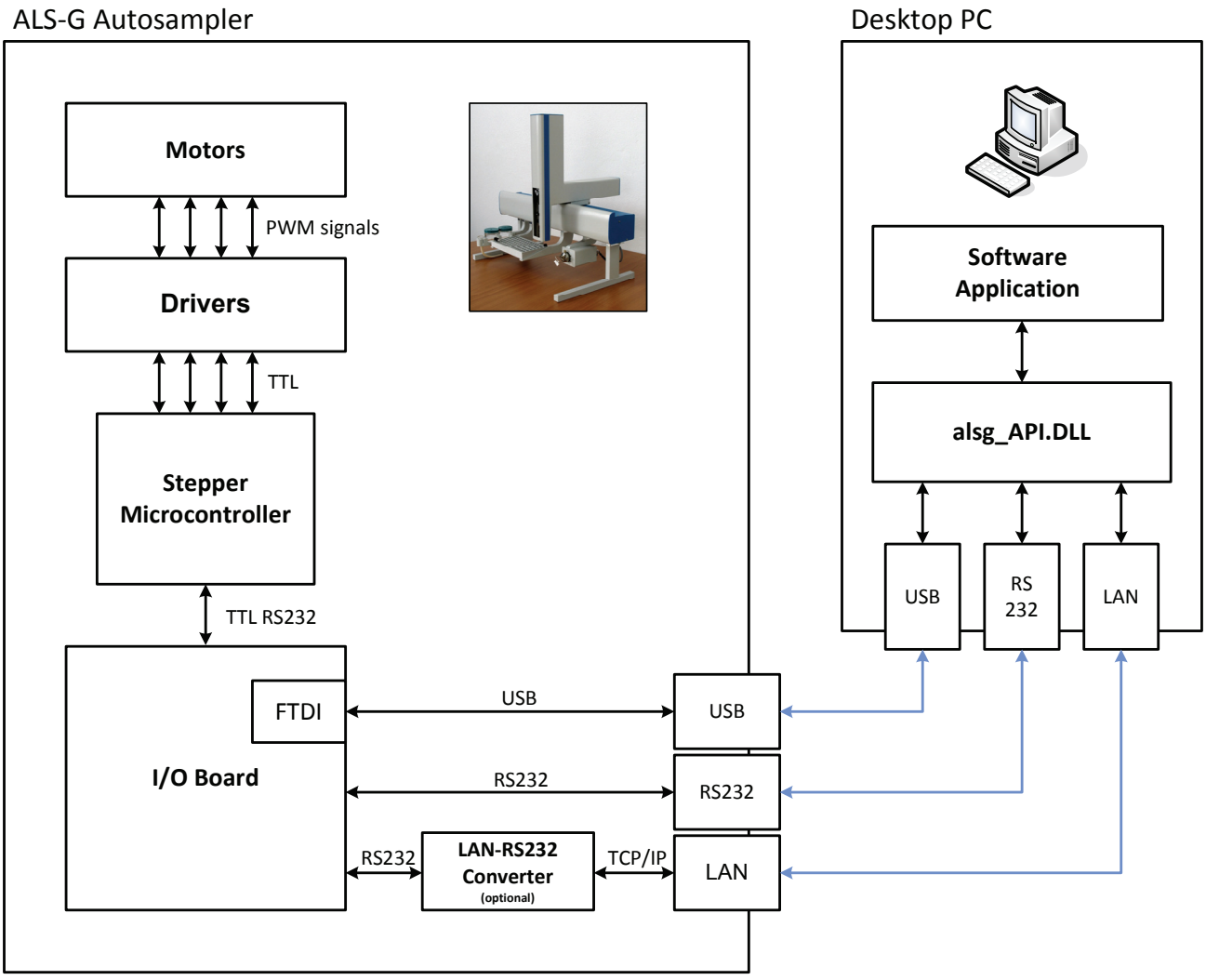

fig. 1: Overview of the ALS-G hardware structure.

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:

```
BOOL 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`: Comport to be used, (USB and LAN using virtual COM Port)

Return Value:

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

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.

2.3 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

`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:

```
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

`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 alsgSeInstallWashStationPos()

Prototype:

```
alsgSetInstallWashStationPos(double dxPos, double dyPos)
```

Description:

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

Parameters:

`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:

```
alsgSetInstallWashStationSolventDepth(double dzDepth)
```

Description:

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

Parameters:

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

Return Value:

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

2.3.5 alsgSetInstallWashStationWasteDepth()

Prototype:

```
alsgSetConfigWashStationWasteDepth(double dzDepth)
```

Description:

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

Parameters:

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

Return Value:

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

2.3.6 alsgSetInstallWashStationISTDDepth()

Prototype:

```
alsgSetInstallWashStationISTDDepth(double dzDepth)
```

Description:

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

Parameters:

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

Return Value:

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

2.3.7 alsgSetInstallMotorSpeed()

Prototype:

```
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 [60…260 mm/s]

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

Return Value:

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

2.3.8 alsgSetInstallInjPointPos()

Prototype:

```
alsgSetConfigInjPointPos(INT16 nInj, double dxPos, double dyPos, double dzPos)
```

Description:

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

Parameters:

`INT16 nInj`: Injection point [0…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.

2.4 Configuration Exported Functions

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

2.4.1 alsgSetConfigMode()

Prototype:

```
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 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.

2.4.5 alsgSetConfigTray()

Prototype:

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

Description:

Set the preconfigured tray type.

Parameters:

`INT16 nTray`: 0 = 1st Tray
1 = 2nd 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.

2.4.6 alsgSetConfigTrayMetrics()

Prototype:

```
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
`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.

2.4.7 alsgSetConfigWashStation()

Prototype:

```
alsgSetConfigWashStation(INT16 nType)
```

Description:

Set wash station type.

Parameters:

`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.

2.4.8 alsgSetConfigSyringeVol()

Prototype:

```
alsgSetConfigSyringeVol(INT16 nSyrVol)
```

Description:

Set the volume of the syringe used.

Parameters:

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

Return Value:

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

2.4.9 alsgSetConfigSyringeSpeed()

Prototype:

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

Description:

Set speed of syringe for z-Axis movements.

Parameters:

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

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

Return Value:

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

2.4.10 alsgSetConfigStartCycleBuzzer()

Prototype:

```
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.

2.5 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 [0…3]

Return Value:

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

2.5.2 alsgSetMethodSimpleInjection()

Prototype:

```
BOOL alsgSetMethodSimpleInjection(double dSampleVolume, INT16 nStrokes, 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.

2.5.3 alsgSetMethodSandwichInjection()

Prototype:

```
BOOL alsgSetMethodSandwichInjection(INT16 nSolvent, double dSolventVolume, double
AirGapVolumePre, 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 AirGapVolumePre`:	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.

2.5.4 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 AirGapVolumePost`:	Airgap Volume after sample (µl)
`double dSampleVolume`:	Sample Volume (µl)
`double dAirGapVolume`:	Airgap Volume (µl)

Return Value:

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

2.5.5 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.

2.5.6 alsgSetMethodPostInjWashes()

Prototype:

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

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

`INT16 nSample`: Number of Sample washes

Return Value:

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

2.5.7 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.

2.5.8 alsgSetMethodValveWashes()

Prototype:

```
BOOL alsgSetMethodValveWashCycles(INT16 nSolvent1, INT16 nSolvent2)
```

Description:

Defines the number of clean cycles for washing 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.

2.5.9 alsgSetMethodSpeed()

Prototype:

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

Description:

Defines the number of clean strokes after injection.

Parameters:

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

`double dFSampleDraw`: 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.

2.5.10 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 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 alsgRunAbort()

Prototype:

```
BOOL alsgRunAbort(void)
```

Description:

Aborts a RunMethod or Syringe or Valve Wash action.

Parameters:

-

Return Value:

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

2.6.5 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.

2.6.6 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.

2.7 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
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

2.8 Error Codes

0	ERROR_NO	No Error
1	PARAMETER_FILE_NOT_FOUND	Paramter file not found
2	UART_OUT_OF_RANGE	Illegal COM Port specified
3	OPENING_INTERFACE_FAILED	COM Port could not be opened
10	INJ_POINT_OUT_OF_RANGE	Specified Injection Point out of range
11	START_SIGNAL_OUT_OF_RANGE	Specified Start Signal out of range

12	GCREADY_CONTACT_OUT_OF_RANGE	Specified GC Ready contact out of range
13	WAIT_POSITION_OUT_OF_RANGE	Specified Wait Position out of range
14	TRAY_TYP_OUT_OF_RANGE	Specified Tray Type out of range
15	WASH_STATION_TYP_OUT_OF_RANGE	Specified Wash Station Type out of range
16	X_SPEED_OUT_OF_RANGE	Specified X-Speed out of range
17	Y_SPEED_OUT_OF_RANGE	Specified Y-Speed out of range
20	SYRINGE_VOLUME_OUT_OF_RANGE	Syringe Volume out of range
21	PLUNGER_ZEROPOS_OUT_OF_RANGE	Plunger Zero Position out of range
22	Z_UP_SPEED_OUT_OF_RANGE	Z-Speed Up out of range
23	Z_DOWN_SPEED_OUT_OF_RANGE	Z-Speed Down out of range
24	BUZZER_OUT_OF_RANGE	Buzzer out of range
25	INJ_POINT_POSITION_OUT_OF_RANGE	Injection xyz Position out of range
26	TRAY_I_POSITION_OUT_OF_RANGE	Tray 0 xyz Position out of range
27	TRAY_II_POSITION_OUT_OF_RANGE	Tray 1 xyz Position out of range
28	WASH_STATION_POSITION_OUT_OF_RANGE	Wash Station xyz Position out of range
30	INJ_MODE_OUT_OF_RANGE	Injection Mode out of range
31	INJ_MODE_PARAM_OUT_OF_RANGE	Injection Mode Parameter out of range
32	INJ_VOLUME_OUT_OF_RANGE	Injection Volume out of range
33	WASH_SYRINGE_OUT_OF_RANGE	Wash Syringe out of range
34	VALVE_CLEAN_OUT_OF_RANGE	Valve Clean out of range
35	TRAY_OUT_OF_RANGE	Tray out of range
36	VIAL_OUT_OF_RANGE	Vial number out of range
37	SOLVENT_OUT_OF_RANGE	Solvent out of range
38	WASTE_OUT_OF_RANGE	Waste out of range
39	STROKES_OUT_OF_RANGE	Strokes out of range
40	VOLUME_OUT_OF_RANGE	Volume out of range
41	DELAY_OUT_OF_RANGE	Delay value out of range
60	HPLC_VALVE_POSITION_ERROR	
61	NO_HPLC_FUNCTION	
62	NO_GC_FUNCTION	
63	VIAL_MISSING	No vial at sample position

70	AXIS_OUT_OF_RANGE	Axis value out of range
71	REF_STARTFREQ_OUT_OF_RANGE	Reference run start frequency out of range
72	REF_ENDFREQ_OUT_OF_RANGE	Reference run end frequency out of range
73	VEK_STARTFREQ_OUT_OF_RANGE	Vector run start frequency out of range
74	VEK_ENDFREQ_OUT_OF_RANGE	Vector run end frequency out of range
75	Z_SPEED_OUT_OF_RANGE	Z-Speed out of range
76	U_SPEED_OUT_OF_RANGE	U-Speed out of range

References

[1] alsg_v3rev04_dll.h

Specification Review History

Date	Reviewers
Findings	

Date	Reviewers
Findings	

Date	Reviewers
Findings	