Software Development Kit (SDK)

BASED ON THE DEVICE CONTROL DYNAMIC-LINK LIBRARY

DEVCTRL.DLL

DELPHI DEVELOPER MANUAL

Version 4.0

December 11, 2008

Mensk, BELARUS

Introduction

This Manual provides a brief description of all functions and procedures exported by Device Control DLL (DEVCTRL.DLL) software. These routines are used to control and communicate with a Solar TII spectral device. In all cases the **stdcall** calling convention is used. The routines are listed in the order used in the **exports** clause of the software, i.e., the routine described in Sect. 1 has index of 1 and so on. Hereafter the notions of "spectral instrument" and "spectral device" (or simply "device" for short) will be used interchangeably.

Software versions covered: 1.26.0.0 and higher.

Control panel constants

Constant	Expression	Value
PANEL_ALL	0	0
PANEL_GRATING1	int64(1) shl 0	1
PANEL_GRATING2	int64(1) shl 1	2
PANEL_GRATING3	int64(1) shl 2	4
PANEL_GRATING4	int64(1) shl 3	8
PANEL_SLIT1	int64(1) shl 4	16
PANEL_SLIT2	int64(1) shl 5	32
PANEL_SLIT3	int64(1) shl 6	64
PANEL_SLIT4	int64(1) shl 7	128
PANEL_TURNSLIT1	int64(1) shl 8	256
PANEL_TURNSLIT2	int64(1) shl 9	512
PANEL_TURRET1	int64(1) shl 10	1 024
PANEL_TURRET2	int64(1) shl 11	2 048
PANEL_SHUTTER1	int64(1) shl 12	4 096
PANEL_SHUTTER2	int64(1) shl 13	8 192
PANEL_SHUTTER3	int64(1) shl 14	16 384
PANEL_SHUTTER4	int64(1) shl 15	32 768
PANEL_FILTER1	int64(1) shl 16	65 536
PANEL_FILTER2	int64(1) shl 17	131 072
PANEL_FILTER3	int64(1) shl 18	262 144
PANEL_FILTER4	int64(1) shl 19	524 288
PANEL_MIRROR1	int64(1) shl 20	1 048 576
PANEL_MIRROR2	int64(1) shl 21	2 097 152
PANEL_MIRROR3	int64(1) shl 22	4 194 304
PANEL_MIRROR4	int64(1) shl 23	8 388 608
PANEL_MULTIPOS1	int64(1) shl 26	67 108 864
PANEL_MULTIPOS2	int64(1) shl 27	134 217 728
PANEL_MULTIPOS3	int64(1) shl 28	268 435 456
PANEL_MULTIPOS4	int64(1) shl 29	536 870 912
PANEL_DIGITPOS1	int64(1) shl 31	2 147 483 648
PANEL_DIGITPOS2	int64(1) shl 32	4 294 967 296
PANEL_DIGITPOS3	int64(1) shl 33	8 589 934 592
PANEL_DIGITPOS4	int64(1) shl 34	17 179 869 184
PANEL_DIGITPOS5	int64(1) shl 35	34 359 738 368
PANEL_DIGITPOS6	int64(1) shl 36	68 719 476 736
PANEL_DIGITPOS7	int64(1) shl 37	137 438 953 472
PANEL_DIGITPOS8	int64(1) shl 38	274 877 906 944
PANEL_BISTABLE1	int64(1) shl 39	549 755 813 888
PANEL_BISTABLE2	int64(1) shl 40	1 099 511 627 776
PANEL_QUERY1	int64(1) shl 43	8 796 093 022 208
PANEL_QUERY2	int64(1) shl 44	17 592 186 044 416
PANEL_QUERY3	int64(1) shl 45	35 184 372 088 832
PANEL_QUERY4	int64(1) shl 46	70 368 744 177 664
PANEL_DIGITPOSEX1	int64(1) shl 47	140 737 488 355 328
PANEL_DIGITPOSEX2	int64(1) shl 48	281 474 976 710 656
PANEL_DIGITPOSEX3	int64(1) shl 49	562 949 953 421 312
PANEL_DIGITPOSEX4	int64(1) shl 50	1 125 899 906 842 624

Error codes

Constant	Value
ALL_OK	0
RECEIVING_DATA	1
PREVIOUSPOS_NEED	2
DEVICE_NOTFOUND	6
ERROR_OVERFULL	7
ERROR_SYMBOLFAULT	8
ERROR_OVERTIME	9
ERROR_INTERFACE	10
ERROR_HARDWARE	11
ERROR_STEPMOTOR	12
ERROR_UNKNOWNSYMBOL	13
ERROR_UNKNOWNERROR	14
ERROR_READTIMEOUT	15
ERROR_WRITETIMEOUT	16
ERROR_RECEIVEDCODE	17
ERROR_DEVICENOTREADY	18
ERROR_NOTCONNECT	19
ERROR_INITIALISATION	20
ERROR_UNKNOWNPORT	21
ERROR_CONVERSION	22
ERROR_INCORRECTPARAM	23
ERROR_UNKNOWNDEVICE	24
ERROR_RANGE	25
ERROR_RESET	26
ERROR_MEMORY	27
ERROR_FILENOTFOUND	28
ERROR_CONFIGFILEREAD	29
ERROR_CONFIGFILEWRITE	30
ERROR_CONFIGPARAM	31
ERROR_CONFIGFILE	32
ERROR_FILTER_RANGE	37
DEMO_MODE	38
ONEOFDEVICE_NOTFOUND	39
ERROR_SERIALNUM	40
ERROR_BLOCKEDCONTROL	41
ERROR_BACKUPFILEWRITE	42
ERROR_OPENCOVER	43
ERROR_2RDBYTE	44
ERROR_BUSY	45

Other constants

Constant	Value
STATE_OFF	-1
STATE_ON	1
FC_AUTO	0
FC_MANUAL	1
INFO_TIME	1
INFO_DIFFS	2
SHUTTER_OPEN	3
SHUTTER_CLOSE	4
SDT_GRATING	1
SDT_TURRET	2
SDT_SLIT	3
SDT_TURNSLIT	4
SDT_SHUTTER	5
SDT_MIRROR	6
SDT_FILTER	7
SDT_DOUBLEFILTER	8
SDT_MULTIPOS	9
SDT_DIGITPOS	10
SDT_BISTABLEDEVICE	11
SDT_QUERYDEVICE	12
SDT_DIGITPOSEX	13

Contents

Exported Routines

1 InitDevice

```
function InitDevice(hAppWnd:  HWND): boolean;
```

Function **InitDevice** is **obsolete** and retained for backward compatibility.
Replaced by **InitDeviceEx2(hAppWnd, 0, nil, True)**.

2 FreeDevice

```
procedure FreeDevice;
```

Procedure **FreeDevice** is used to free active device. This routine closes the used COM-port and saves all the infos on the device state into the corresponding ***.cfg** file.

Parameters: None.

Usage example(s): **FreeDevice;**

3 ShowDeviceWindow

```
procedure ShowDeviceWindow(Panel:  longint);
```

Procedure **ShowDeviceWindow** is **obsolete** and retained for backward compatibility.
Replaced by **ShowDeviceWindowEx(Panel)**.

4 OpenShutter

```
function OpenShutter(Panel:  longint = PANEL_SHUTTER1; NotIsActive:  boolean =
True):  boolean;
```

Function **OpenShutter** is used to open a shutter.

Parameters:
Panel is the shutter ID (valid values range from **PANEL_SHUTTER1** to **PANEL_SHUTTER4**). Can be omitted; the default value is **PANEL_SHUTTER1**.

If **NotIsActive** is **True** then routine tries to open the so-called "active" shutter (e.g., feature of MSDD1000 device line) even if **Panel** refers to another shutter. Can be omitted; the default value is **True**.

Result: **True** if operation is started, **False** otherwise.

Usage example(s): **OpenShutter; OpenShutter(PANEL_SHUTTER1, False)**.

Remark: Use function **IsOperationFinished** to check whether the operation is still in progress or already finished. Use function **GetDeviceError** to check whether any error is encountered.

5 CloseShutter

```
function CloseShutter(Panel:  longint = PANEL_SHUTTER1; NotIsActive:  boolean =
True):  boolean;
```

Function **CloseShutter** is used to close a shutter.

Parameters:
Panel is the shutter ID (valid values range from **PANEL_SHUTTER1** to **PANEL_SHUTTER4**). Can be omitted; the default value is **PANEL_SHUTTER1**.

If **NotIsActive** is **True** then routine tries to close the so-called "active" shutter (e.g., feature of MSDD1000 device line) even if **Panel** refers to another shutter. Can be omitted; the default value is **True**.

Result: `True` if operation is started, `False` otherwise.

Usage example(s): `CloseShutter; CloseShutter(PANEL_SHUTTER1, False)`.

Remark: Use function `IsOperationFinished` to check whether the operation is still in progress or already finished. Use function `GetDeviceError` to check whether any error is encountered.

6 GetShutterState

```
function GetShutterState(Panel:  longint = PANEL_SHUTTER1):  byte;
```

Function `GetShutterState` is used to query the shutter state.

Parameters:
`Panel` is the shutter ID (valid values range from `PANEL_SHUTTER1` to `PANEL_SHUTTER4`). Can be omitted; the default value is `PANEL_SHUTTER1`.

Result: `SHUTTER_OPEN` if the software is in demo mode or device is not yet initialised; `ERROR_INCORRECTPARAM` if `Panel` is out of valid range; and for normal operation `SHUTTER_OPEN`/`SHUTTER_CLOSE` if the shutter is open/closed, respectively.

Usage example(s): `if GetShutterState <> SHUTTER_OPEN then...`

7 GetSupportedDevicesCount

```
function GetSupportedDevicesCount:  integer;
```

Function `GetSupportedDevicesCount` is used to retrieve the number of different spectral instruments supported by the software.

Parameters: None.

Result: Number of different spectral instruments supported by the software.

Usage example(s): `YourVar:= GetSupportedDevicesCount;`

8 GetActiveDevice

```
function GetActiveDevice:  integer;
```

Function `GetActiveDevice` is used to retrieve the index of the type of active spectral instrument.

Parameters: None.

Result:

Index	Type	Index	Type	Index	Type
0	S3804	7	MS2704	14	MS3501
1	NP2502	8	MS750MSE	15	MSDD1000a
2	MSDD1000	9	LPS_S380	16	MS2004
3	MS3504	10	TELTUBE	17	MS2001
4	MS7504	11	SolarJS	18	DM160
5	M1601	12	VPMSL40212		
6	MSE750	13	SLM4800		

Usage example(s): `YourVar:= GetActiveDevice;`

9 GetDeviceName

```
procedure GetDeviceName(DeviceIndex:  integer; DeviceName:  PChar);
```

Procedure `GetDeviceName` is used to retrieve the type of a spectral instrument by its index.

Parameters:
`DeviceIndex` is the index of device type. Can be retrieved by using function `GetActiveDevice` (see Sect. 8).
`DeviceName` contains the name of device type if successful, or an empty string otherwise.

Usage example(s): `GetDeviceName(GetActiveDevice, YourVar);`

10 SetPermissionMessageBox

`procedure SetPermissionMessageBox(IsPermitted: boolean);`

Procedure `SetPermissionMessageBox` is used to permit or forbid displaying error messages.

Parameters:

If `IsPermitted` is `True` then error messages will be displayed, otherwise not.

Usage example(s): `SetPermissionMessageBox(True);`

Remark: By default (i.e., before the very first call of this routine), the error messages are displayed.

11 IsOperationFinished

`function IsOperationFinished: boolean;`

Function `IsOperationFinished` is used to check whether the current operation (e.g., changing grating, wavelength, slit width, etc.) is still in progress or already finished.

Parameters: None.

Result: `False` if current operation is still in progress, `True` otherwise.

Usage example(s): `while not IsOperationFinished do Application.ProcessMessages;`

12 GetDeviceError

`function GetDeviceError: byte;`

Function `GetDeviceError` is used to check whether any error encountered during the last operation.

Parameters: None.

Result: `ERROR_INITIALISATION` if device is not yet initialised, `ALL_OK` if everything is OK (no error), or specified error code otherwise (see Error codes for details).

Usage example(s): `YourVar:= GetDeviceError; if GetDeviceError = ALL_OK then...`

13 WND_MSG_Register

`procedure WND_MSG_Register(hWindow: HWND; MsgID: UINT);`

Procedure `WND_MSG_Register` is used to register a spesific message and a destination window.

Parameters:
`hWindow` is the handle of the destination window.

`MsgID` is the message ID.

Usage example(s): `WND_MSG_Register(Application.MainWindow.Handle, YourVar);`

14 GetWaveLength

`function GetWaveLength(Panel: longint; var WL: single): boolean;`

Function `GetWaveLength` is used to query the current wavelength.

Parameters:
`Panel` is the ID of the current grating (valid values range from `PANEL_GRATING1` to `PANEL_GRATING4`).
`WL` is the value of wavelength [nm], providing the returned value is `True`.

Result: `True` if successful, `False` otherwise.

Usage example(s): `if GetWaveLength(PANEL_GRATING1, YourVar) then...`

15 GetDispersion

```
function GetDispersion(Panel:  longint; var Dispersion:  single):  boolean;
```

Function **GetDispersion** is used to query the current reciprocal dispersion.

Parameters:
Panel is the ID of the current grating (valid values range from **PANEL_GRATING1** to **PANEL_GRATING4**).

Dispersion is the value of the current reciprocal dispersion [nm/mm], providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if GetDispersion(PANEL_GRATING1, YourVar) then...`

16 MoveDeviceEnabled

```
procedure MoveDeviceEnabled(IsEnabled:  boolean);
```

Procedure **MoveDeviceEnabled** is used to permit or forbid movement of mobile parts of the active spectral instrument.

Parameters:

If **IsEnabled** is **True** then all mobile parts are allowed to change their state, otherwise not.

Usage example(s): `MoveDeviceEnabled(True);`

Remark: By default (i.e., before the very first call of this routine), all mobile parts are allowed to change their state, providing that the device is correctly initialised. After execution of **MoveDeviceEnabled(False)** the function **IsOperationFinished** will return **False** until **MoveDeviceEnabled(True)** is executed.

17 SetWaveLength

```
function SetWaveLength(Panel:  longint; WL: single; IsReset:  boolean = False):
boolean;
```

Function **SetWaveLength** is used to set a new value of the wavelength.

Parameters:
Panel is the ID of the current grating (valid values range from **PANEL_GRATING1** to **PANEL_GRATING4**).

WL is the new value of wavelength [nm] to be set.

If **IsReset** is **True** then a reset operation will be executed prior to setting the new wavelength, otherwise the new wavelength will be set directly (i.e., without resetting). Can be omitted; the default value is **False**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if SetWaveLength(PANEL_GRATING1, YourVar, False) then...`

Remark: Use function **IsOperationFinished** to check whether the operation is still in progress or already finished. Use function **GetDeviceError** to check whether any error is encountered.

18 SetResetGrating

```
function SetResetGrating(Panel:  longint):  boolean;
```

Function **SetResetGrating** is **obsolete** and retained for backward compatibility.
Replaced by **SetResetEx(Panel)**.

19 GetWaveLengthByPixel

```
function GetWaveLengthByPixel(Panel: longint; Shift, CentralWavelength: single;
var WL:single;PixelIndex: integer = -1): boolean;
```

Function **GetWaveLengthByPixel** is used to calculate the value of wavelength at a specified distance from the central pixel.

Due to dispersive elements (e.g., diffraction gratings) present in spectral devices an incident polychromatic beam is separated into its constituent wavelength components. Each wavelength of input beam spectrum is thus sent into a different direction, producing a rainbow of colors under white light illumination. In this way one can register that spectrum, e.g., by using a charged-coupled device or active pixel sensor (also called CMOS sensor).

Parameters:
Panel is the ID of the grating, for which calculations is required (valid values range from **PANEL_GRATING1** to **PANEL_GRATING4**).

Shift is the distance [μm] from the so-called "central" pixel to the point in which we need to know the value of wavelength. Its value must be negative or positive for pixels with indices lower or higher than that of the central pixel, respectively.

Central Wavelength is the value of wavelength [nm] for the central pixel. If this parameter <= 0 then current grating and its current wavelength are used. If you set Central Wavelength > 0 then you should pass the correct Panel parameter for the desired grating to the function. This allows to calculate calibration for different gratings and different wavelengths without actually changing or moving grating, which can take some time.

WL is the value of wavelength [nm], providing the returned value is **True**.

PixelIndex is the value of the pixel index. This parameter is **obsolete** and not used anymore (e.g., any value can be passed in its place). Can be omitted; the default value is **-1**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if GetWaveLengthByPixel(PANEL_GRATING1, 5, 0, YourVar)) then...`

Remark: Function **GetWaveLengthByPixel** performs now faster and more correct calculations. In fact, in the current implementation this function uses only the values of **Panel** and **Shift** and assigns the calculated value to **WL**. The other two parameters (**PixelSize** and **PixelIndex**) are ignored and retained only for backward compatibility. It means also that this function calculates the value of the wavelength independently of the pixel index, i.e., **Shift** can be multiple of **PixelSize**, but also can be, e.g., **0.5*PixelSize** or **-200.437*PixelSize** (as before, negative (positive) values of **Shift** stand for pixels with indices lower (higher) than that of the central pixel, respectively).

20 GetCentralWLByNullPixelWL

```
function GetCentralWLByNullPixelWL(Panel: longint; Length, PixelSize, NullWL:
single; var slCentralWL, slRightWL: single; NumPixel: integer = -1): boolean;
```

Function **GetCentralWLByNullPixelWL** is **obsolete** and retained for backward compatibility. Please, avoid using this function!

21 GetWLPixelByCentralWL

```
function GetWLPixelByCentralWL(Panel: longint; Length, PixelSize, CentralWL:
single; var WL: single; NumPixel: integer = -1): boolean;
```

Function **GetWLPixelByCentralWL** is **obsolete** and retained for backward compatibility. Please, avoid using this function!

22 SetSlitWidth

```
function SetSlitWidth(Panel: longint; Width: single; IsSpectralWidth: boolean =
False): boolean;
```

Function **SetSlitWidth** is used to set a new value of the slit width.

Parameters:

Panel is the slit ID (valid values range from **PANEL_SLIT1** to **PANEL_SLIT4**).

Width is the new value of the width to be set.

If **IsSpectralWidth** is **True** then the slit width will be set so as to provide a spectral width of **Width** [nm] (according to the current value of dispersion), otherwise the slit width will be set equal to **Width** [μm]. Can be omitted; the default value is **False**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if SetSlitWidth(PANEL_SLIT1, 100) then...`

Remark: Use function **IsOperationFinished** to check whether the operation is still in progress or already finished. Use function **GetDeviceError** to check whether any error is encountered.

23 SetFilter

```
function SetFilter(Panel: longint; FilterIndex: longint): boolean;
```

Function **SetFilter** is used to set a new filter.

Parameters:
Panel is the filter ID (valid values range from **PANEL_FILTER1** to **PANEL_FILTER4**).

FilterIndex is the index of filter to be set (starting from 1).

Result: **True** if successful, **False** otherwise.

Usage example(s): `if SetFilter(PANEL_FILTER1, 1) then...`

Remark: Use function **IsOperationFinished** to check whether the operation is still in progress or already finished. Use function **GetDeviceError** to check whether any error is encountered.

24 SetMirror

```
function SetMirror(Panel: longint; Position: longint): boolean;
```

Function **SetMirror** is used to set new position of a mirror.

Parameters:
Panel is the mirror ID (valid values range from **PANEL_MIRROR1** to **PANEL_MIRROR4**).

Position is the index of the mirror position to be set (starting from 1).

Result: **True** if successful, **False** otherwise.

Usage example(s): `if SetMirror(PANEL_MIRROR1, 1) then...`

Remark: Use function **IsOperationFinished** to check whether the operation is still in progress or already finished. Use function **GetDeviceError** to check whether any error is encountered.

25 SetMultiPosDev

```
function SetMultiPosDev(Panel: longint; Position: longint): boolean;
```

Function **SetMultiPosDev** is used to set new position of a MultiPos.

Parameters:
Panel is the MultiPos ID (valid values range from **PANEL_MULTIPOS1** to **PANEL_MULTIPOS4**).

Position is the index of the new position to be set (starting from 1).

Result: **True** if successful, **False** otherwise.

Usage example(s): `if SetMultiPosDev(PANEL_MULTIPOS1, 1) then...`

Remark: Use function **IsOperationFinished** to check whether the operation is still in progress or already finished. Use function **GetDeviceError** to check whether any error is encountered.

26 SetTurnSlit

```
function SetTurnSlit(Panel:  longint; Angle:  single):  boolean;
```

Function **SetTurnSlit** is used to set new position of a TurnSlit.

Parameters:
Panel is the TurnSlit ID (valid values range from **PANEL_TURNSLIT1** to **PANEL_TURNSLIT2**).

Angle is the new position (angle [angular degree], valid range from 0 to 22°) to be set.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if SetTurnSlit(PANEL_TURNSLIT1, 1) then...`

Remark: Use function **IsOperationFinished** to check whether the operation is still in progress or already finished.
Use function **GetDeviceError** to check whether any error is encountered.

27 SetTurret

```
function SetTurret(Panel:  longint; Position:  longint):  boolean;
```

Function **SetTurret** is used to set new position of a Turret (i.e., to set new diffraction grating).

Parameters:
Panel is the Turret ID (valid values range from **PANEL_TURRET1** to **PANEL_TURRET2**).

Position is the index of the new position (i.e., index of grating) to be set (starting from 1).

Result: **True** if successful, **False** otherwise.

Usage example(s): `if SetTurret(PANEL_TURRET1, 1) then...`

Remark: Use function **IsOperationFinished** to check whether the operation is still in progress or already finished.
Use function **GetDeviceError** to check whether any error is encountered.

28 MoveBySteps

```
function MoveBySteps(Panel:  longint; StepCount:  int64):  boolean;
```

Function **MoveBySteps** is used to set new position of a Drive (incremental or decremental operation, without resetting).

Parameters:
Panel is the Drive ID (valid values are listed in Table above).

StepCount is the number of steps to move from the current position (can be negative or positive).

Result: **True** if successful, **False** otherwise.

Usage example(s): `if MoveBySteps(PANEL_TURRET1, 100) then...`

Remark: Use function **IsOperationFinished** to check whether the operation is still in progress or already finished.
Use function **GetDeviceError** to check whether any error is encountered.

29 SetReset

```
function SetReset(Panel:  longint):  boolean;
```

Function **SetReset** is **obsolete** and retained for backward compatibility.
Replaced by **SetResetEx(Panel)**.

30 StepByStep

```
function StepByStep(Panel:  longint; StepCount:  longint):  boolean;
```

Function **StepByStep** is **obsolete** and retained for backward compatibility. Please, avoid using this function!

31 SetPreviousPosition

```
function SetPreviousPosition:  boolean;
```

Function **SetPreviousPosition** is used to restore correct positions of all mobile parts after switching the device on.

Parameters: None.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if SetPreviousPosition then...`

Remark: Use function **IsOperationFinished** to check whether the operation is still in progress or already finished. Use function **GetDeviceError** to check whether any error is encountered.

32 GetSlitWidth

```
function GetSlitWidth(Panel:  longint; var Width:  single):  boolean;
```

Function **GetSlitWidth** is used to query the current width of a slit.

Parameters:
Panel is the slit ID (valid values range from **PANEL_SLIT1** to **PANEL_SLIT4**).

Width is the value of the current slit width, providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if GetSlitWidth(PANEL_SLIT1, YourVar) then...`

Remark: Depending on the current setting of **IsSpectralWidth** in **SetSlitWidth**, the value of **Width** will be in [μm] or [nm] (see Sect. 22 for details).

33 GetMirrorPos

```
function GetMirrorPos(Panel:  longint; var Position:  longint):  boolean;
```

Function **GetMirrorPos** is used to query the current position of a mirror.

Parameters:
Panel is the mirror ID (valid values range from **PANEL_MIRROR1** to **PANEL_MIRROR4**).

Position is the index of the current position of the mirror, providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if GetMirrorPos(PANEL_MIRROR1, YourVar) then...`

34 GetMirrorPosName

```
function GetMirrorPosName(Panel, Position:  longint; Name:  PChar):  boolean;
```

Function **GetMirrorPosName** is used to query the name of specified position of a mirror.

Parameters:
Panel is the mirror ID (valid values range from **PANEL_MIRROR1** to **PANEL_MIRROR4**).

Position is the index of the specified position of the mirror (starting from 1).

Name is the name of the position of the mirror, providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if GetMirrorPosName(PANEL_MIRROR1, 1, YourVar) then...`

35 GetTurretPos

```
function GetTurretPos(Panel:  longint; var Position:  longint):  boolean;
```

Function **GetTurretPos** is used to query the current position of a turret (i.e., the index of the active diffraction grating).

Parameters:
Panel is the turret ID (valid values range from **PANEL_TURRET1** to **PANEL_TURRET2**).

Position is the current position of the turret (i.e., the index of the active diffraction grating), providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if GetTurretPos(PANEL_TURRET1, YourVar) then...`

36 GetTurretPosName

```
function GetTurretPosName(Panel, Position:  longint; Name:  PChar):  boolean;
```

Function **GetTurretPosName** is used to query the name of specified position of a turret.

Parameters:
Panel is the turret ID (valid values range from **PANEL_TURRET1** to **PANEL_TURRET2**).

Position is the index of the turret position (starting from 1).

Name is the name of the turret position, providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if GetTurretPosName(PANEL_TURRET1, 1, YourVar) then...`

37 GetCurGratingParam

```
function GetCurGratingParam(Panel:  integer; var Lines, Blaze:  integer):  boolean;
```

Function **GetCurGratingParam** is used to query the main parameters of a grating.

Parameters:
Panel is the grating ID (valid values range from **PANEL_GRATING1** to **PANEL_GRATING4**).

Lines is the number of lines (grooves, rulings) in [mm^{-1}], providing the returned value is **True**.

Blaze is the blazing wavelength in [nm], providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if GetCurGratingParam(PANEL_GRATING1, YourVar1, YourVar2) then...`

38 ShowDeviceWindowEx

```
procedure ShowDeviceWindowEx(Panel:  int64);
```

Procedure **ShowDeviceWindowEx** is used to display the main software window with selected control panel(s).

Parameters:
Panel is the control panel ID (valid values are listed in Table above). If **Panel** = PANEL_ALL then the main window will contain panels for all subdevices (drives) installed in your spectral instrument.

Usage example(s): `ShowDeviceWindowEx(PANEL_ALL);`

39 SetDigitPosValue

```
function SetDigitPosValue(Panel:  int64; Value:  single):  boolean;
```

Function **SetDigitPosValue** is used to set new value of a DigitPos.

Parameters:
Panel is the DigitPos ID (valid values range from **PANEL_DIGITPOS1** to **PANEL_DIGITPOS8**).

Value is the new value to be set.

Result: `True` if successful, `False` otherwise.

Usage example(s): `if SetDigitPosValue(PANEL_DIGITPOS1, 0) then...`

Remark: Use function `IsOperationFinished` to check whether the operation is still in progress or already finished. Use function `GetDeviceError` to check whether any error is encountered.

40 GetDevCtrlVersion

```
function GetDevCtrlVersion:  integer;
```

Function `GetDevCtrlVersion` is used to retrieve the minor version of the software (currently, the major version is `1`).

Parameters: None.

Result: The minor version of the software, providing the file "Devctrl.dll" resides in the same directory as your application's executable, otherwise "`0`".

Usage example(s): `YourVar:= GetDevCtrlVersion;`

41 GetDigitPosValue

```
function GetDigitPosValue(Panel:  int64; var Value:  single):  boolean;
```

Function `GetDigitPosValue` is used to query the current value of a DigitPos.

Parameters:
`Panel` is the DigitPos ID (valid values range from `PANEL_DIGITPOS1` to `PANEL_DIGITPOS8`).

`Value` is the current value of the DigitPos, providing the returned value is `True`.

Result: `True` if successful, `False` otherwise.

Usage example(s): `if GetDigitPosValue(PANEL_DIGITPOS1, YourVar) then...`

42 SetState

```
function SetState(Panel:  int64; IsDeviceOn:  integer):  boolean;
```

Function `SetState` is used to switch certain subdevice on or off.

Parameters:
`Panel` is the subdevice ID (valid values range from `PANEL_DIGITPOS1` to `PANEL_DIGITPOS8` and from `PANEL_BISTABLE1` to `PANEL_BISTABLE2`).

`IsDeviceOn` must be either `STATE_ON` or `STATE_OFF` (see Other constants).

Result: `True` if successful, `False` otherwise.

Usage example(s): `if SetState(PANEL_DIGITPOS1, STATEON_ON) then...`

43 QueryData

```
function QueryData(Panel:  int64; Timeout:  integer):  boolean;
```

Function `QueryData` is used to query data from a subdevice (e.g., PMT). More specifiedally, this routine performs single measurement.

Parameters:
`Panel` is the subdevice ID (valid values range from `PANEL_QUERY1` to `PANEL_QUERY4`).

`Timeout` is the timeout [ms] for this operation.

Result: `True` if successful, `False` otherwise.

Usage example(s): `if QueryData(PANEL_QUERY1, YourVar) then...`

Remark: The measured value can be retrieved by using function `ReadBuffer`.

44 InitInstrument

```
function InitInstrument:  boolean;
```

Function **InitInstrument** is used to initialise a measuring device. This function must be called **prior to** any measurements.

Parameters: None.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if InitInstrument then...`

45 ReadBuffer

```
function ReadBuffer(Destination:  Pointer; var DestSize:  DWORD): boolean;
```

Function **ReadBuffer** is used to retrieve the measured data from the software local buffer.

Parameters:
Destination is the pointer to the data buffer.

DestSize is the data size in bytes.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if ReadBuffer(YourVar1, YourVar2) then...`

Remark: The measurement itself is started by function **QueryData**.

46 GetGratingMechanismCount

```
function GetGratingMechanismCount:  integer;
```

Function **GetGratingMechanismCount** is used to retrieve the number of grating mechanisms (e.g., turrets) installed in your spectral instrument.

Parameters: None.

Result: Number of grating mechanisms (≥ 0).

Usage example(s): `YourVar:= GetGratingMechanismCount;`

47 GetGratingMechanismName

```
function GetGratingMechanismName(MechanismIndex:  integer):  PChar;
```

Function **GetGratingMechanismName** is used to query the name of specified grating mechanism (e.g., turret).

Parameters:
MechanismIndex is the index of the grating mechanism (starting from 1).

Result: Name of the specified grating mechanism.

Usage example(s): `YourVar:= GetGratingMechanismName(1);`

48 InitDeviceEx

```
function InitDeviceEx(hAppWnd:  HWND; var ComPort:  byte; FilePath:  PChar):
boolean;
```

Function **InitDeviceEx** is **obsolete** and retained for backward compatibility.
Replaced by **InitDeviceEx2(hAppWnd, ComPort, FilePath, True)**.

49 InitDeviceEx2

```
function InitDeviceEx2(hAppWnd: HWND; var ComPort: byte; FilePath: PChar;
IsShowDialog: boolean): boolean;
```

Function **InitDeviceEx2** is used to initialise your spectral instrument. This routine opens the specified COM-port, communicates with the device, and loads the necessary configuration (***.cfg**) file.

Parameters:
hAppWnd is the main window handle.

ComPort is the COM-port index (starting from 1). If **ComPort** = **0**, then an automatic search will be performed, otherwise starting from the specified index.

FilePath is the path to the configuration file of your device (e.g., if you have a MSDD1000, it must be a fully qualified path to the directory where **MSDD1000.cfg** resides). If **FilePath** = **nil**, then the software will try to access the configuration file in the directory where the Device Control DLL (**Devctrl.dll**) resides.

If **IsShowDialog** is **True**, then an additional message dialog will be displayed asking whether to execute the so-called "previous position" operation (see **SetPreviousPosition**) just after the device initialisation.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if InitDeviceEx2(Application.MainWindow.Handle, 0, nil, True) then...`

50 SetParentControl

```
procedure SetParentControl(ParentControl: TWinControl);
```

Procedure **SetParentControl** is used to set a parent control for the software main window. That is, the main window of the DLL will be contained in the TWinControl of your choice (e.g., in your form).

Parameters:
ParentControl is a TWinControl to serve as a container for the DLL main window.

Usage example(s): `SetParentControl(YourForm);`

Remark: To show the main window of the DLL in a container, this routine must be executed **prior to** calling procedure **ShowDeviceWindow** or **ShowDeviceWindowEx**.

51 LockPanels

```
function LockPanels(IsLockPanels: boolean): boolean;
```

Function **LockPanels** is used to lock or unlock all control panels.

Parameters:
If **IsLockPanels** is **True** (**False**), then all control panels in the device main window will be locked (unlocked), respectively.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if LockPanels(True) then...`

52 GetDevicePurpose

```
function GetDevicePurpose: integer;
```

Function **GetDevicePurpose** is used to retrieve the purpose (class) of a Solar JS device.

Parameters:

Result: **-1** stands for error, **0** for a Fluorimeter, and **1** for a Photometer.

Usage example(s): `YourVar:= GetDevicePurpose;`

Remark: If your spectral instrument is not a Solar JS device, the routine will return **-1**.

53 GetLampState

```
function GetLampState:  integer;
```

Function **GetLampState** is used to retrieve the lamp state of a Solar JS device.

Parameters:

Result: The lamp state.

Usage example(s): **YourVar:= GetLampState;**

Remark: If your spectral instrument is not a Solar JS device, the routine will return **-1**.

54 GetMultiPosDev

```
function GetMultiPosDev(Panel:  longint; var Position:  longint):  boolean;
```

Function **GetMultiPosDev** is used to query the current position of a MultiPos.

Parameters:
Panel is the MultiPos ID (valid values range from **PANEL_MULTIPOS1** to **PANEL_MULTIPOS4**).

Position is the current position of the MultiPos, providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): **if GetMultiPosDev(PANEL_MULTIPOS1, YourVar) then...**

55 CheckConnection

```
function CheckConnection:  boolean;
```

Function **CheckConnection** is used to check connection with your spectral device.

Parameters: None.

Result: **True** if connection is OK, **False** otherwise.

Usage example(s): **if not CheckConnection then...**

56 SetDigitPosExValue

```
function SetDigitPosExValue(Panel:  int64; Value:  single; IsReset:  boolean):
boolean;
```

Function **SetDigitPosExValue** is used to set new value of a DigitPosEx.

Parameters:
Panel is the DigitPosEx ID (valid values range from **PANEL_DIGITPOSEX1** to **PANEL_DIGITPOSEX4**).

Value is the new value to be set.

If **IsReset** is **True** then a reset operation will be executed prior to setting the new value, otherwise the new value will be set directly (i.e., without resetting).

Result: **True** if successful, **False** otherwise.

Usage example(s): **if SetDigitPosExValue(PANEL_DIGITPOSEX1, 0, False) then...**

Remark: Use function **IsOperationFinished** to check whether the operation is still in progress or already finished. Use function **GetDeviceError** to check whether any error is encountered.

57 GetDigitPosExValue

```
function GetDigitPosExValue(Panel:  int64; var Value:  single):  boolean;
```

Function **GetDigitPosExValue** is used to query the current position of a DigitPosEx.

Parameters:
Panel is the DigitPosEx ID (valid values range from **PANEL_DIGITPOSEX1** to **PANEL_DIGITPOSEX4**).

Value is the current position of the DigitPosEx, providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if GetDigitPosExValue(PANEL_DIGITPOSEX1, YourVar) then...`

58 GetDigitPosExCorrection

```
function GetDigitPosExCorrection(Panel:  int64; var Value:  single):  boolean;
```

Function **GetDigitPosExCorrection** is used to query the correction coefficient of a DigitPosEx.

Parameters:
Panel is the DigitPosEx ID (valid values range from **PANEL_DIGITPOSEX1** to **PANEL_DIGITPOSEX4**).

Value is the correction coefficient of the DigitPosEx, providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if GetDigitPosExCorrection(PANEL_DIGITPOSEX1, YourVar) then...`

59 SetInfoType

```
function SetInfoType(InfoType:  byte):  boolean;
```

Function **SetInfoType** is used to set a new info type for sending so-called “trace messages” to the **TraceMonitor** utility used in Solar TII internally for debugging.

Parameters:
InfoType is the info type to be set (valid values are **INFO_TIME** and **INFO_DIFFS**).

Result: **True** if successful, **False** otherwise.

Usage example(s): `if SetInfoType(INFO_TIME) then...`

60 SetResetEx

```
function SetResetEx(Panel:  int64):  boolean;
```

Function **SetResetEx** is used to reset a subdevice or drive in your spectral device.

Parameters:
Panel is the subdevice ID (valid values are from **PANEL_GRATING1** to **PANEL_GRATING4**; from **PANEL_SLIT1** to **PANEL_SLIT4**; from **PANEL_TURNSLIT1** to **PANEL_TURNSLIT2**; from **PANEL_TURRET1** to **PANEL_TURRET2**; from **PANEL_FILTER1** to **PANEL_FILTER4**; from **PANEL_MIRROR1** to **PANEL_MIRROR4**; from **PANEL_MULTIPOS1** to **PANEL_MULTIPOS4**; and from **PANEL_DIGITPOSEX1** to **PANEL_DIGITPOSEX4**).

Result: **True** if successful, **False** otherwise.

Usage example(s): `if SetResetEx(PANEL_GRATING1) then...`

Remark: Use function **IsOperationFinished** to check whether the operation is still in progress or already finished. Use function **GetDeviceError** to check whether any error is encountered.

61 GetTurnSlit

```
function GetTurnSlit(Panel:  longint; var Angle:  single):  boolean;
```

Function **GetTurnSlit** is used to query the current position of a TurnSlit.

Parameters:
Panel is the TurnSlit ID (valid values range from **PANEL_TURNSLIT1** to **PANEL_TURNSLIT2**).

Angle is the current position of the TurnSlit, providing the returned value is **True**.

Result: `True` if successful, `False` otherwise.

Usage example(s): `if GetTurnSlit(PANEL_TURNSLIT1, YourVar) then...`

62 GetFilter

`function GetFilter(Panel: longint; var FilterIndex: longint): boolean;`

Function `GetFilter` is used to query the active filter index (e.g., in a filter wheel).

Parameters:
`Panel` is the filter ID (valid values range from `PANEL_FILTER1` to `PANEL_FILTER4`).

`FilterIndex` is the index of the active filter, providing the returned value is `True`.

Result: `True` if successful, `False` otherwise.

Usage example(s): `if GetFilter(PANEL_FILTER1, YourVar) then...`

63 GetFilterNum

`function GetFilterNum(Panel: longint; var PositionCount: longint): boolean;`

Function `GetFilterNum` is used to query the number of positions in a filters holder (e.g., in a filter wheel).

Parameters:
`Panel` is the filter ID (valid values range from `PANEL_FILTER1` to `PANEL_FILTER4`).

`PositionCount` is the number of positions in the filters holder (filter wheel), providing the returned value is `True`.

Result: `True` if successful, `False` otherwise.

Usage example(s): `if GetFilterNum(PANEL_FILTER1, YourVar) then...`

64 GetFilterCaption

`function GetFilterCaption(Panel: longint; FilterIndex: longint; Caption: PChar): boolean;`

Function `GetFilterCaption` is used to query the caption of a filter.

Parameters:
`Panel` is the filter ID (valid values range from `PANEL_FILTER1` to `PANEL_FILTER4`).

`FilterIndex` is the index of the filter.

`Caption` is the caption (name) of the specified filter, providing the returned value is `True`.

Result: `True` if successful, `False` otherwise.

Usage example(s): `if GetFilterCaption(PANEL_FILTER1, 1, YourVar) then...`

65 CheckDevice

`function CheckDevice: boolean;`

Function `CheckDevice` is used to check whether your spectral device found OK.

Parameters: None.

Result: `True` if the device found OK, `False` otherwise.

Usage example(s): `if not CheckDevice then...`

66 ShowConfig

`function ShowConfig(Panel: integer): boolean;`

Function **ShowConfig** is used to display a form showing the current configuration of a specified subdevice or all subdevices at once.

Parameters:
Panel is the subdevice (control panel) ID (valid values are listed in Table above) for which its configuration must be shown. If **Panel** = PANEL_ALL then the displayed form will contain the current configuration of all subdevices (drives) installed in your spectral device.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if ShowConfig(PANEL_GRATING1) then...`

67 MakeScan

```
function MakeScan(Panel:  int64; FinalWL, StepSize:  single; var PointCount:
longint):  boolean;
```

Function **MakeScan** is used to scan a specified wavelength range (e.g., with a PMT).

The scanning will be started from the current wavelength!

Parameters:
Panel is the grating ID (valid values range from PANEL_GRATING1 to PANEL_GRATING4).

FinalWL is the wavelength [nm] where the scan will be finished.

StepSize is the wavelength step [nm] of the current scan.

PointCount is the total number of measured values obtained during the current scan, providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if MakeScan(PANEL_GRATING1, YourVar1, YourVar2, YourVar3) then...`

Remark: The measured values can be retrieved by using function **GetMeasuredValue**.

68 GetMeasuredValue

```
function GetMeasuredValue(var Value:  DWORD): boolean;
```

Function **GetMeasuredValue** is used to retrieve the next single value measured during a scan (e.g., with a PMT).

In order to retrieve all measured values after the scan is complete, call this function **PointCount** times (see parameter **PointCount** in function **MakeScan**).

Parameters:

Value is the next measured value, providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if GetMeasuredValue(YourVar) then...`

Remark: The scan itself is started by function **MakeScan**.

69 GetRange

```
function GetRange(Panel:  int64; var Min, Max:  single):  boolean;
```

Function **GetRange** is used to retrieve the valid range for a specified subdevice.

Parameters:

Panel is the subdevice ID (valid values range from PANEL_GRATING1 to PANEL_GRATING4; from PANEL_DIGITPOS1 to PANEL_DIGITPOS8; and from PANEL_DIGITPOSEX1 to PANEL_DIGITPOSEX4).

Min is the valid minimum value for the subdevice (e.g., wavelength range [nm] for a grating), providing the returned value is **True**.

Max is the valid maximum value for the subdevice (e.g., wavelength range [nm] for a grating), providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if GetRange(PANEL_GRATING1, YourVar1, YourVar2) then...`

70 GetConfigInfo

```
function GetConfigInfo(DeviceName:  PChar; var SN: longint):  boolean;
```

Function **GetConfigInfo** is used to retrieve both the name and the serial number of the active device from its configuration file (***.cfg**).

Parameters:

DeviceName is the device name, providing the returned value is **True**.

SN is the device serial number, providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if GetConfigInfo(YourVar1, YourVar2) then...`

71 GetDeviceInfo

```
function GetDeviceInfo(DeviceName:  PChar; var SN: DWORD): boolean;
```

Function **GetDeviceInfo** is used to retrieve both the name and the serial number of an active **Solar JS** device from its flash memory.

This routine is to be used only for **Solar JS**, but not **Solar TII** devices!

Parameters:

DeviceName is the device name, providing the returned value is **True**.

SN is the device serial number, providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if GetDeviceInfo(YourVar1, YourVar2) then...`

Remark: The returned value will be **False** if your device is not a **Solar JS** instrument.

72 GetMinWLStep

```
function GetMinWLStep(Panel:  longint; var MinStep:  single):  boolean;
```

Function **GetMinWLStep** is used to calculate the current (hardware-limited) minimum wavelength step.

Parameters:
Panel is the grating ID (valid values range from PANEL_GRATING1 to PANEL_GRATING4).

MinStep is the current minimum wavelength step [nm], providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if GetMinWLStep(PANEL_GRATING1, YourVar) then...`

73 GetNumSteps

```
function GetNumSteps(Panel:  longint; var StartWL, FinishWL, StepWL: single;
var PointCount:  integer):  boolean;
```

Function **GetNumSteps** is used to calculate the number of points within specified wavelength range using specified wavelength step.

Parameters:
Panel is the grating ID (valid values range from PANEL_GRATING1 to PANEL_GRATING4).

StartWL is the required starting wavelength [nm].

FinishWL is the required final wavelength [nm].

StepWL is the required wavelength step [nm].

PointCount is the calculated number of points, providing the returned value is **True**.

Providing the returned value is **True**, the required values of **StartWL**, **FinishWL** and **StepWL** will be substituted by real (hardware-bound) values!

Result: **True** if successful, **False** otherwise.

Usage example(s): **GetNumSteps(PANEL_GRATING1, YourVar1, YourVar2, YourVar3, YourVar4);**

74 GetWLInRange

```
function GetWLInRange(Panel:  longint; FromWL, StepWL: single; Index:  integer;
var WL: single):  boolean;
```

Function **GetWLInRange** is **obsolete** and retained for backward compatibility. Please, avoid using this function!

75 GetTurnSlitControlMode

```
function GetTurnSlitControlMode(Panel:  longint; var IsAuto:  boolean):  boolean;
```

Function **GetTurnSlitControlMode** is used to query the current control mode of a TurnSlit.

Parameters:
Panel is the TurnSlit ID (valid values range from **PANEL_TURNSLIT1** to **PANEL_TURNSLIT2**).

IsAuto is the current control mode of the TurnSlit, providing the returned value is **True**. If **IsAuto** = **True**, then the TurnSlit is turned automatically, otherwise manually.

Result: **True** if successful, **False** otherwise.

Usage example(s): **if GetTurnSlitControlMode(PANEL_TURNSLIT1, YourVar) then...**

76 SetTurnSlitControlMode

```
function SetTurnSlitControlMode(Panel:  longint; IsAuto:  boolean):  boolean;
```

Function **SetTurnSlitControlMode** is used to set the control mode of a TurnSlit.

Parameters:
Panel is the TurnSlit ID (valid values range from **PANEL_TURNSLIT1** to **PANEL_TURNSLIT2**).

IsAuto is the new control mode of the TurnSlit to be set. If **IsAuto** = **True**, then the TurnSlit will be turned automatically, otherwise manually.

Result: **True** if successful, **False** otherwise.

Usage example(s): **if SetTurnSlitControlMode(PANEL_TURNSLIT1, True) then...**

77 SendByteToCOM

```
function SendByteToCOM(Value: byte): boolean;
```

Function **SendByteToCOM** is used to send a byte to the active COM-port (debugging feature). This enables a direct communication with your spectral device if you have the corresponding communication protocol.

Parameters:
Value is the byte value to be sent.

Result: **True** if successful, **False** otherwise.

Usage example(s): `if SendByteToCOM(0) then...`

78 IsEchelleGrating

```
function IsEchelleGrating(GratingIndex: integer): boolean;
```

Function **IsEchelleGrating** is used to check whether specified grating is an Echelle grating.

Parameters:
GratingIndex is the grating index (starting from 1). If **GratingIndex** is set to **-1**, then the active grating will be analysed.

Result: **True** if the grating is an Echelle grating, **False** otherwise.

Usage example(s): `if IsEchelleGrating(1) then...`

79 GetGratingInformation

```
function GetGratingInformation(GratingIndex: integer; var Focus, AngularDeviation,
FocalPlaneTilt: single): boolean;
```

Function **GetGratingInformation** is used to retrieve some parameters of specified diffraction grating.

Parameters:
GratingIndex is the grating index (starting from 1). If **GratingIndex** is set to **-1**, then the active grating will be analysed.

Focus is the focal length [m].

AngularDeviation is the angular deviation ($\theta/2$) [radian].

FocalPlaneTilt (**obsolete**) is the focal plane tilt [radian], providing the returned value is **True**.

Result: **True** if parameter **FocalPlaneTilt** retrieved from the configuration file successfully, **False** otherwise.

Usage example(s): `GetGratingInformation(1, YourVar1, YourVar2, YourVar3);`

80 GetNumStepsEx

```
function GetNumStepsEx(Panel: longint; var StartWL, FinishWL, StepWL: single;
var PointCount: integer): boolean;
```

Function **GetNumStepsEx** is used to calculate the number of points within specified wavelength range using specified wavelength step.

Parameters:
Panel is the grating ID (valid values range from PANEL_GRATING1 to PANEL_GRATING4).

StartWL is the required starting wavelength [nm].

FinishWL is the required final wavelength [nm].

StepWL is the required wavelength step [nm].

PointCount is the calculated number of points, providing the returned value is **True**.

Providing the returned value is **True**, the required values of **StartWL**, **FinishWL** and **StepWL** will be substituted by real (hardware-bound) values!

Result: **True** if successful, **False** otherwise.

Usage example(s): **GetNumStepsEx(PANEL_GRATING1, YourVar1, YourVar2, YourVar3, YourVar4);**

81 GetWLInRangeEx

function GetWLInRangeEx(Panel: longint; FromWL, StepWL: single; Index: integer; var WL: single): boolean;

Function **GetWLInRangeEx** is **obsolete** and retained for backward compatibility. Please, avoid using this function!

82 GetCurGratingParamEx

function GetCurGratingParamEx(GratingIndex: integer; var Lines, Blaze: integer): boolean;

Function **GetCurGratingParamEx** is used to query the main parameters of a grating.

Parameters:
GratingIndex is the grating index (starting from 1).

Lines is the number of lines (grooves, rulings) in [mm^{-1}], providing the returned value is **True**.

Blaze is the blazing wavelength in [nm], providing the returned value is **True**.

Result: **True** if successful, **False** otherwise.

Usage example(s): **if GetCurGratingParamEx(1, YourVar1, YourVar2) then...**

83 GetGratingsCount

function GetGratingsCount(Panel: integer; var GratingsCount: integer): boolean;

Function **GetGratingsCount** is used to query the number of gratings installed in your spectral device.

Parameters:
Panel is the turret ID (valid values range from PANEL_TURRET1 to PANEL_TURRET2).

GratingsCount is the number of gratings installed (registered) in your spectral device.

Result: **True** in all cases.

Usage example(s): **GetGratingsCount(PANEL_TURRET1, YourVar);**

84 SetShutterMode

function SetShutterMode(Panel: longint; IsTTL: boolean): boolean;

Function **SetShutterMode** is used to trigger the control mode of a shutter (either "**TTL**" or "**Soft**").

Parameters:
Panel is the shutter ID (valid values range from **PANEL_SHUTTER1** to **PANEL_SHUTTER4**).

If **IsTTL** is **True** then the routine will try to set the shutter control mode to "**TTL**", otherwise to "**Soft**".

Result: **True** if successful, **False** otherwise.

Usage example(s): **if SetShutterMode(PANEL_SHUTTER1, False) then...**.

Remark: Use function **IsOperationFinished** to check whether the operation is still in progress or already finished. Use function **GetDeviceError** to check whether any error is encountered.

85 SetFilterControl

```
procedure SetFilterControl(Panel:  longint; FilterControlState:  byte);
```

Procedure **SetFilterControl** is used to trigger the control mode of a filter (either "**Auto**" or "**Manual**").

Parameters:
Panel is the filter ID (valid values range from **PANEL_FILTER1** to **PANEL_FILTER4**).

FilterControlState determines the filter control mode (valid values are **FC_AUTO** and **FC_MANUAL** - self-explaining).

Usage example(s): `SetFilterControl(PANEL_FILTER1, FC_AUTO);`

86 GetFilterControlState

```
function GetFilterControlState(Panel:  longint):  byte;
```

Function **GetFilterControlState** is used to retrieve the control mode of a filter (either "**Auto**" or "**Manual**").

Parameters:
Panel is the filter ID (valid values range from **PANEL_FILTER1** to **PANEL_FILTER4**).

Result: **FC_AUTO** or **FC_MANUAL** if the filter is in the automatic or manual control mode, respectively.

Usage example(s): `if (GetFilterControlState(PANEL_FILTER1) = FC_MANUAL) then...`

87 GetSubDevicesCount

```
function GetSubDevicesCount(SubDeviceType:  byte):  integer;
```

Function **GetSubDevicesCount** is used to retrieve the number of subdevices of specified type installed in your SolarSpectralInstrumentsspectral instrument.

Parameters:
SubDeviceType is the subdevice type (valid values range from **SDT_GRATING** to **SDT_DIGITPOSEX**, i.e., all constants with "**SDT**" prefix in their name among those listed in Other constants).

Result: Number of subdevices of the specified type installed in your SolarSpectralInstrumentsspectral device.

Usage example(s): `if (YourVar:= GetSubDevicesCount(SDT_GRATING);`

88 GetFullDevCtrlVersion

```
function GetFullDevCtrlVersion(Directory:  PChar; var Major, Minor, Release, Build:
integer):  boolean;
```

Function **GetFullDevCtrlVersion** is used to retrieve the full software version.

Parameters:
Directory is a fully-qualified path to the **directory** where the file "Devctrl.dll" resides. If **Directory** is **nil**, the routine will try to retrieve the necessary information from the DLL assuming that it is located in the same directory as your application's executable.

Providing the returned value is **True**, **Major**, **Minor**, **Release**, and **Build** are the major and minor version, release, and build numbers, respectively.

Result: **True** if successful, **False** otherwise.

Usage example(s): `GetFullDevCtrlVersion(nil, YourVar1, YourVar2, YourVar3, YourVar4);`

Index