SpectraSense

Software Developer's Kit

SpectraSense™
Acton Research

User's Guide

SpectraSense OLE COM Interface

The SpectraSense COM interface allows users to create software applications that interface with and control the software and hardware through Microsoft's COM/DCOM protocol. This interface is declared with GUID{3DC4AEF1-331E-11D4-9DBR-00A0C98BBDOA} and recognized by the system as ***SpectraSense.SpectraSense_Interface.*** The interface is self-registering on the invocation of SpectraSense.
The Interface is divided into three levels:
1. High level functions: Direct software control of SpectraSense and some visual interfaces.
2. Mid level functions: Manual control of scan parameters.
3. Low level functions: Direct control of attached hardware

The High level functions can be used by the programmer to load SpectraSense from within his program and initiate complete acquisition routines. The Mid level functions allow the user to create or change acquisition parameters from within their program. The Low level functions allow the user to control individual actions of the hardware such as turning a mirror, moving a monochromator to a specific position, or changing a grating.
A complete set of example programs written in Visual Basic, and Delphi are included on the SpectraSense installation disk.

Instrument Control and Data Acquisition Functions

High Level Functions

Scan	Acquire Data
LoadRoutine	Loads an acquisition routine and Map (if applicable)
getScanStorageName	Returns the name under which the data will be stored
LastSavedFileNames	Returns a list of the files saved from the last acquisition
AbortScan	Stops an acquisition
RoutineName	Returns the name of the current acquisition routine
getPageNum	Returns the current SpectraSense screen
SelectPage	Brings the specified SpectraSense page to the front
isSpectraSenseUp	Returns SpectraSense loading status
SpectraSenseToFront	Brings SpectraSense to top of desktop
SpectraSenseToBack	Sends SpectraSense to bottom position on desktop

Mid Level Functions

setCCDScanType	Sets spectral or peak monitoring acquisition mode
setCCDScanParams	Input of parameters necessary for an acquisition
setCCDMathParams	Input for real-time processing parameters
setNCLScanType	Sets spectral or time based acquisition mode
setNCLScanParams	Input of parameters for an acquisition
setNCLMathParams	Input for real-time processing parameters

Low Level Functions

SpectraSense

UpdateSpectraSense	Updates the values on the SpectraSense pages

Monochromator control functions

Mono_Present	Tests to see if a monochromator is in the system
MonoDouble	Defines a monochromator as a double monochromator
getMonoWavelength	Returns a monochromator position in specified units
setMonoWavelength	Sends the monochromator to a wavelength in specified units
getMonoGrating	Returns the grating number in use
setMonoGrating	Change the grating to the number specified
getMonoDiverterPosition	Returns the position of the active port
setMonoDiverterPosition	Change the diverter mirror position
MonoSlitMotorized	Tests to see if a slit is motorized or manual
getMonoSlitWidth	Returns the slit width of a motorized slit
setMonoSlitWidth	Set a motorized slit to a specific width

NCL control functions

NCL_Present	Tests to see if an NCL is present in the system
getNCLITime	Returns the current integration time
setNCLITime	Sets the integration time in **ms**
NCL_ChPresent	Tests to see if a specific channel is present in the configuration
getNCLChHVon	Returns the high voltage setting for a specified channel
setNCLChHVon	Sets the high voltage for a specific on or off
getNCLChHV	Returns the high voltage setting on a specific channel
setNCLChHV	Sets the high voltage to a value on a specific channel
NCLChRead	Read the output of a specific channel in specified count units

NCLChReadAll	Reads all of the I/O lines at one time
getifNCLInputLineAvail	Checks if the specific I/O line is available
acquireNCLInputLine	Assigns an input line to the COM interface
getNCLInputLine	Reads the State of an NCL input line
getifNCLOutputLineAvail	Checks if the specified input line is available
acquireNCLOutputLine	Assigns an output line to the COM interface
getNCLOutputLine	Reads the state of an NCL output line
setNCLOutputLine	Sets the state of an NCL output line

Shutter control functions

Shutter_Present	Test to see if a shutter is present
getNCLShutterPosition	Checks if shutter is open or closed
setNCLShutterPosition	Sets the shutter opened or closed

Filter wheel control functions

Filter_Present	Test to see if a filter wheel is present
getFilterPosition	Returns the current filter position (1 – 6)
setFilterPosition	Moves the specified filter into position
isFilterSetToAutoInsert	Test if automatic insertion off filters is active
setFilterAutoInsertFlag	Activates automated filter insertion. Note filter positions and insertion wavelength must be defined in Hardware Config.

CCD control functions

CCD_Present	Tests to see if a CCD is in the configuration
CCDInLightPath	Tests to see if the CCD is in the light path
getCCDITime	Returns the current CCD integration time
setCCDITime	Sets a new CCD integration time
CCDLoadMapFile	Loads a CCD area map file
CCDMapName	Returns the name of the current CCD area map file
getCCDInputByte	Returns the state of all 8 input lines on the ST-133 controller

getCCDInputLine	Returns the state of a specific input line on the controller
getCCDOutputByte	Returns the state of all 8 output lines on the St-133 controller
getCCDOutputLine	Returns the state of a specific output line on the controller
setCCDOutputByte	Sets the state of all output lines on the controller
setCCDOutputLine	Sets the state of a specific output line on the controller

Data Manipulation and Presentation Functions

The following functions are available through the ARC Data Viewer Active X component. All functions must be preceded by **ARCDataViewX1. _**. The component must be registered with the operating system before use. Instructions on registration follow the function descriptions.

Display Functions

arc_OpenFile	Opens a data file for display and manipulation
arc_CloseFile	Closes a data file
arc_SaveFile	Saves a data file
arc_DisplayChartToolbar	Shows the data viewer's tool bar
arc_ChartPreview	Shows how a spectrum will be presented on a printed page
arc_ChartEdit	Allows editing of chart formatting aspects
arc_ChartPrint	Prints out a spectrum chart
arc_ZoomAutoScale	Automatically scales the spectral intensity axis
arc_ZoomPercent	Expands the spectral display the specified percent
arc_ZoomX	Expands the X axis between two specified values
arc_ZoomY	Expands the Y axis between two specified values

Scalar Math Functions

arc_ScalarAddK	Adds a constant value, K, to each Y (intensity) value
arc_ScalarSubK	Subtracts a constant value,K, from each Y (intensity) value
arc_ScalarSubFromK	Subtracts each Y (intensity) value from a constant, K
arc_ScalarDivideByK	Divides each Y (intensity) value by a constant, K
arc_ScalarDivIntoK	Divides each Y (intensity) value into a constant, K
arc_ScalarMultK	Multiplier each Y (intensity) value by a constant, K
arc_ScalarRaiseK	Raises each Y (intensity) value by a constant, K
arc_Ln	Takes the natural log of each Y (intensity) value

arc_Log	Takes the Log base 10 of each Y (intensity) value
arc_AntiLn	Takes the antilog (base e) of each Y (intensity) value
arc_AntiLog	Takes the antilog (base 10) of each Y (intensity) value
arc_FirstDerivative	Takes the first derivative of the Y axis
arc_SecondDerivative	Takes the second derivative of the Y axis
arc_Xshift	Adds a constant to each X (wavelength) value

File Math Functions

arc_AddSpectrum	Adds a specified spectrum to the current spectrum
arc_SubSpectrum	Subtracts a specified spectrum from the current spectrum
arc_DivSpectrum	Divides the current spectrum by a specified spectrum
arc_MultSpectrum	Multiplies the current spectrum by a specified spectrum
arc_DataSmooth	Smooths data using the Savitsky-Golay algorithm
arc_JoinSpectrum	Combines the data from the current spectrum with a specified spectrum to create a new spectrum
arc_Trunc	Removes data from a spectrum outside of specified X axis limits
arc_CurveArea	Calculates the integrated intensity between two specified X axis values
arc_FindPeak	Locates X axis positions with intensities above a specified intensity criterion
arc_ChangeWaveUnit	Converts data presentation from one spectral unit value to another, i.e., nanometers to wavenumbers
arc_Linearize	This function is used after conversion from X value units to another. It will respace the X-axis by extrapolation. The X values are based upon the smallest X increment in the original data set. This function is typically used to convert from pixels to selected spectral working units and conversion between wavelenth and wavenumbers or eV.

Other Functions

arc_getY	Returns the Y value for any specified X value
arc_getXY	Produces an array of XY pairs

arc_getPeaks	Produces an array of XY pairs calculated to be peaks from the FindPeaks funtion
arc_DisplayPeaks	Labels the peaks on the graphical output
arc_DisplayComponents	When a files is created using either two data channel inputs or is the result of a mathematical treatment of two data sets, the raw data sets are displayed with the resultant spectrum on the screen. The raw data is always stored in the original data file
arc_Recalculate	This function allows reconstruction real-time processed spectra from the raw components.
arc_getLog	This function retrieves the data file header with all of the stored parameters and commentary

Registering the ARCDataView ocx component

To use ARCDataView.ocx in your application you must do two things:

1. You must first register ARCDataView.ocx with the operating system.
2. You must install the control in your development environment.

To register ARCDataView.ocx must be opened with **regsvr32**. ARCDataView.ocx is located in the \SpectraSense\SpectraSense_SDK\DataView_SDK directory

Regsvr32.exe is located in the \Windows\System directory on Win95 systems.
Regsvr32.exe is located in the \Windows\System directory on Win98 systems.
Regsvr32.exe is located in the \WinNT\System32 directory on windows NT and Windows 2000 systems.
ARCDataView.ocx can be registered from the command line by typing:
prompt > {path}\regsvr32 {path}\ARCDataView.ocx
An alternative method is to work in Windows Explorer. Drag ARCDataView.ocx onto regsvr32.ocx. This will register the control.
The system should respond to either of these actions by telling you it succeeded.

To install the ActiveX component in Delphi follow the below proceedure:

1. Open Delphi
2. Go to the Menu item, File -> Close All
3. Go to the Menu item, Component -> Import ActiveX Control...
4. Select "ARCDataView Library"
5. Click the install button
6. Exit, then restart Delphi.

On the component tool bar under ActiveX there should now be a component labeled 'ARCDataViewX'.

To install the ActiveX component in VB6:

1. Open VB6
2. Go to the Menu item, Project -> Components (Ctrl-T)
3. Check ARCDataView Library
4. Click "OK"

The component is now on your tool bar, and can be used like any other component in VB.

AbortScan

This function will abort an acquisition that is in progress.

Description: Aborts an acquisition in progress.
Note: COM/DCOM commands do not return until completed. Therefore if this is to be used with Scan, it needs to be in a separate thread or program.

Parameters:

Definitions:

Delphi

Procedure AbortScan; safecall;

Microsoft IDL

[id(0x00000007)]
HRESULT AbortScan ( );

Visual Basic Sample Code

```
SpectraSense_COM.AbortScan
```

Delphi Sample Code

```
SpectraSenseServer.AbortScan;
```

Note : This command only works, if SpectraSense is not processing another COM command. This is a short coming in Microsoft COM/DCOM and not SpectraSense.

acquireNCLInputLine

Description: Assigns a line to the COM interface

Parameters:

Line_Num	Valid numbers (1-4)	
Setup_Bool	0	Return the line to SpectraSense
	1	Assign the line to the COM interface
Results	0	An invalid request
	!0	A valid request

Definitions:

Delphi

```
Function acquireNCLInputLine(      Line_Num: Integer;
                                   Setup_Bool: Integer
                              ): Integer; safecall;
```

Microsoft IDL

```
[id(0x0000002f)]
HRESULT acquireNCLOutputLine(
                                   [in] long Line_Num,
                                   [in] long Setup_Bool,
                                   [out, retval] long*
                                   ValidRequest);
```

Visual Basic Sample Code

```
If SpectraSense_COM.acquireNCLInputLine(1, 1) = 0 Then
   MsgBox (“Failed to Acquire Line 1”)
   End If
```

Delphi Sample Code

```
// allocate line 1 from spectrasense
if SpectraSenseServer.acquireNCLInputLine(1,1) = 0
   then showmessage(‘Failed to Acquire Line 1’);
```

acquireNCLOutputLine

Description: Assigns a line to the COM interface

Parameters:

Line_Num	Valid line numbers (1 – 4)	
Setup_Bool	0	Returns the line to SpectraSense
	!0	Assigns the line to the COM interface

Result:

0	An invalid request
!0	A valid request

Definitions:

Delphi

```
Function acquireNCLOutputLine(        Line_Num: Integer;
                                      Setup_Bool: Integer
                                    ): Integer; safecall;
```

Microsoft IDL

```
[id(0x0000002c)]
HRESULT acquireNCLInputLine(
                                      [in] long Line_Num,
                                      [in] long Setup_Bool,
                                      [out, retval] long*
                                       ValidRequest);
```

Visual Basic Code

```
‘ allocate line 1 from spectrasense
If SpectraSense_COM.acquireNCLOutputLine(1, 1) = 0 Then
   MsgBox (“Failed to Acquire Line 1”)
   End If
```

Delphi Sample Code

```
// allocate line 1 from spectrasense
if SpectraSenseServer.acquireNCLOutputLine(1,1) = 0
   then showmessage(‘Failed to Acquire Line 1’);
```

arc_AddSpectrum

Description:	This function will add the Y values from a specified data file to the currently open data file at equivalent X values
Parameters:	
FileName	The name of the data file to be added to the current file
Result	0 Failed to open data file !0 Data file opened

Definitions:

Delphi

function arc_AddSpectrum(FileName : OleVariant): Integer; safecall;

Microsoft IDL

[id(0x00000027)]
long arc_AddSpectrum([in] VARIANT FileName);

Visual Basic Sample Code

```
‘add stored data to open file
Dim FileName  As Variant
If ARCDataViewX1.arc_AddSpectrum(FileName) <> 0 Then
               MsgBox (“Added File”)
               Else
               MsgBox (“Failed to Add File”)
               End If
```

Delphi Sample Code

```
FileVar := newFileName;
if ARCDataViewX1.arc_AddSpectrum(FileVar) <> 0
       then showmessage(‘Added File : ‘ +ExtractFileName(newFileName))
       else showmessage(‘Failed to Add File : ‘ +ExtractFileName(newFileName));
```

arc_AntiLn

Description: This function will take the antilog (base e) of all Y components in the data file.

Parameters: None

Definitions:

Delphi

Procedure arc_AntiLn; safecall;

Microsoft IDL

[id(0x00000020)]
void arc_AntiLn();

Visual Basic Sample Code

```
'Take the AnitLn of the Y data
ARCDataViewX1.arc_AntiLn
```

Delphi Sample Code

```
ARCDataViewX1.arc_AntiLn
```

arc_AntiLog

Description: This function will take the antilog (base 10) of all Y components in the data file.

Parameters: None

Definitions:

Delphi

Procedure arc_AntiLog; safecall;

Microsoft IDL

[id(0x00000021)]
void arc_AntiLog();

Visual Basic Sample Code

```
‘Take the AntiLog of the Y data
ARCDataViewX1.arc_AntiLog
```

Delphi Sample Code

```
ARCDataViewX1.arc_AntiLog
```

arc_ChangeWaveUnit

Description: This function will convert the X axis units between micron, nm, A, eV, absolute wavenumbers and relative wavenumbers

Parameters:

NewUnits

0	none
1	nm
2	Angstroms
3	microns
4	absolute wavenumbers
5	relative wavenumbers
6	eV

RelWaveNMCenter For working in relative wavenumber spectral units the excitation wavelength in nanometers must be supplied. For Raman spectroscopy this is the laser wavelength.

Definitions:

Delphi

```
procedure arc_ChangeWaveUnit(newUnits: Integer;
          relWaveNMCenter: Double); safecall;
```

Microsoft IDL

```
[id(0x0000002f)]
void arc_ChangeWaveUnit(
[in] long newUnits,
[in] double RelWaveNMCenter);
```

Visual Basic Sample Code

```
Call ARCDataViewX1.arc_ChangeWaveUnit(WaveCombo.ListIndex, newCenter)
```

Delphi Sample Code

```
ARCDataViewX1.arc_ChangeWaveUnit(WaveCombo.itemindex,0.0);
```

arc_ChartEdit

Description: This function will allow editing of the chart characteristics

Definitions:

Delphi

Procedure arc_ChartEdit; safecall;

Microsoft IDL

[id(0x00000039)]
void arc_ChartEdit();

Visual Basic Sample Code

```
ARCDataViewX1.arc_ChartEdit
```

Delphi Sample Code

```
ARCDataViewX1.arc_ChartEdit;
```

arc_ChartPreview

Description: This function will display how a spectrum will be printed on a page

Definitions:

Delphi

Procedure arc_ChartPreview; safecall;

Microsoft IDL

[id(0x00000038)]
void arc_ChartPreview();

Visual Basic Sample Code

```
ARCDataViewX1.arc_ChartPreview
```

Delphi Sample Code

```
ARCDataViewX1.arc_ChartPreview;
```

arc_ChartPrint

Description: This function will print out a spectrum

Definitions:

Delphi

Procedure arc_ChartPrint; safecall;

Microsoft IDL

[id(0x0000003a)]
void arc_ChartPrint ();

Visual Basic Sample Code

```
ARCDataViewX1.arc_ChartPrint
```

Delphi Sample Code

```
ARCDataViewX1.arc_ChartPrint;
```

arc_CloseFile

Description: This function is used to close the current data file

Parameters:

Definitions:

Delphi procedure arc_CloseFile; safecall

Microsoft IDL [id(0x00000011)]
void arc_CloseFile();

Visual Basic Sample Code

```
ARCDataViewX1.arc_CloseFile
```

Delphi Sample Code

```
ARCDataViewX1.arc_CloseFile;
```

arc_CurveArea

Description:	This function calculates the integrated Y values between two specified X value boundaries

Parameters:

X1	The lower X value for the truncated data file
X2	The upper X value for the truncated data file
Total Area	Returns the total area between X1 and X2
Baseline Area	Returns the baseline area between X1 and X2

Definitions:

Delphi

procedure arc_CurveArea(X1: Double; X2: Double; **out** TotalArea: Double; out BaseLineArea: Double); safecall;

Microsoft IDL

[id(0x0000002d)]
void arc_CurveArea(
[in] double X1,
[in] double X2,
[out] double* TotalArea,
[out] double* BaseLineArea);

Visual Basic Sample Code

```
‘get the area under a peak
Dim LowLimit  As Double
Dim UpperLimit As Double
Dim Total Area As Double
Dim BaseLineArea As Double
ARCDataViewX1.arc_CurveArea(LowLimit, UpperLimit)
```

Delphi Sample Code

```
X1 := StrToFloat(Trim(SpectralX1Edt.text));
X2 := StrToFloat(Trim(SpectralX2Edt.text));
ARCDataViewX1.arc_Trunc(X1,X2);
```

arc_DataSmooth

Description:	This function will perform a Savitsky-Golay smoothing of the data
Parameters:	
PtSmooth	The smoothing values of 5, 9, 15. And 25 are valid inputs
Result:	0 Failed to open data file !0 Data file opened

Definitions:

Delphi

procedure arc_DataSmooth(PtSmooth : Integer);Integer; safecall;

Microsoft IDL

[id(0x00000036)]
void arc_DataSmooth([in] long PtSmooth);

Visual Basic Sample Code

‘divide current file by a stored file
Dim PtSmooth As Integer

```
If ARCDataViewX1.arc_DataSmooth(PtSmooth) <> 0 Then
               MsgBox (“Smoothing executed”)
               Else
               MsgBox (“Failed to Smooth File”)
               End If
```

Delphi Sample Code

```
SmoothVar := smoothing factor;
if ARCDataViewX1.arc_DataSmooth(SmoothVar) <> 0
       then showmessage(‘Smoothing performed')
       else showmessage(‘Failed to smooth File');
```

arc_DisplayChartToolbar

Description: This function will display the chart tool bar

Parameters:

Visible	0	hide
	!0	show

Definitions:

Delphi

Procedure arc_DisplayChartToolbar (Visible: Integer) ; safecall;

Microsoft IDL

[id(0x00000037)]
void arc_DisplayChartToolbar([in] long Visible);

Visual Basic Sample Code

```
‘ Show/Hide the chart tool bar
ARCDataViewX1.arc_DisplayChartToolbar (ToolBarChk.Value)
```

Delphi Sample Code

```
if ToolBarChk.Checked
   then ARCDataViewX1.arc_DisplayChartToolbar(1)
   else ARCDataViewX1.arc_DisplayChartToolbar(0);
```

arc_DisplayComponents

Description:	In data files where two or more data inputs are processed in real time such as when the input from one data channel is divided by a second or a stored data file, the resultant spectrum is stored as the data file. However, the raw data from the separate channels are also stored in the data file. These data can be displayed by using this function

Parameters:

Visible	0	hide
	!0	show

Definitions:

Delphi	procedure arc_DisplayComponents (Visible: Integer); safecall;
Microsoft IDL	[id(0x00000034)] void arc_DisplayPeaks([in] long Visible);

Visual Basic Sample Code

ARCDataViewX1.arc_DisplayComponents (PeakChk.Value)

Delphi Sample Code

```
if PeakChk.checked
   then ARCDataViewX1.arc_DisplayComponents(1)
   else ARCDataViewX1.arc_DisplayComponents(0);
```

arc_DisplayPeaks

Description:	This function will label the peaks on the chart display

Parameters:

Visible	0	hide
	!0	show

Definitions:

Delphi

procedure arc_DisplayPeaks (Visible: Integer); safecall;

Microsoft IDL

[id(0x00000032)]
void arc_DisplayPeaks([in] long Visible);

Visual Basic Sample Code

ARCDataViewX1.arc_DisplayPeaks (PeakChk.Value)

Delphi Sample Code

```
if PeakChk.checked
   then ARCDataViewX1.arc_DisplayPeaks(1)
   else ARCDataViewX1.arc_DisplayPeaks(0);
```

arc_DivSpectrum

Description: This function will divide the Y values of the currently open file by the Y values of the specified data file at equivalent X values

Parameters:

FileName The name of the data file to be divided into the current file

Result:

0	Failed to open data file
!0	Data file opened

Definitions:

Delphi

function arc_DivSpectrum(FileName : OleVariant): Integer; safecall;

Microsoft IDL

[id(0x00000029)]
long arc_DivSpectrum([in] VARIANT FileName);

Visual Basic Sample Code

```
‘divide current file by a stored file
Dim FileName  As Variant
If ARCDataViewX1.arc_DivSpectrum(FileName) <> 0 Then
               MsgBox (“Division executed”)
               Else
               MsgBox (“Failed to Divide File”)
               End If
```

Delphi Sample Code

```
FileVar := newFileName;
if ARCDataViewX1.arc_DivSpectrum(FileVar) <> 0
       then showmessage(‘Divided File : ‘ +ExtractFileName(newFileName))
       else showmessage(‘Failed to Divide File : ‘ +ExtractFileName(newFileName));
```

arc_FindPeak

Description: This function finds the highest peak in the data file and returns its X and Y coordinates as well as the peak FWHM in the current Spectral units. To find multiple peaks use the arc_getPeaks function.

Parameters:

X	X axis value of the peak
Y	Y axis value of the peak
Peak Halfwidth	Full width half max value of the peak (resolution)

Definitions:

Delphi

procedure arc_FindPeak(**out** X: Double; **out** Y: Double; **out** PeakHalfWidth: Double); safecall;

Microsoft IDL

```
[id(0x0000002e)]
void arc_FindPeak(
[out] double* X,
[out] double* Y,
[out] double* PeakHalfWidth);
```

Visual Basic Sample Code

```
‘get the largest peak in the file
Dim PeakPosition  As Double
Dim PeakIntensity As Double
Dim PeakHalfWidth As Double

ARCDataViewX1.arc_FindPeak(PeakPosition, PeakIntensity, PeakHalfwidth)
  MsgBox (“The peak is at : “ & PeakPosition _
                    & “ Peak value is : “ & PeakIntensity _
                    & “ Peak Half Width is : “ & PeakHalfWidth)
```

Delphi Sample Code

```
begin //peak
ARCDataViewX1.arc_FindPeak(PeakX,PeakY,PeakHalfWidth);
     showmessage(‘The peak is at : ‘ +FloatToStr(PeakX)
     +^M +’Peak value is : ‘ +FloatToStr(PeakY)
     +^M +’Peak Half Width is : ‘ +FloatToStr(PeakHalfWidth));
end;
```

arc_FirstDerivative

Description: This function will take the first derivative of the data set

Parameters: None

Definitions:

Delphi

Procedure arc_FirstDerivative; safecall;

Microsoft IDL

[id(0x00000022)]
void arc_FirstDerivative();

Visual Basic Sample Code

```
‘Take the first derivative
ARCDataViewX1.arc_FirstDerivative
```

Delphi Sample Code

```
ARCDataViewX1.arc_FirstDerivative
```

arc_getLog

Description:	This function will display all of the parameters stored in the data file header
Parameters:	
Log Array	Variant array of log text
Line Count	Number of elements in log array

Results:

0	Invalid
!0	Valid

Definitions:

Delphi

procedure arc_getLog (**out**: LogArray: OleVariant; **out** lineCount: integer): Integer; safecall;

Microsoft IDL

```
[id(0x00000033)]
long arc_getLog(
[out] VARIANT* LogArray,
[out] long* LineCount);
```

Visual Basic Sample Code

```
If ARCDataViewX1.arc_getLog(LogData, DataCount) <> 0 Then
GenericMemo.Clear
‘ show the Log we received
If DataCount > 0 Then
  For DataLoop = 0 To DataCount - 1 Step 1
     GenericMemo.AddItem (LogData(DataLoop))
     Next DataLoop
  End If
Else
GenericMemo.Clear
End If
```

Delphi Sample Code

```
if ARCDataViewX1.arc_getLog(LogData,DataCount) <> 0
then begin
     GenericMemo.Lines.Clear;
     if DataCount > 0
     then begin
          for DataLoop := 0 to DataCount - 1
           do begin
              GenericMemo.Lines.Add (String(LogData[DataLoop]));
              end;
          end;
     end
else begin
     GenericMemo.Lines.Clear;
     GenericMemo.Lines.Add('No Log Data!');
     end;
```

arc_getPeaks

Description:	This function produces an array of XY values calculated to be peaks using the discriminate value for peak determination

Parameters:

DiscrimPercent	Using the highest peak as a reference, all peaks above the specified percent of the maximum Y value will be labeled output as XY pairs.
X Array	Array of X values for all of the peaks
Y Array	Array of Y values for all of the peaks
Count	Number of XY pairs

Result:

0	Invalid
!0	Valid

Definitions:

Delphi

procedure arc_getPeaks(**out** Xarray: Olevariant; **out** Yarray: OleVariant; **out** Count: Integer): Integer; safecall;

Microsoft IDL

```
[id(0x00000031)]
long arc_getPeaks(
[in] double DiscrimPercent,
[out] VARIANT* Xarray,
[out] VARIANT* Yarray,
[out] long* Count);
```

Visual Basic Sample Code

```
‘ get a listing of peaks
Dim Discrim As Double
Discrim = PeakSortEdt.Text
If ARCDataViewX1.arc_getPeaks(Discrim, XData, YData, DataCount) <> 0 Then
GenericMemo.Clear
‘ show the XY pairs we received
If DataCount > 0 Then
   For DataLoop = 0 To DataCount - 1 Step 1
      GenericMemo.AddItem (XData(DataLoop) & “  “ & YData(DataLoop))
      Next DataLoop
   End If
Else
GenericMemo.Clear
End If
```

Delphi Sample Code

```
var
   XData : OLEVariant;
   YData : OLEVariant;
   DataCount : integer;
   DataLoop  : Integer;
   Discrim : double;

Discrim := StrToFloat(Trim(PeakSortEdt.text));
if ARCDataViewX1.arc_getPeaks(Discrim,XData,YData,DataCount) <> 0
then begin
     GenericMemo.Lines.Clear;
     if DataCount > 0
     then begin
          for DataLoop := 0 to DataCount - 1
           do begin
              GenericMemo.Lines.Add (String(XData[DataLoop]) +’  ‘ +String(YData[DataLoop]));
              end;
          end;
     end
else begin
     GenericMemo.Lines.Clear;
     GenericMemo.Lines.Add(‘No Data!’);
     end;
```

arc_getXY

Description:	This function will return an array of XY pairs
Parameters:	None
X values array	Array of all X values
Y values array	Array of all Y values
Count	Number of XY pairs

Result:

0	Invalid
!0	Valid

Definitions:

Delphi

function arc_getXY(**out** Xarray: OleVariant; **out** Yarray: OleVariant;
out Count: Integer): safecall;

Microsoft IDL

```
[id(0x00000030)]
long arc_getXY(
[out] VARIANT* Xarray,
[out] VARIANT* Yarray,
[out] long* Count);
```

Visual Basic Sample Code

```
‘retrieve the XY data into variant array’s
Private Sub XYBtn_Click()
Dim XData As Variant
Dim YData As Variant
Dim DataCount As Long
Dim DataLoop As Long

If ARCDataViewX1.arc_getXY(XData, YData, DataCount) <> 0 Then
GenericMemo.Clear
‘ show the XY pairs we received
If DataCount > 0 Then
  For DataLoop = 0 To DataCount - 1 Step 1
    GenericMemo.AddItem (XData(DataLoop) & “  “ & YData(DataLoop))
    Next DataLoop
  End If
Else
GenericMemo.Clear
End If
End Sub
```

Delphi Sample Code

```
var
   XData : OLEVariant;
   YData : OLEVariant;
   DataCount : integer;
   DataLoop  : Integer;
begin

if ARCDataViewX1.arc_getXY(XData,YData,DataCount) <> 0
then begin
     GenericMemo.Lines.Clear;
     if DataCount > 0
     then begin
          for DataLoop := 0 to DataCount - 1
           do begin
              GenericMemo.Lines.Add (String(XData[DataLoop]) +’  ‘ +String(YData[DataLoop]));
              end;
          end;
     end
else begin
     GenericMemo.Lines.Clear;
     GenericMemo.Lines.Add(‘No Data!’);
     end;
GenericMemo.visible := true;

end;
```

arc_getY

Description:	This function will return the Y value for a specified X value

Parameters:

X	The value of the X coordinat e
Result	The Value of the Y coordinate

Definitions:

Delphi	function arc_getY(X: Double): Double; safecall;
Microsoft IDL	[id(0x00000026)] double arc_getY([in] double X);

Visual Basic Sample Code

```
‘get the Y value for a specified X value
Dim XValue As Double
Dim ResultValue As Double
ResultValue = ARCDataViewX1.arc_getY(XValue)
            MsgBox (“Y Value is : “ + ResultValue)
```

Delphi Sample Code

```
Var
Xvalue: Double;
ResultValue: Double;
ResultValue := ARCDataViewX1.arc_getY(Xvalue);
ShowMessage(‘Y Value is : ‘ +FloatToStr(ResultValue));
```

arc_JoinSpectrum

Description: This function will append the XY pairs from a specified data file to the currently open data file. In the case where the X values of the files overlap the data from the current file will remain.

Parameters:

FileName	The name of the data file to be joined to the current file

Result:

0	Failed to open data file
!0	Data file opened

Definitions:

Delphi

function arc_JoinSpectrum(FileName : OleVariant): Integer; safecall;

Microsoft IDL

[id(0x0000002b)]
long arc_JoinSpectrum([in] VARIANT FileName);

Visual Basic Sample Code

```
‘append data from another file
Dim FileName  As Variant
If ARCDataViewX1.arc_JoinSpectrum(FileName) <> 0 Then
               MsgBox (“Joined Files”)
               Else
               MsgBox (“Failed to Join Files”)
               End If
```

Delphi Sample Code

```
FileVar := newFileName;
if ARCDataViewX1.arc_JoinSpectrum(FileVar) <> 0
       then showmessage(‘JoinedFile : ‘ +ExtractFileName(newFileName))
       else showmessage(‘Failed to Join File : ‘ +ExtractFileName(newFileName));
```

arc_Linearize

Description: This function is used after conversion from X value units to another It will respace the X-axis by extrapolation. The X values are based upon the smallest X increment in the original data set. This function is typically used to convert from one spectral working to another; nanometers to wavenumbers for example. Data collected in pixels can not be converted to other working units.

Parameters: None

Definitions:

Delphi

procedure arc_Linearize; safecall;

Microsoft IDL

[id(0x00000025)]
void arc_Linearize();

Visual Basic Sample Code

```
ARCDataViewX1.arc_Linearize
```

Delphi Sample Code

```
ARCDataViewX1.arc_Linearize
```

arc_Ln

Description: This function will take the log (base e) of all Y components in the data file.

Parameters: None

Definitions:

Delphi

Procedure arc_Ln; safecall;

Microsoft IDL

[id(0x0000001e)]
void arc_Ln();

Visual Basic Sample Code

```
'Take the Ln of the Y data
ARCDataViewX1.arc_Ln
```

Delphi Sample Code

```
ARCDataViewX1.arc_Ln
```

arc_Log

Description: This function will take the log (base 10) of all Y components in the data file.

Parameters: None

Definitions:

Delphi

Procedure arc_Log; safecall;

Microsoft IDL

[id(0x0000001f)]
void arc_Log();

Visual Basic Sample Code

```
'Take the Log of the Y data
ARCDataViewX1.arc_Log
```

Delphi Sample Code

```
ARCDataViewX1.arc_Log
```

arc_MultSpectrum

Description: This function will multiply the Y values from the currently open Data file by the Y values of a specified data file at equivalent X values

Parameters:

FileName

The name of the data file to be the multiplier

Result:

0	Failed to open data file
!0	Data file opened

Definitions:

Delphi

function arc_MultSpectrum(FileName : OleVariant): Integer; safecall;

Microsoft IDL

[id(0x0000002a)] long arc_MultSpectrum([in] VARIANT FileName);

Visual Basic Sample Code

```
‘add stored data to open file
Dim FileName  As Variant
If ARCDataViewX1.arc_AddSpectrum(FileName) <> 0 Then
               MsgBox (“Added File”)
               Else
               MsgBox (“Failed to Add File”)
               End If
```

Delphi Sample Code

```
FileVar := newFileName;
if ARCDataViewX1.arc_AddSpectrum(FileVar) <> 0
       then showmessage(‘Added File : ‘ +ExtractFileName(newFileName))
       else showmessage(‘Failed to Add File : ‘ +ExtractFileName(newFileName));
```

arc_OpenFile

Description:	This function is used to open a stored file for display in the data viewer
Parameters:	
FileName	The name of the file to be opened
Result:	0 Failed to open
	!0 File opened

Definitions:

Delphi	procedure arc_OpenFile (FileName: OleVariant): Integer; safecall
Microsoft IDL	[id(0x00000010)] long arc_OpenFile([in] VARIANT FileName);

Visual Basic Sample Code

```
‘ load the data file
   If ARCDataViewX1.arc_OpenFile(FileName) <> 0 Then
      MsgBox (“Opened File”)
      Else
      MsgBox (“Failed to Open File”)
      End If
```

Delphi Sample Code

```
FileVar := newFileName;
  if ARCDataViewX1.arc_OpenFile(FileVar) <> 0
       then showmessage(‘Opened File : ‘ +ExtractFileName(newFileName))
        else showmessage(‘Failed to Open File : ‘ +ExtractFileName(newFileName));
```

arc_Recalculate

Description: This function will allow you to recalculate the real-time processed data from the raw data stored in the file

Parameters:

Visible	0	hide
	!0	show

Definitions:

Delphi procedure arc_Recalculate (EqnIndex: Integer); safecall;

Microsoft IDL

[id(0x00000035)]
long arc_Recalculate([in] long EqnIndex);

Visual Basic Sample Code

```
If ARCDataViewX1.arc_Recalculate(ReCalcCombo.ListIndex) = 0 Then
  MsgBox (“Failed to recalculate”)
  End If
```

Delphi Sample Code

```
if ARCDataViewX1.arc_Recalculate(ReCalcCombo.itemindex) = 0
   then showmessage(‘Failed to recalculate !’);
```

arc_SaveFile

Description:	This function is used to store the currently displayed data in a file
Parameters:	
FileName	The name of the file to be saved

Result:

0	Failed to save
!0	File saved

Definitions:

Delphi procedure arc_SaveFile (FileName: OleVariant): Integer; safecall

Microsoft IDL [id(0x00000012)]
long arc_SaveFile([in] VARIANT FileName);

Visual Basic Sample Code

```
‘ save the data
If ARCDataViewX1.arc_SaveFile(FileName) <> 0 Then
      MsgBox (“File saved”)
      Else
      MsgBox (“Failed to save file”)
      End If
```

Delphi Sample Code

```
FileVar := newFileName;
  if ARCDataViewX1.arc_SaveFile(FileVar) <> 0
       then showmessage(‘Saved File : ‘ +ExtractFileName(newFileName))
        else showmessage(‘Failed to save File : ‘ +ExtractFileName(newFileName));
```

arc_ScalarAddK

Description: This function will add a constant value, K, to all Y components if the data file.

Parameters:

K The value to be added to each data point

Definitions:

Delphi

Procedure arc_ScalarAddK(K: Double); safecall;

Microsoft IDL

[id(0x00000017)]
void arc_ScalarAddK([in] double K);

Visual Basic Sample Code

```
‘Add a constant
Dim Kvalue as Double
ARCDataViewX1.arc_scalarAddK(Kvalue)
```

Delphi Sample Code

```
var
Kvalue : double;
ARCDataViewX1.arc_ScalarAddK(Kvalue)
```

arc_ScalarDivideByK

Description: This function will all Y components in the data file by a value, K.

Parameters:

K	The value to be used as the divisor

Definitions:

Delphi

Procedure arc_ScalarDivideByK(K: Double); safecall;

Microsoft IDL

[id(0x0000001a)]
void arc_ScalarDivideByK([in] double K);

Visual Basic Sample Code

```
‘Divide Y values by a constant
Dim Kvalue as Double
ARCDataViewX1.arc_scalarDivideBy(Kvalue)
```

Delphi Sample Code

```
var
Kvalue : double;
ARCDataViewX1.arc_ScalarDivideByK(Kvalue)
```

arc_ScalarDivIntoK

Description:	This function will divide all Y components in the file into a constant, K.
Parameters:	
K	The value to be used as the dividend

Definitions:

Delphi	Procedure arc_ScalarDivIntoK(K: Double); safecall;
Microsoft IDL	[id(0x0000001b)] void arc_ScalarSubK([in] double K);

Visual Basic Sample Code

```
‘Divide Y values into a constant
Dim Kvalue as Double
ARCDataViewX1.arc_scalarDivIntoK(Kvalue)
```

Delphi Sample Code

```
var
Kvalue : double;
ARCDataViewX1.arc_ScalarDivIntoK(Kvalue)
```

arc_ScalarMultK

Description: This function will multiply all Y components in the data file by a value, K.

Parameters:

K	The value of the multiplier

Definitions:

Delphi

Procedure arc_ScalarMultK(K: Double); safecall;

Microsoft IDL

[id(0x0000001c)]
void arc_ScalarMultK([in] double K);

Visual Basic Sample Code

```
‘Multiply Y values by a constant
Dim Kvalue as Double
ARCDataViewX1.arc_scalarMultK(Kvalue)
```

Delphi Sample Code

```
var
Kvalue : double;
ARCDataViewX1.arc_ScalarMultK(Kvalue)
end;
```

arc_ScalarRaiseK

Description: This function will raise all Y components in the data file to the power value, K.

Parameters:

K	The value to of the power

Definitions:

Delphi

Procedure arc_ScalarRaiseK(K: Double); safecall;

Microsoft IDL

[id(0x0000001d)]
void arc_ScalarRaiseK([in] double K);

Visual Basic Sample Code

```
‘Raise all Y values to the power K
Dim Kvalue as Double
ARCDataViewX1.arc_scalarRaiseK(Kvalue)
```

Delphi Sample Code

```
var
Kvalue : double;
ARCDataViewX1.arc_ScalarRaiseK(Kvalue)
```

arc_ScalarSubFromK

Description:	This function will subtract all Y components in the data file from a Value, K.
Parameters:	
K	The value from which each data point is subtracted
Definitions:	
Delphi	Procedure arc_ScalarSubFromK(K: Double); safecall;
Microsoft IDL	[id(0x00000019)] void arc_ScalarSubFromK([in] double K);

Visual Basic Sample Code

```
‘Subtract Y values from a constant
Dim Kvalue as Double
ARCDataViewX1.arc_scalarSubFromK(Kvalue)
```

Delphi Sample Code

```
var
Kvalue : double;
ARCDataViewX1.arc_ScalarSubFromK(Kvalue)
```

arc_ScalarSubK

Description:	This function will subtract a constant value, K, from all Y components in the data file.
Parameters:	
K	The value to be subtracted from each data point
Definitions:	
Delphi	Procedure arc_ScalarSubK(K: Double); safecall;
Microsoft IDL	[id(0x00000018)] void arc_ScalarSubK([in] double K);

Visual Basic Sample Code

```
‘subtract a constant
Dim Kvalue as Double
ARCDataViewX1.arc_scalarSubK(Kvalue)
```

Delphi Sample Code

```
var
Kvalue : double;
ARCDataViewX1.arc_ScalarSubK(Kvalue)
```

arc_SecondDerivative

Description: This function will take the second derivative of the data set

Parameters: None

Definitions:

Delphi

Procedure arc_SecondDerivative; safecall;

Microsoft IDL

[id(0x00000023)]
void arc_SecondDerivative();

Visual Basic Sample Code

```
'Take the second derivative of the data
ARCDataViewX1.arc_SecondDerivative
```

Delphi Sample Code

```
ARCDataViewX1.arc_SecondDerivative
```

arc_SubSpectrum

Description: This function will subtract the Y values from a specified data file from the currently open data file at equivalent X values

Parameters:

FileName	The name of the data file to be subtracted from the current file	
Result	0	Failed to open data file
	!0	Data file opened

Definitions:

Delphi

function arc_SubSpectrum(FileName : OleVariant): Integer; safecall;

Microsoft IDL

[id(0x00000028)]
long arc_SubSpectrum([in] VARIANT FileName);

Visual Basic Sample Code

```
‘subtract stored data from open file
Dim FileName  As Variant
If ARCDataViewX1.arc_SubSpectrum(FileName) <> 0 Then
               MsgBox (“Subtraction succeeded “)
               Else
               MsgBox (“Failed to Subtract File”)
               End If
```

Delphi Sample Code

```
FileVar := newFileName;
if ARCDataViewX1.arc_SubSpectrum(FileVar) <> 0
       then showmessage(‘Subtracted File: ‘ +ExtractFileName(newFileName))
       else showmessage(‘Failed to Add File : ‘ +ExtractFileName(newFileName));
```

arc_Trunc

Description:	This function removes XY pairs from a data file outside specified X value boundaries.

Parameters:

X1	The lower X value for the truncated data file
X2	The upper X value for the truncated data file

Definitions:

Delphi

procedure arc_Trunc(X1: Double; X2: Double); safecall;

Microsoft IDL

[id(0x0000002c)]
void arc_Trunc(
[in] double X1,
[in] double X2);

Visual Basic Sample Code

```
‘add stored data to open file
Dim LowLimit  As Double
Dim UpperLimit As Double
ARCDataViewX1.arc_Trunc(LowLimit, UpperLimit)
```

Delphi Sample Code

```
X1 := StrToFloat(Trim(SpectralX1Edt.text));
X2 := StrToFloat(Trim(SpectralX2Edt.text));
ARCDataViewX1.arc_Trunc(X1,X2);
```

arc_Xshift

Description:	This function will increment the values of the X components of the data fileby the value (Xshift)

Parameters:

XShift	The numeric value in the current working units that each X component of the XY pairs is to be incremented

Definitions:

Delphi	Procedure arc_XShif(XShif : Double) ; safecall;
Microsoft IDL	[id(0x00000024)] void arc_XShift([in] double XShift);

Visual Basic Sample Code

```
‘Shift X axis values by a given amount
Dim XShift As Double
ARCDataViewX1.arc_Xshift(Xshift)
```

Delphi Sample Code

```
XShift : double;

ARCDataViewX1.arc_XShift(XShift)
```

arc_ZoomAutoScale

Description: This function zooms the display between the minimum and maximum X and Y values

Definitions:

Delphi

Procedure arc_ZoomAutoScale; safecall;

Microsoft IDL

[id(0x00000013)]
void arc_ZoomAutoScale();

Visual Basic Sample Code

```
ARCDataViewX1.arc_ZoomAutoScale
```

Delphi Sample Code

```
ARCDataViewX1.arc_ZoomAutoScale;
```

arc_ZoomPercent

Description: This function zooms the display to the specified percentage of full Scale

Parameters:

Percent The percentage of full scale desired

Definitions:

Delphi Procedure arc_ZoomPercent(Percent: Double); safecall;

Microsoft IDL [id(0x00000014)]
void arc_ZoomPercent([in] double Percent);

Visual Basic Sample Code

```
Dim percent As Double

percent = ZoomPercentEdt.Text ‘ convert value
ARCDataViewX1.arc_ZoomPercent (percent) ‘ zoom percent
```

Delphi Sample Code

```
var Percent : double;

Percent := StrToFloat(Trim(ZoomPercentEdt.text));
ARCDataViewX1.arc_ZoomPercent(Percent);
```

arc_ZoomX

Description: This function zooms the X axis between two specified limits

Parameters:

X1	The lower X limit value (wavelength)
X2	The upper X limit value (wavelength)

Definitions:

Delphi

Procedure arc_ZoomX(X1 : Double; X2 : Double); safecall;

Microsoft IDL

[id(0x00000015)]
void arc_ZoomX(
[in] double X1,
[in] double X2);

Visual Basic Sample Code

```
Dim X1 As Double
Dim X2 As Double

X1 = ZoomX1Edt.Text ‘ convert X1
X2 = ZoomX2Edt.Text ‘ convert X1

Call ARCDataViewX1.arc_ZoomX(X1, X2)
```

Delphi Sample Code

```
X1,X2 : double;

X1 := StrToFloat(Trim(ZoomX1Edt.text));
X2 := StrToFloat(Trim(ZoomX2Edt.text));
ARCDataViewX1.arc_ZoomX(X1,X2);
```

arc_ZoomY

Description: This function zooms the Y axis between two specified limits

Parameters:

Y1 The lower Y limit value (intensity)

Y2 The upper Y limit value (intensity)

Definitions:

Delphi

Procedure arc_ZoomY(Y1 : Double; Y2 : Double); safecall;

Microsoft IDL

```
[id(0x00000016)]
void arc_ZoomY(
[in] double Y1,
[in] double Y2);
```

Visual Basic Sample Code

```
Dim Y1 As Double
Dim Y2 As Double

Y1 = ZoomY1Edt.Text ‘ convert Y1
Y2 = ZoomY2Edt.Text ‘ convert Y1

Call ARCDataViewX1.arc_ZoomY(Y1, Y2)
Exit Sub
```

Delphi Sample Code

```
  Y1,Y2 : double;

Y1 := StrToFloat(Trim(ZoomY1Edt.text));
Y2 := StrToFloat(Trim(ZoomY2Edt.text));
ARCDataViewX1.arc_ZoomY(Y1,Y2);
```

CCDInLightPath

Description: Indicates if the CCD is in the light path (active detector).

Parameters:

Result:	0	Not in the light path (not active detector)
	!0	In the light path (currently active detector)

Definitions:

Delphi

Function CCDInLightPath: Integer; safecall;

Microsoft IDL

[id(0x00000021)]
HRESULT CCD_Present([out, retval] long* Present);

Visual Basic Sample Code

```
If SpectraSense_COM.CCDInLightPath = 0 Then
       CCDLightLabel.Caption = "CCD is not in the light path"
    Else
       CCDLightLabel.Caption = "CCD is in the light path"
    End If
```

Delphi Sample Code

```
// display if the CCD is in the light path
    if SpectraSenseServer.CCDInLightPath = 0
       then CCDLightLabel.Caption := 'CCD is not in the light path'
       else CCDLightLabel.Caption := 'CCD is in the light path';
```

CCDLoadMapFile

Description:	Loads a stored CCD area map file

Parameters:

MapFileName	The name of the file to be loaded
Result:	0 Failed to load the map !0 Successfully loaded the map file

Definitions:

Delphi

function CCDLoadMapFile(MapFileName: OleVariant):
Integer; safecall;

Microsoft IDL

[id(0x00000005)]
HRESULT CCDLoadMapFile(
[in] VARIANT MapFileName,
[out, retval] long* FileLoaded);

Visual Basic Sample Code

```
FileVar = “C:\fooMap.arc_map”
If  SpectraSense_COM.CCDLoadMapFile(FileVar) = 0 then
    MsgBox(“Failed to Load File”)
End If
```

Delphi Sample Code

```
filevar := ‘C:\FooMap.arc_map’
if SpectraSenseServer.CCDLoadMapFile(filevar) = 0
   then showmessage(‘Failed to Load Map File’);
```

CCDMapName

Description: Returns the name of the current CCD area map

Parameters:

Result: The name of the CCD area map

Definitions:

Delphi

function CCDMapName: OleVariant; safecall;

Microsoft IDL

[id(0x00000006)]
HRESULT CCDMapName([out, retval] VARIANT* FileName);

Visual Basic Sample Code

```
‘ display the map file name
Label1.Caption = SpectraSense_COM.CCDMapName
```

Delphi Sample Code

```
tempStr := string(SpectraSenseServer.CCDMapName);
Label2.caption: = ‘Map is’ +(tempStr);
```

CCD_Present

Description: Indicates if a CCD is in the hardware configuration

Parameters:

Results:

0	Not present
!0	Present

Definitions:

Delphi

Function CCD_Present: Integer;safecall;

Microsoft IDL

[id(0x00000021)]
HRESULT CCD_Present([out, retval] long* Present);

Visual Basic Sample Code

```
If SpectraSense_COM.CCD_Present = 0 Then
   Label1.Caption = “CCD is not present”
End If
```

Delphi Sample Code

```
if SpectraSenseServer.CCD_Present = 0
    then Label1.caption := ‘CCD is not present’;
```

Filter_Present

Description:	Determines if a filter wheel is defined in the hardware configuration
Parameters:	
Filter_Num	On systems without an NCL only a value of 1 is valid. On NCL based systems 1 or 2 are valid values, however 2 is reserved for special applications
Result:	0 An invalid request !0 A valid request

Definitions
Delphi

function Filter_Present(Filter_Num: Integer): Integer; safecall;

Microsoft IDL

```
[id(0x0000001b)]
HRESULT Filter_Present(
                [in] long Filter_Num,
                [out, retval] long* Present);
```

Visual Basic Sample Code

```
If SpectraSense_COM.Filter_Present(1) <> 0 Then
   Label1.caption = “Filter Present”
   End If
```

Delphi Sample Code

```
if SpectraSenseServer.Filter_Present(1) <> 0
   then Label1.caption := ‘Filter Present’;
```

getCCDInputByte

Description:	Returns the state of all input bits (1 – 8) on the CCD ST-133 controller
Parameters:	
Result:	Returns a byte that corresponds to the TTL state of the Input port of the ST-133 controller
Definitions:	
Delphi	function getCCDInputByte: Shortint; safecall;
Microsoft IDL	[id(0x00000025)] HRESULT getCCDInputByte([out, retval] char* InByte);

Visual Basic Sample Code

Visual Basic does not support binary operators. Each line has to be read out individually using the getCCDInputLine function.

Delphi Sample Code

```
// read the state of the input lines
TempByte := SpectraSenseServer.getCCDInputByte;
```

getCCDInputLine

Description:	Returns the state of a specific input line on the ST-133 controller

Parameters:

Line_Num	Valid line numbers (1 – 8)	
Result	0	TTL false
	!0	TTL true

Definitions:

Delphi

```
Function getCCDInputLine(Line_Num: Integer): Integer; safecall;
```

Microsoft IDL

```
[id(0x00000026)]
HRESULT getCCDInputLine(
                              [in] long Line_Num,
                              [out, retval] long* LineState);
```

Visual Basic Sample Code

```
‘ read Line 1
If SpectraSense_COM.getCCDInputLine(1) = 0 Then ‘ a 0 if off
   CCDInShape1.FillColor = &H0&
   Else
   CCDInShape1.FillColor = &FF&
End if
```

Delphi Sample Code

```
// read an individual Line, no need to read lines at once,
// if your only monitoring one.
if SpectraSenseServer.getCCDInputLine(1) = 0
   then show message (‘line off’)
```

getCCDITime

Description:	Returns the integration time in milliseconds
Parameters:	
Result:	Integration time
Definitions:	
Delphi	Function getCCDITime: Integer; safecall;
Microsoft IDL	[id(0x00000023)] HRESULT getCCDITime([out, retval] long* ITime);

Visual Basic Sample Code

```
‘ display the CCD ITime
    Label1.Text = SpectraSense_COM.getCCDITime
```

Delphi Sample Code

```
Label1.caption  := ‘ITime : ‘ +IntToStr(SpectraSenseServer.getCCDITime) +’ msec’;
```

getCCDOutputByte

Description: Returns the state of all outputs line on the ST-133 controller.

Parameters:

Result: Integer

Definitions:

Delphi

function getCCDOutputByte: Shortint; safecall;

Microsoft IDL

[id(0x00000027)]
HRESULT getCCDOutputByte([out, retval] char* OutByte);

Visual Basic Sample Code

```
if SpectraSenseServer.getCCDOutputbyte = 0 then
   msgbox (“failed to read output”)
end if
```

Delphi Sample Code

```
if SpectraSenseServer.getCCDOutputByte = 0
   then show message (‘lines not read’);
```

Note: Visual Basic does not support this function

getCCDOutputLine

Description: Returns the state of a specific output line on the ST-133

Parameters:

Line_Num	Valid line number (1 – 8)

Result:

0	TTL false
!0	TTL true

Definitions:

Delphi

Function getCCDOutputLine(Line_Num: Integer): Integer;
safecall;

Microsoft IDL

[id(0x00000029)]
HRESULT getCCDOutputLine(
[in] long Line_Num,
[out, retval] long* LineState);

Visual Basic Sample Code

```
if SpectraSenseServer.getCCDOutputLine(2) = 0 then
   msgbox (“line not set”)
end if
```

Delphi Sample Code

```
if SpectraSenseServer.getCCDOutputLine(2) = 0
   then show message (‘line not set’);
```

getFilterPosition

Description: Returns the current filter position number in the light path

Parameters:

Filter_Num On systems without an NCL only a value of 1 is valid. On NCL based systems 1 or 2 are valid values, however 2 is reserved for special applications

Result:

0	An invalid request
1- 6	The filter wheel position that is in the light path

Definitions:

Delphi function getFilterPosition(Filter_Num: Integer): Integer; safecall;

:

Microsoft IDL

```
[id(0x0000001c)]
HRESULT getFilterPosition(
                    [in] long Filter_Num,
                    [out, retval] long*
                    FilterPosition);
```

Visual Basic Sample Code

```
' see which filter is in the optical path
If SpectraSense_COM.getFilterPosition(1) = 6 Then
   Label1.caption = "Filter number" &SpectraSense_COM.getFilterPosition & "is in place"
   End If
```

Delphi Sample Code

```
\if SpectraSenseServer.getFilterPosition(1) = 2
   then label1.caption :=('filter number 2");
```

getifNCLInputLineAvail

Description: Inquire if an input line to the NCL has been assigned to the COM interface.

Parameters:

Line_Num	Valid line numbers (1 – 4)
Result:	0 Not assigned !0 Assigned

Definitions:

Delphi

```
Function getifNCLInputLineAvail(    Line_Num: Integer
                                   ): Integer; safecall;
```

Microsoft IDL

```
[id(0x0000002b)]
HRESULT getifNCLInputLineAvail(
                                [in] long Line_Num,
                                [out, retval] long* Avail);
```

Visual Basic Sample Code

```
‘ see if the input lines are allocated, if so read them
If SpectraSense_COM.getifNCLInputLineAvail(1) <> 0 Then
   If SpectraSense_COM.getNCLInputLine(1) <> 0 Then
        NCLInShape1.FillColor = &HFF00&
        Else
        NCLInShape1.FillColor = &H0&
        End If
```

Delphi Sample Code

```
// see if the input lines are allocated, if so read them
if SpectraSenseServer.getifNCLInputLineAvail(1) <> 0
then begin
     if SpectraSenseServer.getNCLInputLine(1) <> 0
        then NCLInShape1.Brush.color := clLime
        else NCLInShape1.Brush.color := clBlack;
     End;
```

getifNCLOutputLineAvail

Description: Inquire if a specific output line has been assigned to the COM interface

Parameters:

Line_Num	Valid numbers (1 - 4)
Result	0 Not available !0 Available

Definitions:

Delphi

Function getifNCLOutputLineAvail(Line_Num: Integer):
Integer; safecall;

Microsoft IDL

[id(0x0000002e)]
HRESULT getifNCLOutputLineAvail(
[in] long Line_Num,
[out, retval] long* Avail);

Visual Basic Sample Code

```
‘ see if the output lines are allocated and read them
If SpectraSense_COM.getifNCLOutputLineAvail(1) <> 0 Then
   If SpectraSense_COM.getNCLOutputLine(1) <> 0 Then
      NCLOutShape1.FillColor = &HFF00&
      Else
      NCLOutShape1.FillColor = &H0&
      End If
   End If
```

Delphi Sample Code

```
// see if the output lines are allocated and read them
if SpectraSenseServer.getifNCLOutputLineAvail(1) <> 0
then begin
     if SpectraSenseServer.getNCLOutputLine(1) <> 0
        then NCLOutShape1.Brush.color := clLime
        else NCLOutShape1.Brush.color := clBlack;
     end;
```

getMonoDiverterPosition

Description: This function indicates which slit is in the light path. Note that the same function is used with different variables for determining the active entrance and exit slit.

Parameters:

Mono_Num The number of the monochromator being interrogated

Divert_Num

1	Entrance mirror
2	Exit mirror
1	Slave Entry Mirror (double monochromator)
2	Slave Exit mirror (double monochromator)

Result:

0	Failed
1	side entrance slit
2	front entrance slit
3	front exit slit
4	side exit slit
5	slave side entrance slit
6	slave front entrance slit
7	slave front exit slit
8	slave side exit slit

Note: If you have only one monochromator and are not using an NCL then use monochromator 1. When using an NCL use the number of the of the connection between the monochromator and NCL.

Defintions:

Delphi

```
Function getMonoDiverterPosition(       Mono_Num: Integer;
                                        Divert_Num: Integer;
                                    ): Integer; safecall;
```

Microsoft IDL

```
[id(0x00000010)]
HRESULT getMonoDiverterPosition(
                                        [in] long Mono_Num,
                                        [in] long Divert_Num,
                                        [out, retval] long* Slit_Num);
```

Visual Basic Sample Code

```
If SpectraSense_COM.setMonoDiverterPosition(curmono, 1) = 1 Then
   Label1.caption “The side entrance slit is active”
End If
```

Delphi Sample Code

```
case SpectraSenseServer.getMonoDiverterPosition(curmono,1) of
     1  show message (‘Side Entrance’);
     2  show message (‘Front Entrance’)
     end;
```

getMonoGrating

Description:	Returns the currently selected grating number
Parameters:	
Mono_Num	The number of the monochromator being interrogated
Result:	The current grating number in the specified monochromator

Note: If you have only one monochromator and are not using an NCL then use monochromator 1. When using an NCL use the number of the of the connection between the monochromator and NCL.

Definitions:

Delphi

```
Function getMonoGrating(        Mono_Num:integer;
                               ): Integer; safecall;
```

Microsoft IDL

```
[id(0x0000000e)]
HRESULT getMonoGrating(
                        [in] long Mono_Num,
                        [out, retval] long* Grating_Num);
```

Visual Basic Sample Code

```
Dim newGrat As Integer
newGrat = SpectraSense_COM.getMonoGrating(1)
GratingNumEdt.Text = newGrat
```

Delphi Sample Code

```
var GratingNumber : integer;
GratingNumber : = SpectraSenseServer.getMonoGrating(1)
ShowMessage(‘Mono 1 grating is : ‘ +int to Str(grating number);
```

getMonoSlitWidth

Description: Returns the slit width in microns only.

Parameters:

Mono_Num	The number of the monochromator being interrogated

Slit_Num:

0	Failed
1	side entrance slit
2	front entrance slit
3	front exit slit
4	side exit slit
5	slave side entrance (doubles only usually the intermediate slit)
6	slave front entrance (doubles only)
7	slave front exit slit (doubles only)
8	slave side exit slit (doubles only)

Result: The width of the slit in microns

Note: If you have only one monochromator and are not using an NCL then use monochromator 1. When using an NCL use the number of the of the connection between the monochromator and NCL.

Definitions:

Delphi

```
Function getMonoSlitWidth(   Mono_Num: Integer;
                             Slit_Num: Integer
                          ):  Integer; safecall;
```

Microsoft IDL

```
[id(0x00000013)]
HRESULT getMonoSlitWidth(
                              [in] long Mono_Num,
                              [in] long Slit_Num,
                              [out, retval] long* Width);
```

Visual Basic Sample Code

```
Slit1Edt.Text = SpectraSense_COM.getMonoSlitWidth(curmono, 1)
```

Delphi Sample Code

```
var curmono: Integer;
Slit1Edt.text := IntToStr(SpectraSenseServer.getMonoSlitWidth(curmono,1));
```

getMonoWavelength

Description:	This function will return the position of the specified monochromator in the spectral units defined in the Hardware Configuration Screen; nm, Anstroms,eV, Relative wavenumbers, Absolute wavenumbers
Parameters:	
Mono_Num	The number of the monochromator being interrogated
Result:	The monochromator position in current units

Note: If you have only one monochromator and are not using an NCL then use monochromator 1. When using an NCL use the number of the of the connection between the monochromator and NCL.

Definitions:

Delphi

```
Function getMonoWavelength(  Mono_Num:Integer
                           ): Double; safecall;
```

Microsoft IDL

```
[id(0x0000000c)]
HRESULT getMonoWaveLength(
                           [in] long Mono_Num,
                           [out, retval] double* WaveLength);
```

Visual Basic Sample Code

```
Dim Monowavelength As Double
Monowavelength = SpectraSense_COM.getMonoWavelength(1)
Label1.Caption = “Mono 1 is at” & Monowavelength “nm”
```

Delphi Sample Code

```
Label1.Caption := FloatToStr(SpectraSenseServer.getMonoWavelength(2));
```

getNCLChHV

Description:	Returns the HV for the specified channel
Parameters:	
Ch_Num	Valid channels in a 2 channel NCL are (1,2) Valid channels in a 3 channel NCL are (1,2,3)
Result:	The value of the HV in volts for the specified channel. A zero is returned if the channel is not set up or there is no HV present.

Definitions:

Delphi

```
Function  getNCLChHV(          Ch_Num: Integer
                    ): Integer; safecall;
```

Microsoft IDL

```
[id(0x0000003b)]
HRESULT getNCLChHV(
                          [in] long Ch_Num,
                          [out, retval] long* HVVoltage);
```

Visual Basic Sample Code

```
Label1.caption = SpectraSense_COM.getNCLChHV(1)
```

Delphi Sample Code

```
Label1.caption := IntToStr(SpectraSenseServer.getNCLChHV(1))
```

getNCLChHVon

Description: Determines if a high voltage associated with a specific channel is on.

Parameters:

Ch_Num	Valid channels in a 2 channel NCL are (1,2) Valid channel in a 3 channel NCL are (1,2,3)

Result:

0	HV not turned on
!0	HV is on

Definitions:

Delphi

Function getNCLChHVon(Ch-Num:Integer): Integer; safecall;

Microsoft IDL

```
[id(0x00000039)]
HRESULT getNCLChHVon(
                    [in] long Ch_Num,
                    [out, retval] long* HVState);
```

Visual Basic Sample Code

```
If SpectraSense_COM.getNCLChHVon(1) <> 0 Then
   Label1.caption = “Channel 1 HV is on!”
Else
   Label1.caption = “Channel 1 HV is off!”
End If
```

Delphi Sample Code

```
if SpectraSenseServer.getNCLChHVon <> 0
   then Label1.caption := ‘Channel 1 HV is on!’
   else Label1.caption := ‘Channel 1 HV is off!’;
```

getNCLITime

Description: Returns the integration time in milliseconds

Parameters:

Result: The integration time in milliseconds.

Definitions:

Delphi

Function getNCLITime: Integer; safecall;

Microsoft IDL

[id(0x00000017)]
HRESULT getNCLITime([out, retval] long* ITime);

Visual Basic Sample Code

```
ITimeLabel.Caption = “ITime : “ & SpectraSense_COM.getNCLITime & “ msec”
```

Delphi Sample Code

```
ITimeLabel.caption := ‘ITime : ‘ +IntToStr(SpectraSenseServer.getNCLITime) +’ msec’;
```

getNCLInputLine

Description:	Reads the state of an input line

Parameters:

Line_Num	Valid line numbers are (1-4)

Result:

0	Not set
!0	Set

Definitions:

Delphi

function getNCLInputLine(Line_Num: Integer): Integer; safecall;

Microsoft IDL

[id(0x00000030)]
HRESULT getNCLOutputLine(
[in] long Line_Num,
[out, retval] long* LineState);

Visual Basic Sample Code

```
‘ see if the input lines are allocated, if so read them
If SpectraSense_COM.getifNCLInputLineAvail(1) <> 0 Then
   If SpectraSense_COM.getNCLInputLine(1) <> 0 Then
        NCLInShape1.FillColor = &HFF00&
   Else
        NCLInShape1.FillColor = &H0&
   End If
End If
```

Delphi Sample Code

```
// see if the input lines are allocated, if so read them
if SpectraSenseServer.getifNCLInputLineAvail(1) <> 0
then begin
     if SpectraSenseServer.getNCLInputLine(1) <> 0
        then NCLInShape1.Brush.color := clLime
        else NCLInShape1.Brush.color := clBlack;
     end;
```

getNCLOutputLine

Description: Reads the state of a particular output line

Parameters:

Line_Num		Valid numbers are (1 – 4)
Result	0	An invalid request
	!0	A valid request

Definitions:

Delphi

Function getNCLOutputLine(Line_Num): Integer; safecall;

Microsoft IDL

```
[id(0x00000030)]
HRESULT getNCLOutputLine(
        [in] long Line_Num,
        [out, retval] long* LineState);
```

Visual Basic Sample Code

```
‘ see if the output lines are allocated and read them
If SpectraSense_COM.getifNCLOutputLineAvail(1) <> 0 Then
   If SpectraSense_COM.getNCLOutputLine(1) <> 0 Then
      NCLOutShape1.FillColor = &HFF00&
      Else
      NCLOutShape1.FillColor = &H0&
      End If
   End If
```

Delphi Sample Code

```
// see if the output lines are allocated and read them
if SpectraSenseServer.getifNCLOutputLineAvail(1) <> 0
then begin
     if SpectraSenseServer.getNCLOutputLine(1) <> 0
        then NCLOutShape1.Brush.color := clLime
        else NCLOutShape1.Brush.color := clBlack;
     end;
```

getNCLShutterPosition

Description:	Returns the current position of the shutter on an NCL (Does not apply to CCD's).

Parameters:

ShutterNum	The number of the shutter being interrogated

Result:

0	shutter is closed
!0	shutter is open

Note: This only works with NCL's that directly control the shutter, and does not work with shutters attached to CCD's

Definitions:

Delphi

Function getNCLShutterPosition(ShutterNum: Integer): Integer; safecall;

Microsoft IDL

```
[id(0x00000041)]
HRESULT getNCLShutterPosition(
        [in] long ShutterNum,
        [out, retval] long* Open);
```

Visual Basic Sample Code

```
If SpectraSense_COM.getNCLShutterPosition(1) = 1 then
   Label1.caption = 'Shutter1 is open'
Else
   Label1.caption = 'Shutter1 is closed'
EndIf
```

Delphi Sample Code

```
If SpectraSenseServer.getNCLShutterPosition(1) = 1
   Then Label1.caption = 'Shutter1 is open'
   Else Label1.caption = 'Shutter1 is closed';
```

getPageNum

Description: This function will return which of the SpectraSense pages is currently up.

Parameters:

Result:

0	About Screen
1	Hardware Configuration Screen
2	Hardware Status Screen
3	Survey Mode Screen
4	Acquisition Screen
5	Live Data Screen
6	Post Processing Screen (no remote possibilities)
7	Terminal Mode Screen (no remote possibilities)
8	Reserved for future use

Definitions:

Delphi

Function GetPageNum: Integer; safecall;

Microsoft IDL

[id(0x00000002)]
HRESULT GetPageNum ([out, retval] long* ValidRequest);

Visual Basic Sample Code

```
Dim PageNum As Long
PageNum = SpectraSense_COM.getPageNum
```

Delphi Sample Code

```
PageNum : Long;
PageNum := SpectraSenseServer.getPageNum;
```

getScanStorageName

This function will return the name under which the data will be stored

Parameters:

Result: The name under which the data is to be stored

Definitions:

Delphi

Function GetScanStorageName:Olevarient; safecall

Microsoft IDL

```
[id (0x00000040)]
HRESULT  getScanStorageName 9([out,  retval] VARIANT*
                                        StorageName);
```

Visual Basic Example Code

```
Dim StorageName As Variant
StorageName = SpectraSense_COM.getScanStorageName
```

Delphi Example Code

```
StorageName : OLEVariant;
StorageName := SpectraSenseServer.getScanStorage;
```

Note : This function is of limited value unless the user manually controls SpectraSense. The Scan function's first argument sets this value.

isFilterSetToAutoInsert

Description:	Determines if the filter AutoInsertion function is active
Parameters:	
Filter_Num	On systems without an NCL only a value of 1 is valid. On NCL based systems 1 or 2 are valid values, however 2 is reserved for special applications
Result:	0 Not set to autoinsert !0 Set to autoinsert

Definitions:

Delphi

```
function isFilterSetToAutoInsert(Filter_Num: Integer):
                                Integer; safecall;
```

:

Microsoft IDL

```
[id(0x0000001e)]
HRESULT isFilterSetToAutoInsert(
                                [in] long Filter_Num,
                                [out, retval] long*
                                AutoInsert);
```

Visual Basic Sample Code

```
' display if it is or is not auto-insertable
    If SpectraSense_COM.isFilterSetToAutoInsert(1) = 0 Then
       Label2.Caption = "Not set to auto insert filters"
       Else
       Label2.Caption = "Set to auto insert filters"
       End If
```

Delphi Sample Code

```
// display if it is or is not auto-insertable
    if SpectraSenseServer.isFilterSetToAutoInsert(1) = 0
       then FilterAutoInsertStateLabel.caption := 'Not set to auto insert filters'
       else FilterAutoInsertStateLabel.caption := 'Set to auto insert filters'
```

isSpectraSenseUp

Definition:	Tells if SpectraSense has finished loading

Parameters:

Result:	0	has not finished
	!0	has finished

Definitions:

Delphi

Function IsSpectraSenseUp: Integer;safecall;

Microsoft IDL

[id(0x0000003d)]
HRESULT IsSpectraSenseUp ([out, retval] long* StateBool);

Visual Basic Sample Code

```
If SpectraSense_COM.IsSpectraSenseUp <> 0 then
    Label1.caption="SpectraSense is loaded"
Endif
```

Delphi Example Code

```
if SpectraSenseServer.IsSpectraSenseUp <> 0
      then label.caption = 'SpectraSense is loaded';
```

LastSavedFileNames

This function will return a list of files saved from the last acquisition

Description: Returns a list of files saved from the last acquisition

Parameters:

NumSavedFiles	Number of files that were saved in the last acquisition
SavedFileNames	A variant array of the files saved

Result:

0	An invalid request
!0	A valid request

Definitions:

Delphi

```
Function LastSavedFile Names(    out NumSavedFiles:Integer
                                 out SavedFileNames: OleVariant
                               ): integer; safecall;
```

Microsoft IDL

```
[id(0x00000009)]
HRESULT LastSavedFileNames(
                [out] long* NumSavedFiles,
                [out] VARIANT* SavedFileNames,
                [out, retval] long* ValidRequest);
```

Visual Basic Sample Code

```
If SpectraSense_COM.LastSavedFileNames(NumSaved, DataSaved) <> 0 Then
   Then MsgBox(NumSaved &” Files Saved”);
```

Delphi Sample Code

```
if SpectraSenseServer.LastSavedFileNames(NumSaved,DataSaved) <> 0
   then showmessage(IntToStr(NumSaved +’ Files Saved’);
```

LoadRoutine

This function is used to load an existing acquisition routine. The routine typically is created within the normal SpectraSense environment and saved.

Description:	Load an acquisition routine
Parameters:	
RoutineName	The name of the routine to be loaded
LoadMapFile	Load the CCD map file if specified
MatchHardware	0 Just load the scan parameters and do not validate hardware parameters
	!0 Validate the hardware parameters while loading
Silent _Match:	0 Alert user to all hardware mismatches if MatchHardware = 1
	!0 Quietly correct all hardware mismatches
Result:	0 An invalid request
	!0 A valid request

Definitions:

Delphi

```
Function LoadRoutine(        RoutineName: Ole Variant
                             LoadMapFile,
                             MatchHardware: Integer;
                             Silent_Match: Integer

                     ): Integer;safecall;
```

MicroSoft IDL

```
 [id (0x000000005)]
HRESULT  LoadRoutine (
                             [in]  VARIANT RoutineName,
                             [in]   long LoadMapFile,
                             [in]   long MatchHardware,
                             [in]   long Silent_Match,
                             [out, retval] long* FileLoaded);
```

Visual Basic Example Code

```
Dim MatchInt As Long
Dim SilentInt As Long
Dim MapInt As Long
Dim FileVar As Variant

FileVar = “c:\fooRoutine.rtn”
If SpectraSense_COM.LoadRoutine(FileVar, MapInt, MatchInt, SilentInt)=0 then
       MsgBox (”Failed to load routine”)
End if
```

Delphi Example Code

```
var matchint,
    silentint,
    mapint    : integer;
    filevar   : OleVariant;
Begin
Filevar: = ‘c:fooRoutine.rtn’

If SpectraSenseServer.LoadRoutine(filevar,mapint,matchint,silentint);
   Then showmessage (‘Failed to load routine’);
end;
```

MonoDouble

Description: Determines if a monochromator is a double or single

Parameters:

Mono_Num The number of the monochromator being interrogated

Result:

0	if it is not a double monochromator
!0	if it is a double monochromator

Note: If you have only one monochromator and are not using an NCL then use monochromator 1. When using an NCL use the number of the of the connection between the monochromator and NCL.

Definitions:

Delphi

```
Function MonoDouble(        Mono_Num: Integer
                    ): Integer;safecall;
```

Microsoft IDL

```
[id(0x0000000b)]
HRESULT MonoDouble(
                        [in] long Mono_Num,
                        [out, retval] long* Present);
```

Visual Basic Sample Code

```
If SpectraSense_COM.MonoDouble(1) = 0 then
       Label1.caption = “Monochromator 1 is not a double”
End if
```

Delphi Sample Code

```
if SpectraSenseServer.MonDouble(2) <> 0
     then Showmessage (‘Monochromator 2 is a double’);
```

Mono_Present

Description: Used to determine if a specific monochromator is in the system (1 or 2)

Parameters:

Mono_Num	The monochromator you are talking to

Result:

0	Not Present
!0	Present

Note: If you have only one monochromator and are not using an NCL then use monochromator 1. When using an NCL use the number of the of the connection between the monochromator and NCL.

Definitions:

Delphi

```
Function Mono_Present(      Mono_Num:Integer): Integer; safecall;
```

Microsoft IDL

```
        [id(0x0000000a)]
    HRESULT Mono_Present(
                    [in] long Mono_Num,
                    [out, retval] long* Present);
```

Visual Basic Sample Code

```
If SpectraSense_COM.Mono_Present(1) = 0 then
       Label1.caption = “Monochromator 1 not present”
End if
```

Delphi Sample Code

```
if SpectraSenseServer.Mon_Present(2) = 0
     then label1.caption = ‘Mono 2 not present’;
```

MonoSlitMotorized

Description: Determines if a particular slit is manual or motorized

Parameters:

Mono_Num The number of the monochromator being interrogated

Slit_Num

0	Failed
1	side entrance slit
2	front entrance slit
3	front exit slit
4	side exit slit
5	slave side entrance (doubles only usually the intermediate slit)
6	slave front entrance (doubles only)
7	slave front exit slit (doubles only)
8	slave side exit slit (doubles only)

Result:

0	Manual Slit
!0	Motorized Slit

Note: If you have only one monochromator and are not using an NCL then use monochromator 1. When using an NCL use the number of the of the connection between the monochromator and NCL.

Definitions:

Delphi

```
Function MonoSlitMotorized(          Mono_Num: Integer;
                                     Slit_Num: Integer
                           ): Integer; safecall;
```

Microsoft IDL

```
[id(0x00000012)]
HRESULT MonoSlitMotorized(
                              [in] long Mono_Num,
                              [in] long Slit_Num,
                              [out, retval] long* Motorized);
```

Visual Basic Sample Code

```
Dim curmono As Long
If SpectraSense_COM.MonoSlitMotorized(curmono, 1) <> 0 Then
            MsgBox (“Side Entrance Slit is Motorized”)
EndIf
```

Delphi Sample Code

```
// side entrance slit , slit # 1
if SpectraSenseServer.MonoSlitMotorized(curmono,1) <> 0
        then showmessage (‘Side Entrance Slit is Motorized’);
```

NCLChRead

Description: Read the signal from a specified channel in the units specified in the Hardware Configuration – Detector tab.

Parameters:

Ch_Num Valid channels in a 2 channel NCL are (1,2)
Valid channels in a 3 channel NCL are (1,2,3)

Read_Value
The current value of the requested channel

Result:

0	An invalid request
!0	A valid request

Definitions:

Delphi

```
function  NCLChRead(        Ch_Num: Integer;
                            out Read_Value: Double
                    ): Integer; safecall;
```

Microsoft IDL

```
[id(0x0000001a)]
HRESULT NCLChRead(
                    [in] long Ch_Num,
                    [out] double* Read_Value,
                    [out, retval] long* ValidRequest);
```

Visual Basic Sample Code

```
Dim ReadValue As Double
‘ read channel 1
If SpectraSense_COM.NCLChRead(1, ReadValue) <> 0 Then
   Ch1Value.Caption = “Value = “ & ReadValue
   Else
   Ch1Value.Caption = “Error Reading Channel 1”
   End If
```

Delphi Sample Code

```
var ReadValue : double;
// read channel 1
if SpectraSenseServer.NCLChRead(1,ReadValue) <> 0
   then Ch1Value.caption := ‘Value = ‘ +Format(‘%.2f’,[ReadValue])
   else Ch1Value.caption := ‘Error Reading Channel 1’;
```

NCLChReadAll

Description:	Reads all of the detector channels at once. The value of the output is in the units specified in the Hardware Configuration Screen Detection Tab.
Parameters:	
Ch_Num:	Valid channels in a 2 channel NCL are (1,2) Valid channels in a 3 channel NCL are (1,2,3)
Ch1_Value	The output value from channel 1
Ch2_Value	The output value from channel 2
Ch3_Value	The output value from channel 3
Result:	0 An invalid request !0 A valid request

Note: All channels in system must be set up. In a two channel NCL Channel 3 will return a 0.

Definitions:

Delphi

```
function NCLChReadAll(     out Ch1_Value: Double;
                           out Ch2_Value: Double;
                           out Ch3_Value: Double
                         ): Integer; safecall;
```

Microsoft IDL

```
[id(0x00000036)]
HRESULT NCLChReadAll(
                     [out] double* Ch1_Value,
                     [out] double* Ch2_Value,
                     [out] double* Ch3_Value,
                     [out, retval] long* ValidRequest);
```

Visual Basic Sample Code

```
Dim ch1 As Double
Dim ch2 As Double
Dim ch3 As Double

If SpectraSense_COM.NCLChReadAll(ch1, ch2, ch3) <> 0 Then
        Ch1all.Caption = ch1
        Ch2all.Caption = ch2
        Ch3all.Caption = ch3
   Else
        Ch1all.Caption = “----“
        Ch2all.Caption = “----“
        Ch3all.Caption = “----“
   End If
```

Delphi Sample Code

```
// read all channels at once
var ch1,ch2,ch3 : double;

if SpectraSenseServer.NCLChReadAll(ch1,ch2,ch3) <> 0
   then begin
        Ch1all.caption := Format(‘%.3f’,[Ch1]);
        Ch2all.caption := Format(‘%.3f’,[Ch2]);
        Ch3all.caption := Format(‘%.3f’,[Ch3]);
        end
   else begin
        Ch1all.caption := ‘----‘;
        Ch2all.caption := ‘----‘;
        Ch3all.caption := ‘----‘;
        end;
```

NCL_ChPresent

Description: Tells if a detector has be assigned to a particular NCL signal input

Parameters:

Ch_Num	Valid channels in a 2 channel system are (1,2) Valid channels in a 3 channel system are (1,2,3)

Result:

0	Channel not assigned or not present
!0	Channel assigned

Definitions:

Delphi

Function NCL_ChPresent(Ch_Num): Integer; safecall;

Microsoft IDL

```
[id(0x00000019)]
HRESULT NCL_ChPresent(
                [in] long Ch_Num,
                [out, retval] long* Present);
```

Visual Basic Sample Code

```
If SpectraSense_COM.NCL_ChPresent(1) <> 0 Then
   Ch1Label.Caption = “Channel 1 is Present”
Else
   Ch1Label.Caption = “Channel 1 is not Present”
End If
```

Delphi Sample Code

```
if SpectraSenseServer.NCL_ChPresent(1) <> 0
   then Ch1Label.caption := ‘Channel 1 is Present’
   else Ch1Label.caption := ‘Channel 1 is not Present’;
```

NCL_Present

Description:	Determines if an NCL is in the system

Parameters:

Result:	0	No NCL present
	!0	NCL present

Definitions:

Delphi

Function NCL_Present: Integer; safecall;

Microsoft IDL

[id(0x00000016)]
HRESULT NCL_Present([out, retval] long* Present);

V**isual Basic Sample Code**

```
If SpectraSense_COM.NCL_Present = 0 Then
         NCLLabel.Caption = “NCL is not Present”
Endif
```

Delphi Sample Code

```
// display some NCL Info
if SpectraSenseServer.NCL_Present = 0
    then NCLLabel.caption := ‘NCL is not Present’;
```

RoutineName

Description: This function returns the name of the current acquisition routine

Parameters:

Result: The name of the current routine

Definitions:

Delphi

Function RoutineName: OleVariant; safecall;

Microsoft IDL

[id (0x00000004)]
HRESULT RoutineName 9[out, retval] VARIANT* FileName);

Visual Basic Sample Code

```
Dim RoutineName As VARIANT
RoutineName = SpectraSense_COM.RoutineName
```

Delphi Sample Code

```
RoutineName : OLEVariant;
RoutineName: = SpectraSenseServer.RoutineName;
```

Scan

This function will cause a spectrum(s) to be taken using the current acquisition parameters

Description: Instructs the software to acquire data

Parameters:

NewDataName	The name under which the data will be stored
NumSavedFiles	Number of files that were saved by the scan routine
SavedFileNames	A variant array of the names of the files saved
Result:	0 Failed to scan !0 Scan succeeded

Definitions:

Delphi

```
Function  Scan  (NewDataName: OleVariant;
                 Out NumSavedFiles: Integer
                 Var SavedFileNames: OleVariant) : Integer; safecall;
```

Microsoft IDL

```
[id (0x00000008)]
HRESULT Scan (
[in] VARIANT NewDataName,
[out] long* NumSavedFileNames,
[out, retval]  long* ValidRequest) ;
```

Visual Basic Example Code

```
Private Sub ScanBtn_Click()
Dim DataVar As Variant ‘the storage name
Dim DataSaved As Variant ‘the name of all files saved
Dim NumSaved As Long ‘number of files saved

DataVar = “c:\foodata.arc_data”
If spectraSense_COM.Scan (DataVar, NumSaved, DataSaved) =0 then
       Msgbox (“scan failed”)
End If
```

Delphi Example Code

```
DataVar   : OleVariant; // the storage name
DataSaved : OleVariant; // the list of all files stored
Numsaved  : Integer;

DataVar:=’C:\foodata.arc_data’;
If SpectraSenseServer.Scan(DataVar,NumSaved,DataSaved) = 0
   Then showmessage (‘Scan Failed’);
```

SelectPage

Description: This function will select which of the SpectraSense pages is to be visible.

Parameters:

PageNum:	0	About Screen
	1	Hardware Configuration Screen
	2	Hardware Status Screen
	3	Survey Mode Screen
	4	Acquisition Screen
	5	Live Data Screen
	6	Post Processing Screen (no remote possibilities)
	7	Terminal Mode Screen (no remote possibilities)
	8	Reserved for future use

Result:	0	An invalid request
	!0	A valid request

Definitions:

Delphi

Function SelectPage(PageNum: Integer): Integer; safecall;

Microsoft IDL

```
[id(0x00000001)]
HRESULT SelectPage(
                    [in] long PageNum,
                    [out, retval] long* ValidRequest);
```

Visual Basic Sample Code

```
Dim PageNum As Long
If SpectraSense_COM.SelectPage(PageNum) = 0 then
Msgbox (“Failed to select page”)
End If
```

Delphi Sample Code

```
PageNum : integer;
SpectraSenseServer.SelectPage(PageNum) = 0
  then show message (‘Failed to select page’);
```

setCCDITime

Description:	Sets the CCD exposure time

Parameters:

NewTime	The new exposure time in milliseconds	
Result:	0	An invalid request
	!0	A valid request

Definitions:

Delphi

Function setCCDITime(newITime: Integer): Integer; safecall;

Microsoft IDL

```
[id(0x00000024)]
HRESULT setCCDITime(
                    [in] long newITime,
                    [out, retval] long* ValidRequest);
```

Visual Basic Sample Code

```
newITime = 500
‘ set the CCD ITime
If SpectraSense_COM.setCCDITime(newITime) = 0 Then
   MsgBox (“Error : unable to set ITime”)
   End If
```

Delphi Sample Code

```
// set the CCD ITime
if SpectraSenseServer.setCCDITime(newITime) = 0
   then showmessage(‘Error : unable to set ITime’);
```

setCCDMathParams

The CCD Math Parameters are used to define how the incoming data will be treated before display and storage. Refer to the SpectraSense software manual for detailed descriptions of each of these parameters.

Description: Input the real-time processing parameters for a CCD acquisition

Parameters:

AreaOp	0	No operation between areas	
	1	subtract	
	2	divide by	
AreaOpNum	The area on the chip that will operate using the AreaOp function on all of the other active areas.		
FileOp	0	Areas	Just show the areas
	1	Areas /File	FileName must be specified
	2	Absorbance	FileName must be specified
	3	Transmittance	FileName must be specified
	4	Reflectance	FileName must be specified
FileName	Input of the file used in calculating the above operations		
CosmicCorrect	0	Not active	
	!0	Operation is performed	
DarkSubtract	0	Not active	
	!0	Operation is performed	
SoftwareBin	0	Not active	
	!0	Operation is performed	
Results:	0	an invalid request	
	!0	a valid request	

Definitions:

Delphi

```
Function setCCDMathParams(      AreaOp: Integer;
                                AreaOpNum: Integer;
                                FileOp: Integer;
                                FileName: OleVariant;
                                CosmicCorrect: Integer;
                                DarkSubtract: Integer,
                                SoftwareBin: Integer
                           ): Integer; safecall;
```

Microsoft IDL

```
[id(0x00000034)]
HRESULT setCCDMathParams(
                        [in] long AreaOp,
                        [in] long AreaOpNum,
                        [in] long FileOp,
                        [in] VARIANT FileName,
                        [in] long CosmicCorrect,
                        [in] long DarkSubtract,
                        [in] long SoftwareBin,
                        [out, retval] long* ValidRequest);
```

Visible Basic Sample Code

```
Dim FileName As VARIANT
Dim newFileName As VARIANT
Dim AreaOp As Integer
Dim AreaOpNum As Integer
Dim FileOp As Integer
Dim CosmicCorrect As Integer
Dim DarkSubtract As Integer
Dim SoftwareBin As Integer

If SpectraSenSe_COM.setCCDMathParams(AreaOP,AreaOpNum,FileOp,FileName,CosmicCorrect,
                                      DarkSubtract,SoftwareBin)<> 0 then
       Label2.Caption = “Parameters successfully loaded”
End If
```

Delphi Sample Code

```
    newFileName  : string;
    varFileName  : OLEVariant;
    AreaOp       : Integer;
    AreaOpNum    : Integer;
    FileOp       : Integer;
    CosmicCorrect: Integer;
    DarkSubtract : Integer;
    SoftwareBin  : Integer;

// load the scan parameters
if spectraSenseServer.setCCDMathParams( AreaOp,AreaOpNum,FileOp,
                                                 varFileName,CosmicCorrect,
                                                 DarkSubtract,SoftwareBin) = 0
// a zero result is failed
then showmessage(‘Failed to Load Parameters!’);
```

setCCDOutputByte

Description:	This function writes all the output bits on the ST-133 Controller
Parameters:	
ShortInt	A mask of what to set the output port to

Results

0	An invalid request
!0	A valid request

Definitions:

Delphi

```
function setCCDOutputByte(   OutByte: ShortInt
                             ): Integer; safecall;
```

Microsoft IDL

```
[id(0x00000028)]
HRESULT setCCDOutputByte(
                             [in] char OutByte,
                             [out, retval] long*
                             ValidRequest);
```

Visual Basic Sample Code

```
Dim byte As Char
Byte = 256
If setCCDOutputByte(0 Byte) = 0 then
   Msgbox (“Failed to set all lines.”)
End if
```

Delphi Sample Code

```
Var
OutByte : Byte;

OutByte = 256
If setCCDOutputByte (OutByte) = 0;
  Then show message (‘Failed’);
```

setCCDOutputLine

Description:	Sets the state of a specific output line on the ST-133 controller

Parameters:

Line_Num	Valid line numbers are (1 – 8)	
LineState	0	TTL false
	!0	TTL true
Result:	0	An invalid request
	!0	A valid request

Definitions:

Delphi

```
Function setCCDOutputLine(   Line_num: Integer;
                              LineState: Integer):
                            Integer; safecall;
```

Microsoft IDL

```
[id(0x0000002a)]
HRESULT setCCDOutputLine(
                               [in] long Line_Num,
                               [in] long LineState,
                               [out, retval] long*
                                ValidRequest);
```

Visual Basic Sample Code

```
'Turn on Line 1
if SpectraSense_COM.setCCDOutputLine (1,1) <> 0 then
   label1.caption = "Line 1 is active."
End If
```

Delphi Sample Code

```
//Turn on Line 1
if SpectraSenseServer.setCCDOutputLine(1,1) = 0
    then showmessage('Error : Setting Output Line 1');
```

setCCDScanParams

The SetCCDScanParams includes parameters for all possible system configurations. For simple, single spectrometer configurations, the EmisStart and EmisStop are used in default. If you only want to set the spectrograph to a single central position, enter the wavelength in EmisStart and set UseStepnGlue to 0. It is advisable to enter 0's in all parameters not germane to the acquisition.

Description: Input parameters needed to define a spectral CCD acquisition

Parameters:

EmisStart	Starting wavelength of Step and Glue acquisition or central wavelength of a single region acquisition.
EmisStop	Ending wavelength of a Step and Glue acquisition.
UseStepnGlue	0 Do not use the step and glue feature !0 Use the step and glue feature
SmothStepnGlue	0 Do not use the smoothing feature !0 Use the smoothing feature
LinearizeStepnGlue	0 Do not linearize a stepped and glued data !0 Linearize the data.
ExciteStart	If your configuration has a second monochromator being used to Illuminate a sample enter the starting or specific wavelength here.
ExciteStop	Enter the final wavelength of the second monochromator here. Enter a zero if you want the monochromator to rest at ExciteStart or if there is no second monochromator in the configuration.
ExciteIncrement	Enter the number of spectral units (nm) that you wish the second monochromator to move between each cycle of an acquisition.
UseExciteScan	0 Do not use this feature !0 Use this feature
ReadsperCycle	You must enter a non-zero value to this function. The CCD will be read out this number of times and averaged at each acquisition position.
NumCycles	Enter the number of times you wish to repeat the data acquisition. You must enter a non-zero positive number
IntervalBetweenCycles	Enter the time in msec to wait between cycles A zero value here will leave no delay between read outs.
FileNameFormat	0 Post fixed incrementing of names 1 Pre fixed incrementing of names 2 Time Stamped incrementing of names

	3	Signal averaging of cycles
SPCFileFormat	0	single file
	1	MultiAreas
	2	MultiCycles
Result:	0	an invalid request
	!0	a valid request

Definitions:

Delphi

```
Function setCCDScanParams (
                              EmisStart: Double;
                              EmisStop:  Double;
                              EmisNumSteps: Integer;
                              UseStepnGlue:  Integer;
                              SmoothStepnGlue: Integer;
                              LinearizeStepnGlue: Integer;
                              ExciteStart: Double;
                              ExciteStop: Double
                              ExciteIncrement: Double;
                              UseExciteScan: Integer;
                              ReadsperCycle: Integer;
                              NumCycles: Integer;
                              IntervalBetweenCycles: Integer;
                              FileNameFormat: Integer;
                              SPCFileFormat: Integer;
                             ): safecall;
```

Microsoft IDL

```
[id(0x00000033)]
HRESULT setCCDScanParams(
                                     [in] double EmisStart,
                                     [in] double EmisStop,
                                     [in] long EmisNumSteps,
                                     [in] long UseStepnGlue,
                                     [in] long SmoothStepnGlue,
                                     [in] long LinearizeStepnGlue,
                                     [in] double ExciteStart,
                                     [in] double ExciteStop,
                                     [in] double ExciteIncrement,
                                     [in] long UseExciteScan,
                                     [in] long ReadsperCycle,
                                     [in] long NumCycles,
                                     [in] long IntervalBetweenCycles,
                                     [in] long FileNameFormat,
                                     [in] long SPCFileFormat,
                                     [out, retval] long* ValidRequest);
```

Visual Basic Sample Code

```
‘You can create variables for each parameter to help you better keep track of them.
Dim EmisStart As Double
Dim EmisStop As Double
Dim EmisNumSteps As Integer
Dim UseStepnGlue As Integer
Dim SmoothStepnGlue As Integer
Dim LinearizeStepnGlue As Integer
Dim ExciteStart As Double
Dim ExciteStop As Double
Dim ExciteIncrement As Double
Dim UseExciteScan AS Integer
Dim ReadsperCycle As Integer
Dim NumCycles As Integer
Dim IntervalBetweenCycles As Integer
Dim FileNameFormat As integer
Dim SPCFileFormat As Integer

EmisStart = 500
EmisStop = 0
EmisNumSteps = 0                             ‘These parameters are for a series of
UseStepnGlue = 0                             ‘5 spectra taken at 500nm using only
Use LinearizeStepnGlue = 0                   ‘a single spectrograph. There will be
ExciteStart =0                               ‘5 second delay between cycles and the
ExciteStop = 0                               ‘file names will be increment as
ExciteIncrement = 0                          ‘filename01.---
UseExciteScan = 0                            ‘filename02 ….filename05.---
ReadsperCycle = 1
NumCycles = 5
IntervaleBetweenCycles = 5000
FileNameFormat = 1
SPCFileFormat = 0

If SpectraSense_COM.setCCDScanParams(EmisStart,EmisStop,EmisNumSteps,UseStepnGlue,
                                     SmoothStepnGlue,LinearizeStepnGlue,ExciteStart,
                                     ExciteStop,ExciteIncrement,UseExciteScan,ReadsperCycle,
                                     NumCycles,IntervalBetweenCycles,FileNameFormat,
                                     SPCFileFormat)<>0 then
   Label2.Caption = “Whew”
Endif
‘or you can just put in the parameter values
If SpectraSense_COM.setCCDScanParams(500,0,0,0,0,0,0,0,0,1,5,5000,1,0)<> 0 then
   Label2.Caption = “Whew”
End if
```

Delphi Sample Code

```
EmisStart: Double;
EmisStop: Double;
EmisNumSteps: Integer;              // create variables to easily note
UseStepnGlue: Integer;              // the values for each parameter
SmoothStepnGlue: Integer;
LinearizeStepnGlue: Integer;
ExciteStart: Double;
ExciteStop: Double;
ExciteIncrement: Double;
UseExciteScan: Integer;
ReadsperCycle: Integer;
NumCycles: Integer;
IntervalBetweenCycles: Integer;
FileNameFormat: Integer;
SPCFileFormat: Integer;

if SpectraSenseServer.setCCDScanParams(EmisStart,EmisStop,EmisNumSteps,
                                                  UseStepnGlue,SmoothStepnGlue,LinearizeStepnGl
                                                  ue,ExciteStart,ExciteStop,ExciteIncrement,Use
                                                  ExciteScan,ReadsperCycle,NumCycles,IntervalBe
                                                  tweenCycles,FileNameFormat,SPCFileFormat)= 0
   then showmessage('Failed to setup scan!');
```

setCCDScanType

Description: Defines whether a spectrum will be acquired or intensity vs time information (Peak Monitoring) will be acquired.

Parameters

Scan_Value	0	No scan defined
	1	Spectral based acquisition
	2	Peak Monitoring acquisition
Result	0	an invalid request
	!0	a valid request

Definitions

Delphi

Function SetCCDScanType(Scan_Value:Integer):Integer; safecall;

Microsoft IDL

[id(0x00000032)]
HRESULT SetCCDScanType (
[in] long Scan_Value
[out, retval] long* ValidRequest);

Visual Basic Sample Code

```
If SpectraSense_COM.SetScanCCDScanType(1) <> 0 then
       Msgbox (“Set Scan Type”)
End if
```

Delphi Sample Code

```
If SpectraSenseServer(1) <> 0
    Then showmessage(‘Set scan type’);
```

Note: The interface does not support creation peak monitoring routines at this time. Create your routines within SpectraSense and use the LoadRoutine function or create your own acquisitions using the low level functions

setFilterAutoInsertFlag

Description: Set or deactivate the filter wheel auto insertion function

Parameters:

Filter_Num	On systems without an NCL only a value of 1 is valid. On NCL based systems 1 or 2 are valid values, however 2 is reserved for special applications
AutoInsertbool:	0 Deactivate !0 Activate

Definitions:

Delphi

```
function setFilterAutoInsertFlag(Filter_Num: Integer):
                                AutoInsertBool: Integer
                                ): Integer; safecall;
```

:
Microsoft IDL

```
[id(0x0000001f)]
HRESULT setFilterAutoInsertFlag(
                                [in] long Filter_Num,
                                [in] long
                                AutoInsertBool,
                                [out, retval] long*
                                ValidRequest);
```

Visual Basic Sample Code

```
If SpectraSense_COM.setFilterAutoInsertFlag(1, 1) = 0 Then
   Label2.Caption = “Error : Failed to set auto insert !”
   End If
```

Delphi Sample Code

```
// change the filter to autoinsert filters
if SpectraSenseServer.setFilterAutoInsertFlag(1,1) = 0
   then Label2.caption := ‘Error : Failed to set auto insert !’;
```

setFilterPosition

Description:	Sets the filter wheel to a specific position. An invalid position is ignored.
Parameters:	
Filter_Num	On systems without an NCL only a value of 1 is valid. On NCL based systems 1 or 2 are valid values, however 2 is reserved for special applications
newPosition	Sets the filter wheel to this position
Result:	0 An invalid request !0 A valid request

Definitions:

Delphi

```
function setFilterPosition(      Filter_Num: Integer;
                                 NewPosition: Integer
                         ): Integer; safecall;
```

:

Microsoft IDL

```
[id(0x0000001d)]
HRESULT setFilterPosition(
                          [in] long Filter_Num,
                          [in] long newPosition,
                          [out, retval] long*
                          ValidRequest);
```

Visual Basic Sample Code

```
If SpectraSense_COM.setFilterPosition(1, newFilterPos) = 0 Then
   MsgBox (“Error : Failed to set filter position”)
   End If
```

Dephi Sample Code

```
// set the filter position
if SpectraSenseServer.setFilterPosition(1,newFilterPos) = 0
   then showmessage(‘Error : Failed to set filter position’);
```

setMonoDiverterPosition

Description: This function is used to put a specific slit in the light path. This is done by moving the appropriate diverter mirror.

Parameters:

Mono_Num	The number of the monochromator being interrogated	
Divert_Num	1	Entrance mirror
	2	Exit mirror
	3	Slave Entry Mirror (double monochromator)
	4	Slave Exit mirror (double monochromator)
NewSlit_Num	0	Failed
	1	side entrance slit
	2	front entrance slit
	3	front exit slit
	4	side exit slit
	5	slave side entrance (doubles only usually the intermediate slit)
	6	slave front entrance (doubles only)
	7	slave front exit slit (doubles only)
	8	slave side exit slit (doubles only)
Result:	0	An invalid request
	!0	A valid request

Note: If you have only one monochromator and are not using an NCL then use monochromator 1. When using an NCL use the number of the of the connection between the monochromator and NCL.

Definitions:

Delphi

```
Function setMonoDiverterPosition(          Mono_Num: Integer;
                                           Divert_Num: Integer;
                                           NewSlit_Num: Integer
                                  ): Integer; safecall;
```

Microsoft IDL

```
[id(0x00000011)]
HRESULT setMonoDiverterPosition(
                              [in] long Mono_Num,
                              [in] long Divert_Num,
                              [in] long NewSlit_Num,
                              [out, retval] long* ValidRequest);
```

Visual Basic Sample Code

```
If SpectraSense_COM.setMonoDiverterPosition(1, 1, 1) = 0 Then
   MsgBox (“Error : Failed to change divertor”)
   End If
```

Delphi Sample Code

```
// flip entrance diverter
if SpectraSenseServer.setMonoDiverterPosition(curmono,1,1) = 0
   then ShowMessage(‘Error : Failed to change diverter’);
```

setMonoGrating

Description: Puts the specified grating number in position for the specified monochromator

Parameters:

Mono_Num	The number of the monochromator being interrogated
New Grating	The grating number to be put in place

Result:

0	An invalid request
!0	A valid request

Note: If you have only one monochromator and are not using an NCL then use monochromator 1. When using an NCL use the number of the of the connection between the monochromator and NCL.

Definitions:

Delphi

```
Function setMonoGrating(    Mono_Num: Integer;
                            NewGrating: Integer
                        ): Integer; safecall;
```

Microsoft IDL

```
[id(0x0000000f)]
HRESULT setMonoGrating(
                            [in] long Mono_Num,
                            [in] long NewGrating,
                            [out, retval] long* ValidRequest);
                            [id(0x00000010)]
```

Visual Basic Sample Code

```
Dim newGrat As Long
newGrat = 3
‘ set the grating
If SpectraSense_COM.setMonoGrating(curmono, newGrat) = 0 Then
   MsgBox (“Error : Failed to move grating”)
   End If
```

Delphi Sample Code

```
var newGrat : integer;
newGrat := 2;
// set the grating
if SpectraSenseServer.setMonoGrating(1,newGrat) = 0
   then ShowMessage(‘Error : Failed to move grating’);
```

setMonoSlitWidth

Description: Sets the width of the selected slit in microns. Valid only for motorized slits

Parameters:

Mono_Num The number of the monochromator being interrogated

Slit_Num

0	Failed
1	side entrance slit
2	front entrance slit
3	front exit slit
4	side exit slit
5	slave side entrance (doubles only usually the intermediate slit)
6	slave front entrance (doubles only)
7	slave front exit slit (doubles only)
8	slave side exit slit (doubles only)

newWidth The new slit width in microns

Result:

0	An invalid request
!0	A valid request

Note: If you have only one monochromator and are not using an NCL then use monochromator 1. When using an NCL use the number of the of the connection between the monochromator and NCL.

Definitions:

Delphi

```
Function setMonoSlitWidth(         Mono_Num: Integer;
                                   Slit_Num: Integer;
                                   NewWidth: Integer
                          ): Integer; safecall;
```

Microsoft IDL

```
[id(0x00000014)]
HRESULT setMonoSlitWidth(
                                   [in] long Mono_Num,
                                   [in] long Slit_Num,
                                   [in] long newWidth,
                                   [out, retval] long* ValidRequest);
```

Visual Basic Sample Code

```
Dim curmono As long
Dim newWidth As long
' Set slit 1 slit width
If SpectraSense_COM.setMonoSlitWidth(curmono, 1, newWidth) = 0 Then
   MsgBox ("Error : Failed to Set Slit 1 Width")
End if
```

Delphi Sample Code

```
var newWidth : integer;
if SpectraSenseServer.setMonoSlitWidth(curmono,1,newWidth) = 0
   then ShowMessage(‘Error : Failed to Set Slit 1 Width’);
```

setMonoWavelength

Description:	This function will change the position of the specified monochromator. The position will be in the spectral units defined in the Hardware Configuration Screen; nm, Anstroms,eV, Relative wavenumbers, Absolute wavenumbers.
Parameters:	
Mono_Num	The number of the monochromator being interrogated
NewWavelength	Enter the new position for the specified monochromator in selected spectral units.
Result:	0 An invalid request
	!0 A valid request

Note: If you have only one monochromator and are not using an NCL then use monochromator 1. When using an NCL use the number of the of the connection between the monochromator and NCL.

Definitions:

Delphi

```
function setMonoWaveLength(          Mono_Num:Integer;
                                     NewWaveLength: Double
                                 ): Integer; safecall
```

Microsoft IDL

```
[id(0x0000000d)]
HRESULT setMonoWaveLength(
                                     [in] long Mono_Num,
                                     [in] double NewWaveLength,
                                     [out, retval] long* ValidRequest);
```

Visual Basic Sample Code

```
Dim newWave As Double
Dim curmono As Double

Curmono = 1
newWave = 500

‘ set the wavelength
If SpectraSense_COM.setMonoWavelength(curmono, newWave) = 0 Then
   MsgBox (“Error : Failed to set wavelength”)
   End If
```

Delphi ExampleCode

```
var newWave : double;
var curmon : double;

newWave := 500;
curmono : = 1;

// set the wavelength
if SpectraSenseServer.setMonoWavelength(curmono,newWave) = 0;
   then ShowMessage(‘Error : Failed to set wavelength’);
```

setNCLChHV

Description:	Sets a channel's high voltage
Parameters	
Ch_Num:	Valid channels in a 2 channel NCL are (1,2) Valid channels in a 3 channel NCL are (1,2,3)
HHVoltage:	This is the value in volts that you are setting
Result:	0 An invalid request !0 A valid request

Definitions

Delphi

```
function  setNCLChHV(           Ch_Num: Integer;
                                HVVoltage: Integer
                    ): Integer; safecall;
```

Microsoft IDL

```
[id(0x0000003c)]
HRESULT setNCLChHV(
                              [in] long Ch_Num,
                              [in] long HVVoltage,
                              [out, retval] long* ValidRequest);
```

Visual Basic Sample Code

```
If SpectraSense_COM.setNCLChHV(1,500) <> 0 Then
      MsgBox (“Failed”)
End if
```

Delphi Sample Code

```
var
   newVoltage : integer;
newVoltage := 300;
// set the Ch 1 High Voltage
if SpectraSenseServer.setNCLChHV(1,newVoltage) = 0
   then showmessage(‘Error : Failed to set HV’);
```

setNCLChHVon

Description: This function turns a channel's associated HV on or off.

Parameters:

Ch_Num	Valid channels in a 2 channel NCL are (1,2) Valid channels in a 3 channel NCL are (1,2,3)
HVState	0 Turn HV off !0 Turn HV on

Result:

0	An invalid request
!0	A valid request

Definitions:

Delphi

```
function setNCLChHVon(           Ch_Nu,: Integer;
                                 HVState: Integer
                      ): Interger; safecall;
```

Microsoft IDL

```
[id(0x0000003a)]
HRESULT setNCLChHVon(
                [in] long Ch_Num,
                [in] long HVState,
                [out, retval] long* ValidRequest);
```

Visual Basic Sample Code

```
If SpectraSense_COM.setNCLChHVon(1,1) = 0 then
   MsgBox("Failed to Turn on HV!")
End if
```

Delphi Sample Code

```
If SpectraSenseServer.setNCLChHVon(1,1) = 0
   Then showmessage('Failed to Turn on HV!');
```

setNCLITime

Description: Sets the NCL integration time in milliseconds

Parameters:

Itime	The integration time to be input in milliseconds

Result:

0	An invalid request
!0	A valid request

Definitions:

Delphi

Function setNCLITime(Itime:Integer):Integer; safecall;

Microsoft IDL

```
[id(0x00000018)]
HRESULT setNCLITime(
                [in] long ITime,
                [out, retval] long* ValidRequest);
```

Visual Basic Sample Code

```
Private Sub ITimeSetBtn_Click()
Dim newITime As Long

‘ set the NCL ITime
If SpectraSense_COM.setNCLITime(newITime) = 0 Then
   MsgBox (“Error : unable to set ITime”)
   End If
```

Delphi Sample Code

```
var newITime : Integer;
// set the NCL ITime
if SpectraSenseServer.setNCLITime(newITime) = 0
   then ShowMessage(‘Error : unable to set ITime’);
```

setNCLMathParams

This function is used to set up the real time mathematical treatment of the data as it is being collected. This function only works with scanning spectral acquisition. To set up the mathematical treatment of time based acquisition, first set up a routine in SpectraSense and then use the LoadRoutine function to input the parameters.

Description:	This function sets up the real-time processing parameters	
Parameters:		
FileOp	0	Channel
	1	Channel/Ref
	2	Channel – Ref
	3	Absorbance
	4	Reflectance
	5	Transitance
	6	All Channels
FileOpCn	Channel to operated on 1,2, or 3	
FileName	The name of the file to be used as a reference	
LoadRefFileParams	0	No
	!0	Yes
LoadRefFileSilent	0	No
	!0	Yes
SourceCompensate	0	No
	!0	Yes
CompCh	The number of the channel used for source compensation; 1, 2, or 3	
DarkSubtract0	No	
	!0	Yes
Result:	0	An invalid request
	!0	A valid request

Definitions:

Delphi

```
Function setNCLMathParams(   FileOp: Integer;
                             FileOpCh: Integer;
                             FileName: OleVariant;
                             LoadRefFileParams: Integer;
                             LoadRefFileSilent: Integer;
                             SourceCompensate: Integer;
                             CompCh: Integer;
                             DarkSubtract: Integer;
                           ): Integer; safecall;
```

Microsoft IDL

```
[id(0x00000038)]
HRESULT setNCLMathParams(
                              [in] long FileOp,
                              [in] long FileOpCh,
                              [in] VARIANT FileName,
                              [in] long LoadRefFileParams,
                              [in] long LoadRefFileSilent,
                              [in] long SourceCompensate,
                              [in] long CompCh,
                              [in] long DarkSubtract,
                              [out, retval] long* ValidRequest);
```

Visual Basic Example Code

```
Dim FileOpType    As Long ‘ 0 = ch
                          ‘ 1 = ch / ref
                          ‘ 2 = ch - ref
                          ‘ 3 = absorbance, log(ref/ch)
                          ‘ 4 = Reftectance, 100 * (ch / ref)
                          ‘ 5 = Transmittance, 100 * (ch / ref)
                          ‘ 6 = all channels
Dim FileOpCh      As Long ‘ the number of the channel to use (1,2 or 3)
Dim newFileName   As String  ‘ the file name of ref, pass “ if not used
Dim FileVar       As Variant ‘ the file name of ref, pass “ if not used
Dim FileLower     As String
Dim FileParams    As Long ‘ 0 = just load the file data
                          ‘ <> 0 (use 1) = load the ref file scan params, if possible
Dim QuietFileLoad As Long ‘ 0 = verbose load
                          ‘ 1 = silent load
Dim SourceComp    As Long ‘ 0 = no source compensation
                          ‘ <> 0 (use 1) source compensation (ch above = FileOpCh / CompCh)
Dim CompCh        As Long ‘ the number of the channel to use (1,2, or 3)
Dim DarkSub       As Long ‘ 0 = do not take a dark reading
                          ‘ <> 0 (use 1) take a dark reading and subtract it from the raw
readings
FileParams = NCLRTLoadRefParamsChk.Value
FileVar = “C:foodata.arc_data
QuietFileLoad = NCLRTRefSilentLoadChk.Value

If SpectraSense_COM.setNCLMathParams(FileOpType, FileOpCh, _
                                     FileVar, FileParams, QuietFileLoad, _
                                     SourceComp, CompCh, _
                                     DarkSub) _
   = 0 Then ‘ a 0 result is it failed
MsgBox (“Failed to set NCL Math Parameters!”)
End If
```

Delphi Example Code

```
var
FileOpType    : integer; // 0 = ch
                         // 1 = ch / ref
                         // 2 = ch - ref
                         // 3 = absorbance, log(ref/ch)
                         // 4 = Reftectance, 100 * (ch / ref)
                         // 5 = Transmittance, 100 * (ch / ref)
                         // 6 = all channels
FileOpCh      : integer; // the number of the channel to use (1,2 or 3)
newFileName   : string;  // the file name of ref, pass “ if not used
FileVar       : OLEVariant; // the file name of ref, pass “ if not used
FileParams    : integer; // 0 = just load the file data
                         // <> 0 (use 1) = load the ref file scan params, if possible
QuietFileLoad : integer; // 0 = verbose load
                         // 1 = silent load
SourceComp    : integer; // 0 = no source compensation
                         // <> 0 (use 1) source compensation (ch above = FileOpCh / CompCh)
CompCh        : integer; // the number of the channel to use (1,2, or 3)
DarkSub       : integer; // 0 = do not take a dark reading
                         // <> 0 (use 1) take a dark reading and subtract it from the raw
readings
begin
if SpectraSenseServer.setNCLMathParams(FileOpType,FileOpCh,
                                                  FileVar,FilePArams,QuietFileLoad,
                                                  SourceComp,CompCh,
                                                  DarkSub) = 0 // a 0 result is it failed
   then showmessage(‘Failed to set NCL Math Parameters!’);
```

setNCLOutputLine

Description:	Sets the state of a specific NCL output line	
Parameters:		
Line_Num	Valid line number (1- 4)	
LineState	0	Not set
	!0	Set
Results	0	An invalid request
	!0	A valid request

Definitions:

Delphi

```
Function  setNCLOutputLine(    Line_Num; Integer;
                               LineState: Integer
                             ): Integer; safecall;
```

Microsoft IDL

Visual Basic Sample Code

```
If SpectraSense_COM.setNCLOutputLine(1, 0) = 0 Then
   MsgBox (“Failed to set output line 1”)
   End If
```

Delphi Sample Code

```
// turn on output line 1
if SpectraSenseServer.setNCLOutputLine(1,1) = 0
   then showmessage(‘Failed to set output Line 1’);
```

setNCLScanParams

This function will input scan parameters of a scanning single channel acquisition. Create a time based acquisition routine in SpectraSense and use the function LoadRoutine for time based acquisitions.

Description:	Input the scanning parameters for a spectral based scanning acquisition

Parameters:

Mono1Start	The starting wavelength for monochromator 1 (uses current spectral units)
Mono1Stop	The ending wavelength for monochromator 1
Mono1Inc	The increment in spectral units between data points
Mono2Start	The starting wavelength for monochromator 2
Mono2Stop	The ending wavelength for monochromator 2
BackwardsExcite	Scans the monochromator used for excitation from high to low wavelength
ReadsPerPoint	The number of readouts averaged at each position. Valid options are 1, 3, 10
NumCycles	The number of times each scan will be taken

IntervalBetweenCycles: The time between the end of one scan and the start of the next in msec.

FilenameFormat	0	Post Fixed increment
	1	Pre Fixed increment
	2	Signal averaged
	3	SPC multifile (Valid only with Grams enabled software)
	4	Time Stamped
Result:	0	An invalid request
	!0	A valid request

Definitions:

Delphi

```
Function setNCLScanParams(  Mono1Start: Double;
                            Mono1Stop: Double;
                            Mono1Inc: Double;
                            Mono2Start: Double;
                            Mono2Stop: Double;
                            Mono2Inc: Double;
                            BackwardsExcite: Integer;
                            ReadsPerPoint: Integer;
                            NumCycles: Integer;
                            IntervalBetweenCycles: Integer;
                            FileNameFormat: Integer;
                         ): Integer; safecall;
```

Microsoft IDL

```
[id(0x00000037)]
HRESULT setNCLScanParams(
                            [in] double Mono1Start,
                            [in] double Mono1Stop,
                            [in] double Mono1Inc,
                            [in] double Mono2Start,
                            [in] double Mono2Stop,
                            [in] double Mono2Inc,
                            [in] long BackwardsExcite,
                            [in] long ReadsPerPoint,
                            [in] long NumCycles,
                            [in] long IntervalBetweenCycles,
                            [in] long FileNameFormat,
                            [out, retval] long* ValidRequest);
```

Visual Basic Sample Code

```
Dim Mono1Start As Double
Dim Mono1Stop As Double
Dim Mono1Inc As Double
Dim Mono2Start As Double
Dim Mono2Stop As Double
Dim Mono2Inc As Double
Dim BackwardsExcite As Long
Dim ReadsPerPoint As Long
Dim NumCycles As Long
Dim IntervalBetweenCycles As Long
Dim FileNameFormat As Long

Mono1Start = 500
Mono1Stop = 600
Mono1Inc = 5
Mono2Start =300
Mono2Stop = 0
Mono2Inc = 0                  'Setting Mono2Inc to 0 moves Mono2 to the Mono2Start
BackwardsExcite = 0           'and leaves it there for the entire scan
```

```
ReadPerPoint = 1              ‘ This can not have a zero value. Only 1, 3, and 10 are valid
NumCycles = 2                 ‘ This can not have a zero value
IntervalBetweenCycles = 10000 ‘ Entering 0 here will cause the spectrometer to
FileNameFormat = 1            ‘ the start position and immediately start the next cycle.


‘With the above values, monochromator 1 will scan between 500 and 600(nm) with a
‘5nm increment between data points. The second monochromator will go to 300nm at the
‘the beginning of the scan and stay there. Only one integration will be taken at each point.
‘Two cycles(spectra)will be acquired with a 10 second delay between the end of the first
‘and the start of the second. The two files will be named “whatever01.___” and “whatever02.___”
if SpectaSense_COM.setNCLScanParams(  mono1Start,Mono1Stop,Mono1Inc,Mono2Start,Mono2Stop,
                                      mono2Inc,BackwardsExcite,ReadsPerPoint,NumCycles,
                                     IntervalBetweenCycles,FileNameFormat) <>0 then
   Label2.Caption = “Set up parameters loaded!”
End if
‘The same function could be written as:
If SpectraSense_COM.setNCLScanParams(500,600,5,300,0,0,0,1,2,10000,1)<>0 then
     MsgBox (“Set up Parameters”)
End if
```

Delphi Example Code

```
Var
Mono1Start: Integer;
Mono1Stop:Integer;
Mono1Inc: Integer;
Mono2Start: Integer;
Mono2Stop: Integer;
Mono2Inc: Integer;
BackwardsExcite: Integer
ReadPerPoint: Integer;               \\ This can not have a zero value
NumCycles: Integer;                  \\ This can not have a zero value
IntervalBetweenCycles: Integer;
FileNameFormat: Integer;


// load the scan parameters
if spectraSenseServer.setCCDMathParams( AreaOp,AreaOpNum,FileOp,
                                                 varFileName,CosmicCorrect,
                                                 DarkSubtract,SoftwareBin) = 0
// a zero result is failed
then showmessage(‘Failed to setup scan!’);
```

setNCLScanType

This function is used to set up spectral scanning parameters using the NCL and a single channel detector. Time based acquisition in not yet implemented in this function. For time based acquisition first create a routine in SpectraSense and save it. Then use the LoadRoutine function to set up the acquisition parameters.

Description: Sets up the type of single channel acquisition to be taken.

Parameters:

Scan_Value	0	No scan defined	
	1	Time Based	(Not implemented)
	2	Scan Mono1 Only	
	3	Scan Mono2 Only	
	4	Synchronized scan of mono1 and mono2	
	5	Excitation/emission scan: scan mono1 step mono2 after each scan	
	6	Excitation/emission scan: scan mono2 step mono1 after each scan	
Result:	0	an invalid request	
	!0	a valid request	

Definitions:

Delphi

```
Function  setNCLScanType(      Scan_Value: Integer
                             ): Integer; safecall;
```

Microsoft IDL

```
[id(0x00000035)]
HRESULT setNCLScanType(
                             [in] long Scan_Value,
                             [out, retval] long* ValidRequest);
```

Visual Basic Sample Code

```
If SpectraSense_COM.setNCLScanType(2)<>0 then
  Label2.Caption = "You're set up to scan just monochromator 1"
End if
```

Delphi Sample Code

```
If SpectraSense_COM.setNCLScanType(3) = 0
 then showmessage ('Set up failed')
```

setNCLShutterPosition

Description: Opens or closes the specified shutter. This function is valid only with an NCL shutter and will not work with a CCD

Parameters:

ShutterNum	The number of the shutter that is being controlled	
Open	0	close
	1	open
Result	0	An invalid request
	!0	A valid request

Note: Not valid with CCD shutters

Definitions:

Delphi

```
Function setNCLShutterPosition(   shutternum: Integer;
                                  Open: Integer): Integer; safecall;
```

Microsoft IDL

```
[id(0x00000042)]
HRESULT setNCLShutterPosition(
                                    [in] long ShutterNum,
                                    [in] long Open,
                                    [out, retval] long*
                                    ValidResult);
```

Visual Basic Sample Code

```
Dim ShutterNum As Long
Dim MoveShutter As Long

If SpectraSense_COM.setNCLShutterPosition(ShutterNum, MoveShutter) = 0 Then
   MsgBox (“Error : unable to set shutter position”)
   End If
```

Delphi Sample Code

```
var  ShutterNum  :  integer;
var  MoveShutter: integer;
// set the CCD ITime
if SpectraSenseServer.setNCLShutterPosition(Sutternumber, Moveshutter) = 0
   then showmessage(‘Error : unable to set shutter position);
```

Shutter_Present

Description: Determines if a shutter is present in the hardware configuration

Parameters:

ShutterNum	On systems without an NCL only 1 is valid. With an NCL in the system 1 and 2 are valid
Result:	0 Not Present !0 Present

Definitions:

Delphi

Function Shutter_Present(ShutterNum: Integer): integer; safecall;

Microsoft IDL

```
[id(0x00000020)]
HRESULT Shutter_Present(
        [in] long ShutterNum,
        [out, retval] long* Present);
```

Visual Basic Sample Code

```
‘ if the shutter is present, bring up the Shutter frame
 If SpectraSense_COM.Shutter_Present(1) <> 0 Then
    MsgBox (“shutter present”)
    End If
```

Delphi Sample Code

```
// if the Shutter is present, bring up the Shutter frame
if SpectraSenseServer.Shutter_Present(1) <> 0
   then label1.caption := ‘Shutter Present’;
```

SpectraSenseToBack

Description: Sends SpectraSense to the bottom of the desktop

Parameters:

Definitions:

Delphi

Procedure SpectraSenseToBack; safecall;

Microsoft IDL

[id(0x00000040)]
HRESULT SpectraSenseToBack ( ) ;

Visual Basic Sample Code

```
SpectraSense_COM.SpectraSenseToBack
```

Delphi Sample Code

```
SpectraSenseServer.SpectraSenseToBack;
```

SpectraSenseToFront

Description: Brings SpectraSense to the top of the desktop

Parameters:

Definitions:

Delphi

Procedure SpectraSenseToFront; safecall;

Microsoft IDL

[id(0x0000003e)]
HRESULT SpectraSenseToFront ( ) ;

Visual Basic Sample Code

```
SpectraSense_COM.SpectraSenseToFront
```

Delphi Sample Code

```
SpectraSenseServer.SpectraSenseToFront;
```

UpdateSpectraSense

This function is used to update the values on the SpectraSense software screens after changes have been initiated in your application. This function is especially useful in debugging your code as it will verify if you have correctly programmed the instrument state, or input values.

Description: Low level changes to SpectraSense are not always immediately translated to the SpectraSense screens. Run this function to resynchronized the screens to the current conditions.

Definitions:

Delphi

UpdateSpectraSense; safecall;

Microsoft IDL

[id(0x00000015), helpstring("Used to update the spectrasense software after low level changes")]
HRESULT UpdateSpectraSense();

Visual BasicSample Code

```
SpectraSense_COM.UpdateSpectraSense
```

Delphi Sample Code

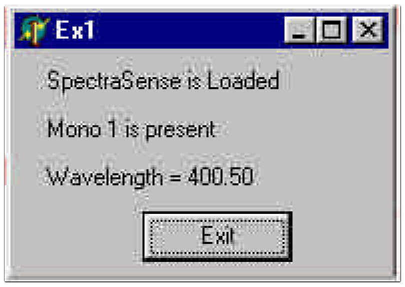
```
SpectraSenseServer.UpdateSpectraSense;
```

Example Programs

The SpectraSense SDK includes programing examples to aid you in developing your own applications. There are six examples that incorporate all of the data acquisition and instrument control functions. Sample code is provided in both Visual Basic and Delphi. These programs are located in the SpectraSense SDK sub directory. A separate program uses all of the data viewer functions.

This example loads SpectraSense software and reads the monochromator position.

Example 1

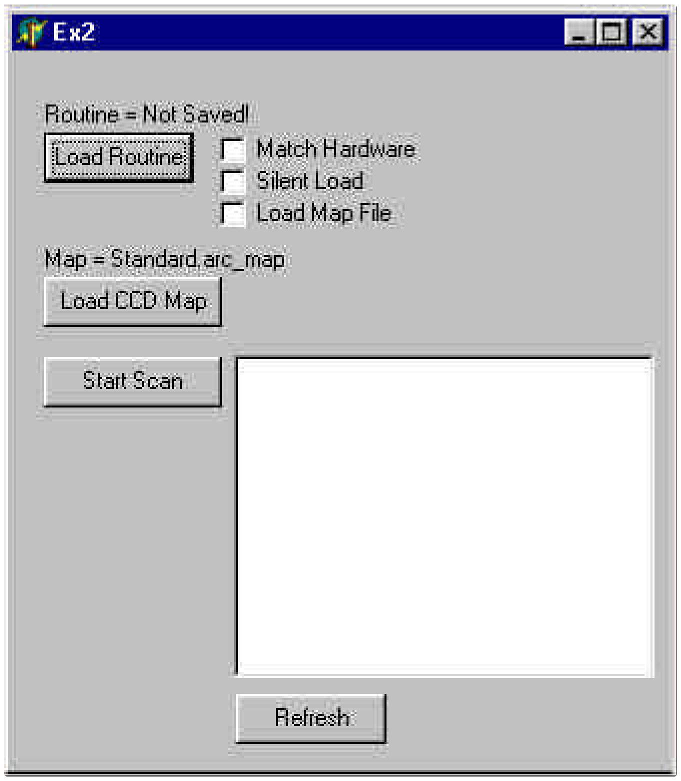


SpectraSense is Loaded

Mono 1 is present

Wavelength = 400.50

Example 2



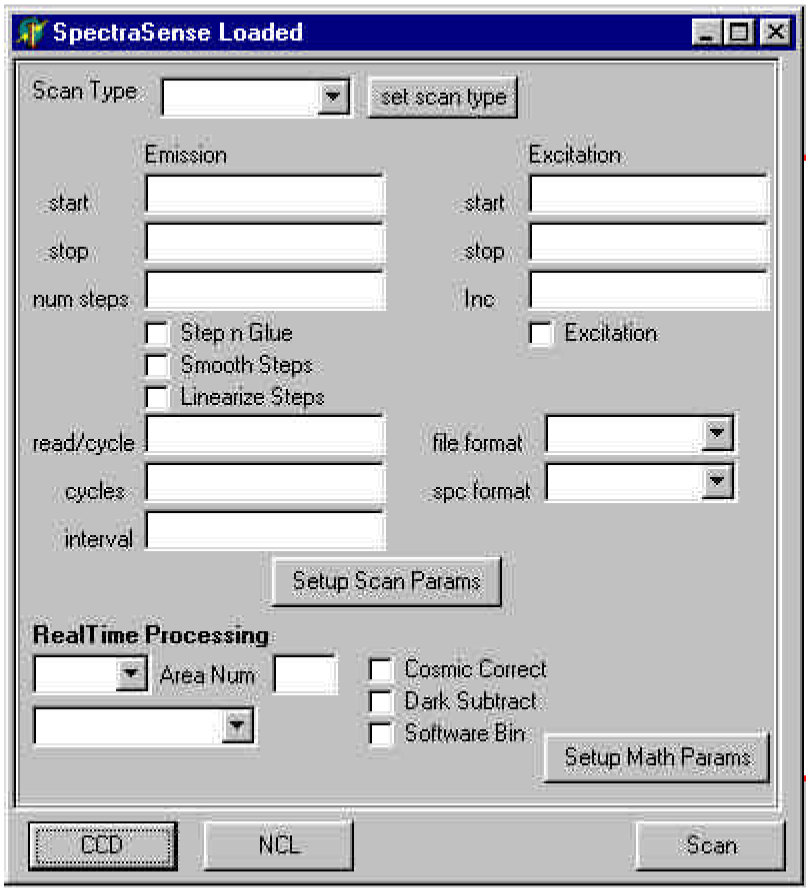
Routine = Not Saved!

Map = Standard.arc_map

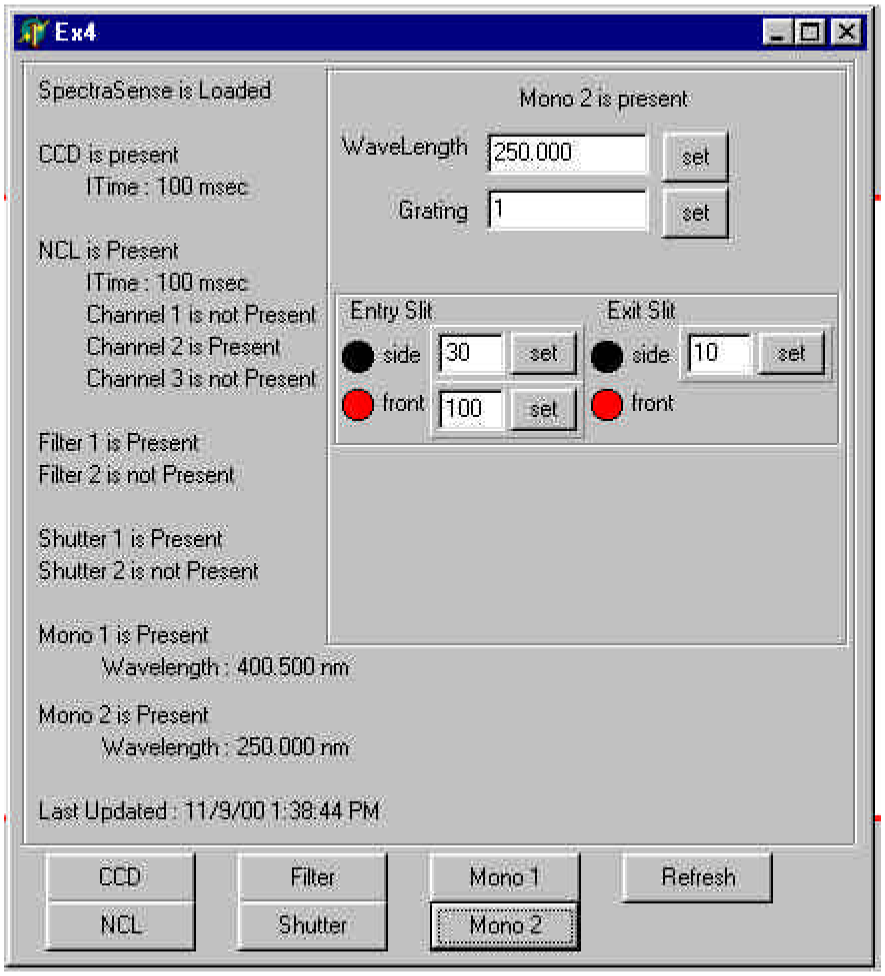
This example employs the higher level functions of loading acquisition routines and area maps. It then acquires spectra and lists the stored file names.

Example 3

This example employs the mid level functions for defining acquisition parameters outside the SpectraSense operating envirionment.



Example 4



Ex4

SpectraSense is Loaded

CCD is present
ITime : 100 msec

NCL is Present
ITime : 100 msec
Channel 1 is not Present
Channel 2 is Present
Channel 3 is not Present

Filter 1 is Present
Filter 2 is not Present

Shutter 1 is Present
Shutter 2 is not Present

Mono 1 is Present
Wavelength : 400.500 nm

Mono 2 is Present
Wavelength : 250.000 nm

Last Updated : 11/9/00 1:38:44 PM

Mono 2 is present

WaveLength: 250.000

Grating: 1

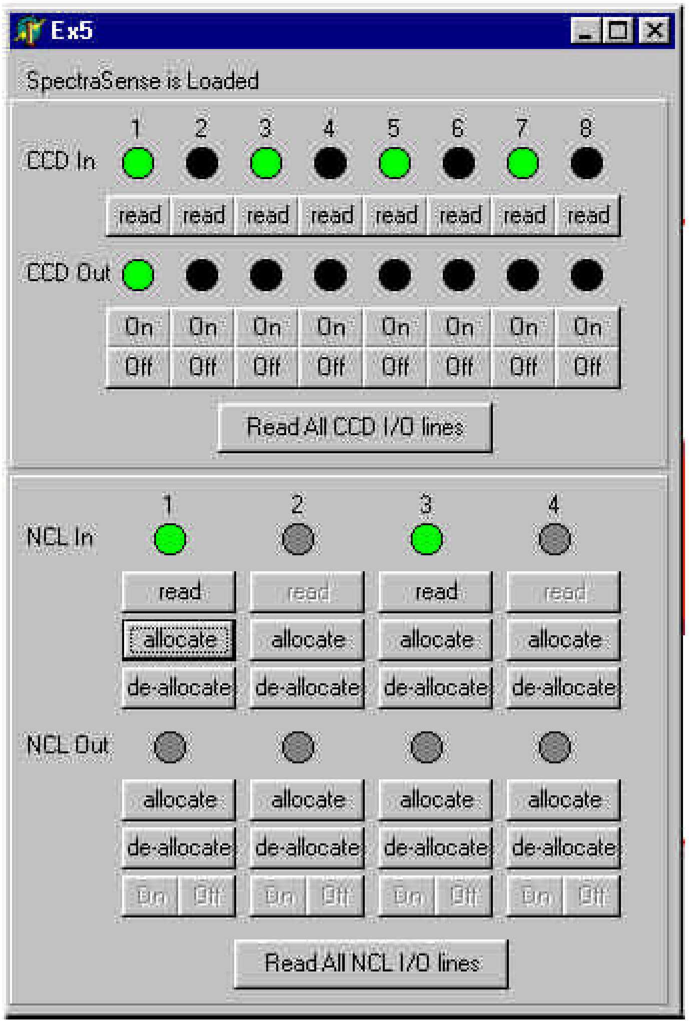
Entry Slit
side: 30
front: 100

Exit Slit
side: 10
front

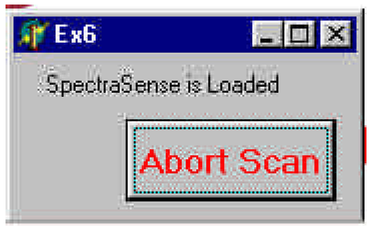
This example shows the code for reading and changing system parameters such as monochromator position, grating, and slit width.

Example 5

This example demonstrates how to access, read, and change the I/O lines on the CCD and NCL.



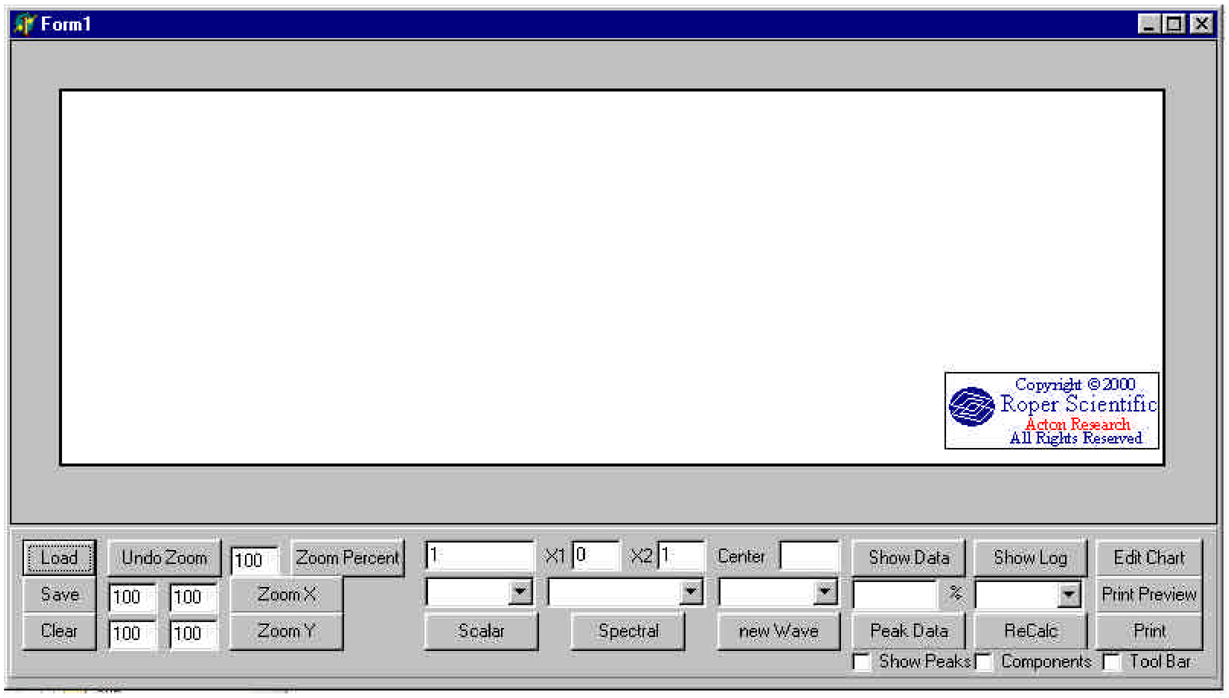
Example 6



This example demonstrates how to abort an acquisition.

Data Viewer

All of the functions for displaying and manipulating data are demonstrated in the data viewer sample program.



Index

The PDF version of this document has hyperlinks between the index entries and the subject pages