Instructions for MonoControl SDK
SpectraPro OLE COM Interface

The SpectraPro COM interface allows users to create software applications that interface with and control the software and hardware of the Acton Research Corporation SpectraPro monochromators through Microsoft’s COM/DCOM protocol. This interface is declared with GUID {999DF143-A7CF-4815-9BBF-D2E71E0929E5} and recognized by the system as ARCSpectraPro.SpectraPro_Interface. The interface is self registering on the invocation of SpectraPro.

The Interface is divided into two parts:

1. Program Control : Functions that control/modify how the software appears on the screen.
2. Hardware Control : Functions that control/modify the Acton hardware in the system.

A complete set of example programs written in Visual Basic, and Delphi are included on the SpectraPro installation disk.
**Program Control Functions**

1. **SetPageNum**
   - Brings the specified SpectraPro page to the front

2. **GetPageNum**
   - Returns the current SpectraPro screen

3. **IsSpectraProUp**
   - Returns SpectraPro loading status

4. **SpectraProToFront**
   - Brings SpectraPro to top of desktop

5. **SpectraProToBack**
   - Sends SpectraPro to bottom position on desktop

6. **SpectraProShow**
   - Restores SpectraPro to the Screen

7. **SpectraProHide**
   - Removes SpectraPro from the Screen

8. **getSpectraProMono**
   - When used with the NCL Data Acquisition Module, returns the number of the mono with which the COM interface is communicating (1 or 2).

9. **setSpectraProMono**
   - When used with the NCL Data Acquisition Module, sets the mono (1 or 2) with which the SpectraPro COM interface is communicating.

10. **UpdateSpectraPro**
    - Refreshes the SpectraPro screen with the latest information
SetPageNum

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

Parameters

PageNum: 0 About Screen  
         1 Main Page  
         2 Operation Page  
         3 Diverter Page  
         4 Motorized Slits Page  
         5 Install Gratings Page  
         6 Wave Offset Page  
         7 G Adjust Page  
         8 Monochromator Defaults Page  
         9 Calibration Info Page  
        10 Terminal Page  

Result: 0 An invalid request  
         !0 A valid request  

Definitions

Delphi

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

Microsoft IDL

[id(0x00000007)]
HRESULT SetPageNum(  
    [in] long Page,
    [out, retval] long* ValidRequest);  

Visual Basic Sample Code

Dim PageNum As Long  
If SpectraPro_COM.SelectPage(PageNum) = 0 then  
    MsgBox("Failed to select page")  
End If

Delphi Sample Code

Var
PageNum : Long;

Begin
SpectraProServer.SelectPage(PageNum) = 0;  
Then show message (’Failed to select page’);  
End;
getPageNum

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

Parameters

Result:
0 About Screen
1 Main Page
2 Operation Page
3 Diverter Page
4 Motorized Slits Page
5 Install Gratings Page
6 Wave Offset Page
7 G Adjust Page
8 Monochromator Defaults Page
9 Calibration Info Page
10 Terminal Page

Definitions

Delphi

Function GetPageNum: Integer; safecall;

Microsoft IDL

[id(0x00000008)]
HRESULT getPageNum([out, retval] long* Page);

Visual Basic Sample Code

Dim PageNum As Long
PageNum = SpectraPro_COM.getPageNum

Delphi Sample Code

Var
PageNum : Long;
Begin
PageNum := SpectraProServer.getPageNum;
End;
IsSpectraProUp

Definition: Tells if SpectraPro has finished loading

Parameters

Result: 

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>has not finished</th>
</tr>
</thead>
<tbody>
<tr>
<td>!0</td>
<td>has finished</td>
<td></td>
</tr>
</tbody>
</table>

Definitions

Delphi

Function IsSpectraProUp: Integer;safecall;

Microsoft IDL

[id(0x00000001)]
HRESULT IsSpectraProUp([out, retval] long* SpectraProState);

Visual Basic Sample Code

If SpectraPro_COM.IsSpectraProUp = 0 then
    Label1.caption = "SpectraPro is loading"
End if

Delphi Example Code

if SpectraProServer.IsSpectraProUp = 0
    then label.caption = 'SpectraPro is loading'
**SpectraProToFront**

**Description:** Brings SpectraPro to the top of the desktop

**Definitions**

**Delphi**

Procedure SpectraProToFront; safecall;

**Microsoft IDL**

[id(0x00000006)]

HRESULT SpectraProToFront();

**Visual Basic Sample Code**

SpectraPro_COM.SpectraProToFront

**Delphi Sample Code**

SpectraProServer.SpectraProToFront;
SpectraProToBack

Description: Sends SpectraPro to the bottom of the desktop

Definitions

Delphi
Procedure SpectraProToBack; safecall;

Microsoft IDL
[id(0x00000005)]
HRESULT SpectraProToBack();

Visual Basic Sample Code

SpectraPro_COM.SpectraProToBack

Delphi Sample Code

SpectraProServer.SpectraProToBack;
SpectraProShow

**Description:** Display’s SpectraPro on the Screen

**Definitions**

**Delphi**

Procedure SpectraProShow; safecall;

**Microsoft IDL**

[id(0x00000002)]

HRESULT SpectraProShow();

**Visual Basic Sample Code**

SpectraPro_COM.SpectraProShow

**Delphi Sample Code**

SpectraProServer.SpectraProShow;
SpectraProHide

Description: Remove SpectraPro from the screen

Definitions

Delphi

Procedure SpectraProShow; safecall;

Microsoft IDL

[id(0x00000003)]
HRESULT SpectraProHide();

Visual Basic Sample Code

SpectraPro_COM.SpectraProHide

Delphi Sample Code

SpectraProServer.SpectraProHide;
**getSpectraProMono**

**Description:** retrieves the number of the monochromator (1 or 2) with which the SpectraPro COM interface software is communicating.

**Parameters**

**Result:** The number of the monochromator with which the SpectraPro COM interface is communicating can be 1 or 2 if an NCL is present. Otherwise a 1 is returned.

**Definitions**

**Delphi**

```delphi
function  getSpectraProMono: Integer; safecall;
```

**Microsoft IDL**

```idl
[id(0x0000001b)]
HRESULT getSpectraProMono([out, retval] long* curMono);
```

**Visual Basic Sample Code**

```vbnet
Dim MonoNum as Long
MonoNum = SpectraPro_COM.getSpectraProMono
```

**Delphi Sample Code**

```delphi
Var MonoNum : integer;
MonoNum := SpectraProServer.getSpectraProMono;
```
setSpectraProMono

Description: Sets which Monohromator with which the SpectraPro COM interface software is communicating.

Parameters

NewMono: The number of the monochromator to be selected, 1 valid for singles mono’s, 1 and 2 valid for mono’s with NCL’s.

Result: 0, Operation failed.

!0, Operation succeeded.

Definitions

Delphi

function setSpectraProMono(newMono: Integer): Integer; safecall;

Microsoft IDL

[id(0x0000001c)]
HRESULT setSpectraProMono(
    [in] long newMono,
    [out, retval] long* ValidResult);

Visual Basic Sample Code

Dim MonoNum as Long

If SpectraPro_COM.setSpectraProMono(MonoNum) = 0 then
    MSGBox("Failed")
End if

Delphi Sample Code

Var MonoNum : integer;

If SpectraProServer.setSpectraProMono(MonoNum) <> 0
    Then showmessage('Failed');
**UpdateSpectraPro**

This function is used to update the values on the SpectraPro 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 SpectraPro are not always immediately translated to the SpectraPro screens. Run this function to resynchronize the screens to the current conditions.

**Definitions**

**Delphi**

Procedure UpdateSpectraPro; safecall;

**Microsoft IDL**

```
[id(0x00000004)]
HRESULT UpdateSpectraPro();
```

**Visual Basic Sample Code**

`SpectraPro_COM.UpdateSpectraPro`

**Delphi Sample Code**

`SpectraProServer.UpdateSpectraPro;`
SpectraPro OLE COM interface Hardware Control  Functions

Monochromators

1. Mono_Present  Tests to see if a monochromator is in the system
2. MonoDouble  Defines a monochromator as a double monochromator
3. getMonoWavelength  Returns a monochromator position in nm
4. setMonoWavelength  Sends the monochromator to a wavelength in nm
5. getMonoGrating  Returns the grating number in use
6. setMonoGrating  Change the grating to the number specified
7. getMonoTurret  Returns the turret number in use
8. setMonoTurret  Change the turret to the number specified
9. getMonoDiverterPosition  Returns the position of the active port
10. setMonoDiverterPosition  Change the diverter mirror position
11. MonoSlitMotorized  Tests to see if a slit is motorized or manual
12. getMonoSlitWidth  Returns the slit width of a motorized slit
13. setMonoSlitWidth  Set a motorized slit to a specific width
14. MonoSetupScan  Setup the parameters for scanning a Monochromator
15. getMonoScanParams  Find out the scan parameters for a Monochromator
16. MonoScanWavelength  Scan a Monochromator

NCL

1. NCL_Present

Filters

1. Filter_present  Test to see if a filter wheel is present
2. getFilterPosition  Returns the current filter position (1 – 6)
3. setFilterPosition  Moves the specified filter into position
4. FilterHome  Homes the filter wheel and sets it to position 1
Mono_Present

Description: Used to determine if a specific monochromator is in the system. When used with an NCL, use the parameters 1 or 2 to determine if one or both monochromators are present in the system. When used without an NCL, use input parameter 1 to determine if the monochromator is actively attached to the system.

Parameters

Mono_Num The number of the monochromator to which commands are addressed.

Note: If you 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 (1 or 2).

Result: 0 Not Present
         !0 Present

Definitions

Delphi

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

Microsoft IDL

[id(0x0000000a)]
HRESULT Mono_Present(
    [in] long MonoNum,
    [out, retval] long* MonoState);

Visual Basic Example Code

If SpectraPro_COM.Mono_Present(1) = 0 then
    Label1.caption = "Monochromator 1 not present"
End if

Delphi Example Code

if SpectraProServer.Mon_Present(2) = 0
    then label1.caption = 'Mono 2 not present';
MonoDouble

Description: Determines if a monochromator is a double or single

Parameters

Mono_Num: The number of the monochromator to which commands are addressed.

Note: If you 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 (1 or 2).

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

Definitions

Delphi

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

Microsoft IDL

[id(0x0000000c)]
HRESULT MonoDouble(
    [in] long MonoNum,
    [out, retval] long* DoubleState);

Visual Basic Sample Code

If SpectraPro_COM.MonoDouble(1) = 0 then
    Label1.caption = "Monochromator 1 is not a double"
End if

Delphi Sample Code

if SpectraProServer.MonoDouble(2) <> 0
    then Showmessage ('Monochromator 2 is a double');
**getMonoWavelength**

**Description:**
This function will return the position of the specified monochromator in nm.

**Parameters**

**Mono_Num:**
The number of the monochromator being interrogated (see note below).

**Result:**
The monochromator position in nm.

**Definitions**

**Delphi**

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

**Microsoft IDL**

[id(0x0000000d)]
HRESULT getMonoWaveLength(
    [in] long MonoNum,
    [out, retval] double* CurWave);

**Note:**
If you 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 (1 or 2).

**Visual Basic Sample Code**

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

**Delphi Sample Code**

```delphi
Var MonoWavelength : double
MonoWavelength := SpectraProServer.getMonoWavelength(1);
Label1.caption := 'Wavelength : ' + format('%.3f',[TempDbl]) + ' nm';
```
**setMonoWaveLength**

**Description:** This function will change the position of the specified monochromator.

**Parameters**

**Mono_Num:** The number of the monochromator to which commands are addressed. **Note:** If you 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 (1 or 2).

**NewWavelength:** Enter the new position for the specified monochromator in NM. **Note:** NewWavelength may be integer or decimal values.

**Result:**
- 0 An invalid request
- !0 A valid request

**Definitions**

**Delphi**

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

**Microsoft IDL**

```idl
[id(0x0000000e)]
HRESULT setMonoWaveLength(
    [in] long MonoNum,  
    [in] double newWave,  
    [out, retval] long* ValidRequest);
```

**Visual Basic Sample Code**

```vbnet
Dim newWave As Double
Dim curmono As Double
' set the wavelength
Curmono = 1
newWave = 500.0
If SpectraPro_COM.setMonoWavelength(curmono, newWave) = 0 Then
    MsgBox ("Error : Failed to set wavelength")
End If
```
Delphi Example Code

```delphi
var newWave : double;
var curmon : double;
// set the wavelength
newWave := 500.0;
curmon := 1;

if SpectraProServer.setMonoWavelength(curmon,newWave) = 0;
  then ShowMessage('Error : Failed to set wavelength');
```
getMonoGrating

Description: Returns the currently selected grating number

Parameters

Mono_Num: The number of the monochromator to which commands are addressed. 
Note: If you 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 (1 or 2).

Result: The current grating number in the specified monochromator

Definitions

Delphi

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

Microsoft IDL

[id(0x0000000f)]
HRESULT getMonoGrating(
    [in] long MonoNum,
    [out, retval] long* curGrating);

Visual Basic Sample Code

Dim newGrat As Integer

newGrat = SpectraPro COM.getMonoGrating(1)
Label1.caption = newGrat

Delphi Sample Code

var GratingNumber : integer;

GratingNumber := SpectraProServer.getMonoGrating(1)
ShowMessage('Mono 1 grating is : ' +inttoStr(grating number));
setMonoGrating

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

Parameters

Mono_Num: The number of the monochromator to which commands are addressed.

Note: If you are not using an NCL then use monochromator 1. When using an NCL use the number of the connection between the monochromator and NCL (1 or 2).

New Grating: The grating number to be put in place

Result:
0 An invalid request
!0 A valid request

Definitions

Delphi

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

Microsoft IDL

[id(0x00000010)]
HRESULT setMonoGrating(
[in] long MonoNum,
[in] long newGrating,
[out, retval] long* ValidRequest);

Visual Basic Sample Code

Dim newGrat As Long
newGrat = 3
'set the grating
If SpectraPro_COM.setMonoGrating(1, newGrat) = 0 Then
MsgBox("Error : Failed to move grating")
End If

Delphi Sample Code

var newGrat : integer;
newGrat := 2;
//set the grating
if SpectraProServer.setMonoGrating(1,newGrat) = 0
then ShowMessage('Error : Failed to move grating');
getMonoTurret

Description: Returns the currently selected grating turret number

Parameters

Mono_Num: The number of the monochromator to which commands are addressed.

Note: If you 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 (1 or 2).

Result: The current turret number in the specified monochromator

Definitions

Delphi

Function getMonoTurret(Mono_Num:integer;
): Integer; safecall;

Microsoft IDL

[id(0x00000011)]
HRESULT getMonoTurret(
[in] long MonoNum,
[out, retval] long* curTur);

Visual Basic Sample Code

Dim newTur As Integer
newTur = SpectraPro_COM.getMonoTurret(1)
Label1.caption = newTur

Delphi Sample Code

var TurretNumber : integer;
TurretNumber := SpectraProServer.getMonoTurret(1)
ShowMessage('Mono 1 turret is : ' +inttoStr(TurretNumber));
setMonoTurret

Description: selects the specified turret for the monochromator

Parameters

Mono_Num: The turret number to be selected
New Grating: The turret number to be selected

Result:
0 An invalid request
!0 A valid request

Definitions

Delphi
Function setMonoTurret( Mono_Num: Integer;
NewTurret: Integer
): Integer; safecall;

Microsoft IDL
[id(0x00000012)]
HRESULT setMonoTurret(
   [in] long MonoNum,
    [in] long newTur,
    [out, retval] long* ValidRequest);

Visual Basic Sample Code

Dim newTur As Long
newTur = 3
' set the turret
If SpectraPro_COM.setMonoGrating(1, newTur) = 0 Then
    MsgBox("Error : Failed to change turret")
End If

Delphi Sample Code

var newTurret : integer;
newTurret := 2;
// set the grating
if SpectraProServer.setMonoGrating(1,newTurret) = 0
then ShowMessage('Error : Failed to change turret');
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 to which commands are addressed. Note: If you are not using an NCL then use monochromator 1. When using an NCL use the number of the connection between the monochromator and NCL (1 or 2).

Divert_Num: 1 Entrance mirror 2 Exit mirror 3 Slave Entry Mirror (double monochromator) 4 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

Definitions

Delphi

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

Microsoft IDL

[id(0x00000013)]
HRESULT getMonoDiverterPosition(
[in] long MonoNum,
[in] long Divert_Num,
[out, retval] long* curSlit);

Visual Basic Sample Code

Dim curmono As Long
If SpectraPro_COM.setMonoDiverterPosition(curmono, 1) = 1 Then
  Label1.Caption = "The side entrance slit is active"
End If
Delphi Sample Code

// entrance position, 1 - side, 2 - front
If SpectraProServer.getMonoDiverterPosition(curmono,1)
  Then showmessage ('Side Entrance')
  else showmessage ('Front Entrance');
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 to which commands are addressed.

**Note:** If you are not using an NCL then use monochromator 1. When using an NCL use the number of the connection between the monochromator and NCL (1 or 2).

**Divert_Num:**

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

**Definitions**

**Delphi**

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

**Microsoft IDL**

```idl
[id(0x00000014)]
HRESULT setMonoDiverterPosition(
    [in] long MonoNum,
    [in] long Divert_Num,
    [in] long NewSlit,
    [out, retval] long* ValidRequest);
```
Visual Basic Sample Code

If SpectraPro_COM.setMonoDiverterPosition(1, 1, 1) = 0 Then
    MsgBox ("Error : Failed to change divertor")
End If

Delphi Sample Code

var curmono : Integer

// flip entrance diverter
if SpectraProServer.setMonoDiverterPosition(curmono, 1, 1) = 0
    then ShowMessage('Error : Failed to change diverter');
MonoSlitMotorized

Description: Determines if a particular slit is manual or motorized

Parameters

Mono_Num: The number of the monochromator to which commands are addressed.
Note: If you are not using an NCL then use monochromator 1. When using an NCL use the number of the connection between the monochromator and NCL (1 or 2).

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 An invalid request
!0 A valid request

Definitions

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

Microsoft IDL
[id(0x00000015)]
HRESULT MonoSlitMotorized(
[in] long MonoNum,
[in] long SlitNum,
[out, retval] long* SlitState);

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

Visual Basic Sample Code
Dim curmono As Long
If SpectraPro_COM.MonoSlitMotorized(curmono, 1) <> 0 Then
    MsgBox ("Side Entrance Slit is Motorized")
End if

Delphi Sample Code

// side entrance slit, slit #1

SpectraPro COM Interface
if SpectraProServer.MonoSlitMotorized(curmono,1) <> 0
    then showmessage ('Side Entrance Slit is Motorized')
else showmessage ('Side Entrance Slit is not Motorized');
getMonoSlitWidth

Description: Returns the slit width in microns only.

Parameter

Mono_Num: The number of the monochromator to which commands are addressed.

Note: If you 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 (1 or 2).

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

Definitions

Delphi

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

Microsoft IDL

[id(0x00000016)]
HRESULT getMonoSlitWidth(
    [in] long MonoNum,
    [in] long SlitNum,
    [out, retval] long* SlitWidth);

Visual Basic Sample Code

Labell.Caption = SpectraPro_COM.getMonoSlitWidth(curmono, 1)

Delphi Sample Code

Labell.caption := IntToStr(SpectraProServer.getMonoSlitWidth(curmono,1));
setMonoSlitWidth

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

Parameters

Mono_Num: The number of the monochromator to which commands are addressed.
Note: If you are not using an NCL then use monochromator 1. When using an NCL use the number of the connection between the monochromator and NCL (1 or 2).

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

Definitions

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

Microsoft IDL
[id(0x00000017)] HRESULT setMonoSlitWidth(
[in] long MonoNum,
[in] long SlitNum,
[in] long newWidth,
[out, retval] long* ValidRequest);

Visual Basic Sample Code
If SpectraPro_COM.setMonoSlitWidth(curmono, 1, newWidth) = 0 Then
    MsgBox ("Error : Failed to Set Slit 1 Width")
End if

Delphi Sample Code
if SpectraProServer.setMonoSlitWidth(curmono,1,newWidth) = 0
    then ShowMessage('Error : Failed to Set Slit 1 Width');
MonoSetupScan

**Description:** Setup the parameters for scanning a monochromator at a fixed rate.

**Parameters**

**Mono_Num:** The number of the monochromator to which commands are addressed.

*Note:* If you are not using an NCL then use monochromator 1. When using an NCL use the number of the connection between the monochromator and NCL (1 or 2).

**StartWave:** Wavelength to start the scan at

**StopWave:** Wavelength to end the scan at

**ScanRate:** Rate in nm per minute to scan at

**NumScans:** Number of times the scan is to be repeated

**SecDelay:** Delay in seconds between repeating scans

**Result:**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>An invalid request</td>
</tr>
<tr>
<td>10</td>
<td>A valid request</td>
</tr>
</tbody>
</table>

**Definitions**

**Delphi**

```delphi
Function MonoSetupScan(   MonoNum: Integer;
                          StartWave: Double;
                          StopWave: Double;
                          ScanRate: Double;
                          NumScans: Integer;
                          SecDelay: Integer): Integer; safecall;
```

**Microsoft IDL**

```idl
[id(0x0000001f)]
HRESULT MonoSetupScan(
    [in] long MonoNum,
    [in] double StartWave,
    [in] double StopWave,
    [in] double ScanRate,
    [in] long NumScans,
    [in] long SecDelay,
    [out, retval] long* ValidRequest);
```
**Visual Basic Sample Code**

Dim newStart As Double
Dim newStop As Double
Dim newRate As Double
Dim newNum As Long
Dim newDelay As Long

' set the values
If SpectraPro_COM.MonoSetupScan(curmono, newStart, newStop, newRate, newNum, newDelay) = 0 Then
    MsgBox("Error : Failed to set Params")
End If

**Delphi Sample Code**

var
    newStart, newStop, newRate : double;
    newNum, newDelay : integer;

// set the values
if SpectraProServer.MonoSetupScan(CurMono, newStart, newStop, newRate, newNum, newDelay) = 0 then
    showmessage('Error : Failed to set Params');
getMonoScanParams

Description: Retrieve the current scan parameters of the selected monochromator

Parameters

Mono_Num: The number of the monohchromator to which commands are addressed.
Note: If you are not using an NCL then use monochromator 1. When using an NCL use the number of the connection between the monochromator and NCL (1 or 2).

StartWave: Wavelength to start the scan at
StopWave: Wavelength to end the scan at
ScanRate: Rate in nm per minute to scan at
NumScans: Number of times the scan is to be repeated
SecDelay: Delay in seconds between repeating scans

Result:
0 An invalid request
!0 A valid request

Definitions

Delphi
Function getMonoScanParams( MonoNum: Integer;
    out StartWave: Double;
    out StopWave: Double;
    out ScanRate: Double;
    out NumScans: Integer;
    out SecDelay: Integer): Integer; safecall;

Microsoft IDL
[id(0x0000001d)]
HRESULT getMonoScanParams(
    [in] long MonoNum,
    [out] double* StartWave,
    [out] double* StopWave,
    [out] double* ScanRate,
    [out] long* NumScans,
    [out] long* SecDelay,
    [out, retval] long* ValidRequest);
Visual Basic Sample Code

Dim newStart As Double
Dim newStop As Double
Dim newRate As Double
Dim newNum As Long
Dim newDelay As Long

If SpectraPro_COM.getMonoScanParams(curmono,newStart,newStop,newRate,newNum,newDelay) <> 0 Then
    ScanStartEdt.Text = newStart
    ScanStopEdt.Text = newStop
    ScanRateEdt.Text = newRate
    ScanNumEdt.Text = newNum
    ScanDelayEdt.Text = newDelay
End if

Delphi Sample Code

var
newStart,newStop,newRate : double;
newNum,newDelay : integer;

if SpectraProServer.getMonoScanParams(curMono,newStart,newStop,newRate,newNum,newDelay) <> 0 then begin
    ScanStartEdt.Text := format('%.3f', [newStart]);
    ScanStopEdt.Text := format('%.3f', [newStop]);
    ScanRateEdt.Text := format('%.3f', [newRate]);
    ScanNumEdt.Text := IntToStr(newNum);
    ScanDelayEdt.Text := IntToStr(newDelay);
End;
MonoScanWavelength

**Description:** Scan a monochromator between two wavelengths, based on *MonoSetupScan* Values

**Parameters**

**Mono_Num:** The number of the monochromator to which commands are addressed.

**Note:** If you are not using an NCL then use monochromator 1. When using an NCL use the number of the connection between the monochromator and NCL (1 or 2).

**Result:**
- 0: An invalid request
- 10: A valid request

**Definitions**

**Delphi**

Function MonoScanWavelength(MonoNum: Integer): Integer; safecall;

**Microsoft IDL**

```idl
[id(0x00000020)]
HRESULT MonoScanWavelength(
    [in] long MonoNum,
    [out, retval] long* ValidResult);
```

**Note 2:** MonoSetupScan should be called at least once before invoking this function

**Visual Basic Sample Code**

```vbnet
If SpectraPro_COM.MonoScanWavelength(curmono) = 0 Then
    MsgBox ("Error : Failed to Scan")
End If
```

**Delphi Sample Code**

```delphi
if SpectraProServer.MonoScanWavelength(curMono) = 0 then showmessage('Error : Failed to Scan');
```
NCL_Present

Description: Determines if an NCL is in the system

Parameters

Result: 

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No NCL present</td>
</tr>
<tr>
<td>!0</td>
<td>NCL present</td>
</tr>
</tbody>
</table>

Definitions

Delphi

Function NCL_Present: Integer; safecall;

Microsoft IDL

[id(0x00000009)]
HRESULT NCL_Present([out, retval] long* NCLState);

Visual Basic Sample Code

If SpectraPro_COM.NCL_Present = 0 Then
   Label1.Caption = "NCL is not Present"
End if

Delphi Sample Code

// display some NCL Info
if SpectraProServer.NCL_Present = 0
   then Label1.caption := 'NCL 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:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>An invalid request</td>
</tr>
<tr>
<td>!0</td>
<td>A valid request</td>
</tr>
</tbody>
</table>

**Definitions**

**Delphi**

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

**Microsoft IDL**

[id(0x0000000b)]
HRESULT Filter_Present(
    [in] long FilterNum,
    [out, retval] long* FilterState);

**Visual Basic Sample Code**

If SpectraPro_COM.Filter_Present <> 0 Then
    Label1.caption = "Filter Present"
End If

**Delphi Sample Code**

if SpectraProServer.Filter_Present <> 0
    then Label1.caption := 'Filter Present';
**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**

```delphi
delphi function getFilterPosition(Filter_Num: Integer): Integer;
safecall;
:
```

**Microsoft IDL**

```
[id(0x00000018)]
HRESULT getFilterPosition(
    [in] long FilterNum,
    [out, retval] long* curFilter);
```

**Visual Basic Sample Code**

```vbnet
If SpectraPro_COM.getFilterPosition(1) = 6 Then
    Label1.caption = "Filter number 6 is in place"
End if
```

**Delphi Sample Code**

```delphi
if SpectraProServer.getFilterPosition(1) = 2
then Label1.caption := 'filter number 2';
```
**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(0x00000019)]
HRESULT setFilterPosition(
    [in] long FilterNum,
    [in] long newPosition,
    [out, retval] long* ValidRequest);

**Visual Basic Sample Code**

If SpectraPro_COM.setFilterPosition(1, newFilterPos) = 0 Then
    MsgBox ("Error : Failed to set filter position")
End If

**Dephi Sample Code**

if SpectraProServer.setFilterPosition(1,newFilterPos) = 0
    then showmessage('Error : Failed to set filter position');
**FilterHome**

**Description:** Centers the filter on the home (Filter 1) position.

**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 FilterHome(Filter_Num: Integer): Integer; safecall;

**Microsoft IDL**

[id(0x0000001a)]
HRESULT FilterHome(
    [in] long FilterNum,
    [out, retval] long* ValidRequest);

**Visual Basic Sample Code**

If SpectraPro_COM.Filter_Present <> 0 Then
    MsgBox ("Homed the Filter wheel")
End If

**Delphi Sample Code**

if SpectraProServer.Filter_Present <> 0
    then showmessage('Homed the Filter wheel');