



OP510.DLL Description

V1.01

The following describes the use of the OP510.DLL that allows controlling OP508 and OP510 optical power meter programmatically. The OP510.DLL will have to be installed in the ..\SYSTEM32\ folder or a location the application software development system can locate. The latest release for the OP510.DLL will be available in the support section of the www.optotest.com website and can be requested from engineering@optotest.com.

General Calling Conventions

The OP510.DLL is written in native PASCAL, the functions are declared as STDCALL to provide a universal calling interface.

Data Types:	Integer	signed 32-bit
	Byte	unsigned 8-bit
	Real	8 byte double precision floating point
	PChar	null terminated string pointer

Corresponding Data Types in VB6:	
Integer	long
Byte	byte
Real	double
PChar	string

Access to multiple modules.

This OP510.DLL does not allow control of ONE module at the time. If multiple modules are connected to the port functions are provided to select one particular module. To have multiple modules concurrently open please contact OptoTest for more information and supporting DLLs.

Direct access to the USB driver.

Direct access to the USB driver is available by calling low level functions of the FTD2XX.DLL. Please request the documentation for this DLL, the DLL as well as the description of the binary USB commands for the OP508/510 module. Please note that by gaining access to the low level functions calibration parameters and other run-time variables of the OP508/510 are exposed. Unwillingly changing such variables can result in malfunction of the module and loss of the calibration.

OP510.DLL Description

InitDLL

This function initializes the DLL again. There is no need to call this function as upon installation of the DLL this will be executed automatically.

```
function InitDLL(): integer;
```

GetDLLStatus

Returns status of initialization of the DLL and the USB ports. Call this function first to make sure the USB port is active and the DLL is loaded correctly.

Upon load of the DLL and success full opening of the first available USB port with a OP250 module connected a test byte is sent to the OP250 module which is mirrored back by the module. This test sequence assures proper communication with the module.

Status: +1 all OK, USB handle<>0
 -5 no USB device found
 -4 error in communication, no match in test byte
 -3 USB Read Error
 -2 USB Write Error

```
function GetDLLStatus: integer
```

GetDLLRev

Returns the revision of this implementation of the DLL.

Parameter: revision: integer

```
function GetDLLRev: integer
```

GetUSBDeviceCount

Returns number of USB devices connected to the computer. Although it will only recognize OptoTest modules it will respond with a count of all OPxxx modules, that includes sources, power meters and other tytypes. Use this command to establish the number of modules connected to the computer then scan each individually using the *GetUSBSerialNumber*.

Parameter: deviceCount: integer
 Number of USB devices (OP modules) connected

Status: +1 always

```
function GetUSBDeviceCount (var deviceCount: integer): integer
```

OP510.DLL Description

GetUSBStatus

Returns USB error flag of last operation as well as the associated error code.

Parameter: haveError: Boolean
 TRUE if an error has occurred during last call
 FALSE no error occurred

Status FT_OK = 0
 FT_INVALID_HANDLE = 1
 FT_DEVICE_NOT_FOUND = 2
 FT_DEVICE_NOT_OPENED = 3
 FT_IO_ERROR = 4
 FT_INSUFFICIENT_RESOURCES = 5
 FT_INVALID_PARAMETER = 6

```
function GetUSBStatus(var haveError:boolean):integer
```

OpenDriver

This function assigns the USB port to the module and initializes the module parameters. The USB port is identified with a *handle*. If only one module is connected or the first available module wants to be addressed set the *handle=0*. For a selection of one out of many modules please refer to the later paragraph **Selection from multiple modules**.

Parameter: handle: integer handle to USB port (default=0)

Status: +1 all OK, USB handle<>0

```
function OpenDriver(handle: integer): integer;
```

GetSerialNumber

This function returns the integer serial number of the connected module. This is the number stored in the EEPROM of the module and should match the serial number of the USB.

Parameter: serialNumber: integer

```
function GetSerialNumber(var serialNumber: integer): integer
```

OP510.DLL Description

GetModuleID

Returns module ID of the module, use to identify what type of module is connected.

Status: module ID
 10 = OP250
 11 = OP508/510
 13 = OP710

function GetModuleID: integer

GetFWRevision

Returns module firmware revision of the module.

Status: revision: integer
 Before shipdate 5/1//04 revision is <=103.

function GetFWRevision: integer;

GetTemperature

Returns the ambient temperature in either degree Celsius or degree Farenheit.

NOTE: The temperature is NOT absolute calibrated however the relative measurement is accurate within 0.5 °C.

Parameter: temp: real Measured temperature in °C
 units: integer 1: celcius 2: fahrenheit

Status: >0 call successful completed
 <0 USB error, module not responding

function GetTemperature(var temp: real; units: integer): integer

GetActiveChannel

Return the number of the current selected channel.

NOTE: For 508/510 devices this is always 1.

Parameter: channel: integer [range 1...24]
 Current active channel

Status: >0 call successful completed
 <0 USB error, module not responding

function GetActiveChannel(var channel: integer): integer;

OP510.DLL Description

SetActiveChannel

Set the active channel.

No function for 508/510 devices.

Parameter: channel: integer
Active channel

Status: >0 call successful completed
<0 USB error, module not responding

function SetActiveChannel(channel: integer): integer;

ReadAnalog

Return the raw digital data from the output of the power meter module before the conversion to dBm and application of the calibration parameters. Use this function if the current set *gain* is of interest.

Parameter: analog: integer; AtoD reading
gain: integer; current gain setting
mode: integer; mode of the power meter, relative=1 or absolute=0

Status: >0 call successful completed
<0 USB error, module not responding

function ReadAnalog(var analog: real; var gain :integer; var mode: integer):
integer;

ReadPower

Return power of current active channel in dBm

Parameter: power: real
Measured power in dBm

Status: >0 call successful completed
<0 USB error, module not responding

function ReadPower(var power: real): integer;

OP510.DLL Description

ReadLoss

Return the loss of current active channel in dB. This assumes the reference has been previously set with *SetReference*. Note that a reference for each of the calibration wavelength can be taken and is stored.

Parameter: power: real
Measured power in dBm

Status: >0 call successful completed
<0 USB error, module not responding

function ReadLoss(var loss: real)L integer;

GetChannelBuffer

Initiate the measurement of all active channels at once. No function for 508/510 devices.

Parameter: none

Status: >0 call successful completed
<0 USB error, module not responding

ReadChannelBuffer

Return the power reading stored in the channel buffer. No function for 508/510 devices.

Parameter: channel: integer
powerLevel: real

Status: >0 call successful completed
<0 USB error, module not responding

SetAbsolute

Set current channel into absolute (dBm) mode. The OP510 will switch to absolute mode where the LED bargraph indicates the approximate absolute power level.

NOTE: The OP710 display will display the power levels measured in dBm.

Parameter: none

Status: >0 call successful completed
<0 USB error, module not responding

function SetAbsolute(): integer;

SetReference

Set current channel into relative (dB) mode, take the reference and store it according to the selected wavelength. For each calibration wavelength a reference is stored. The OP510 will switch to relative mode where the LED bargraph indicates the approximate relative power level starting out in the middle. The scale is approximate +/- 2dB. The pass/fail LED will light up if the measured loss is below that fail criteria, set at 1dB.

NOTE: The OP710 display will display the dB mode and current reference.

Parameter: none

Status: >0 call successful completed
<0 USB error, module not responding

function SetReference(): integer;

ReferencePower

Return the current reference power of the current active channel in dBm.

Parameter: refPower: real
Stored reference power in dBm

Status: >0 call successful completed
<0 USB error, module not responding

function ReferencePower(var refPower: real): integer;

SetAutoRange

Enable (default) or disable autorange. In autorange mode the individual power meter automatically selects the best amplifier range for the power measurement.

Parameter: state: integer
0: autorange is disabled
>0: autorange is enabled (default)

Status: >0 call successful completed
<0 USB error, module not responding

function SetAutoRange(state: integer):integer;

OP510.DLL Description

SetGain

Select the gain for the current power meter, only if the power meter is NOT in autorange.

Parameter: gainSetting: integer
0...7: valid gain range, 0=lowest gain, 7=highest gain.

Status: >0 call successful completed
<0 USB error, module not responding

function SetGain(gain: integer):integer;

SetWavelength

Set the calibration wavelength of the current selected channel. The passed wavelength needs to be within 5m to be accepted.

Parameter: waveLength: integer

Status: >0 call successful completed
<0 USB error, module not responding

function SetWavelength(wavelength: integer):integer;

NextWavelength

Advance calibration wavelength to next wavelength in calibration table, wrap around if at the end of the table.

Parameter: waveLength: integer new wavelength setting
waveIndex: integer; selected wavelength number
count: integer; total count of wavelength

Status: >0 call successful completed
<0 USB error, module not responding

function NextWavelength(var wavelength: integer; var waveIndex: integer;
var count: integer): integer;

OP510.DLL Description

GetWavelength

Get current set calibration wavelength.

Parameter: waveLength: integer new wavelength setting
 waveIndex: integer; selected wavelength number
 count: integer; total count of wavelength

Status: >0 call successful completed
 <0 USB error, module not responding

function GetWavelength(var wavelength: integer; var waveIndex: integer;
 var count: integer): integer;

BackLight

Turn on or off the backlight. Not available for 508/510.

Parameter: state: integer
 0 backlight off
 >0 backlight on (default)

Status: =1

function Backlight(state:integer): integer

CloseDriver

This function **has to be called** upon exit of the program as it closes the USB driver properly.

Status: =1

function CloseDriver: integer

SetDebug

Control a debug flag within the DLL. By default no debug messages are sent to the screen, only use this if debugging applications and lower level USB errors need to be monitored.

Parameter: dbg: integer
 0 no debug messages (default)
 >0 debug messages will be shown in popup windows

Status: =1

function SetDebug(dbg): integer

OP510.DLL Description

FactoryReset

This function resets the power meter module and restores some operational parameters. It also performs a power-on reset on the module. Use this function with caution.

```
function FactoryReset(): integer;
```

Selection from multiple modules

Multiple modules can be connected to the computer if the computer itself has multiple USB ports or by connecting an external USB hub. Multiple hubs can be cascaded per manufacturers specifications.

When selecting a single module the following sequence is recommended:

- 1) Query how many modules are active on the USB bus. The **GetUSBDeviceCount** function will return the number of devices (OptoTest devices) is currently connected to the bus.
- 2) Find a method for selecting one of the devices. The function **GetUSBSerialNumber** is the best method to list all the devices connected by serial number.
- 3) Open the one device. The function **OpenUSBDevice** opens the device that has been selected by device number and returns the handle to the USB port.
- 4) Open the driver and pass on the handle to the USB port using the function **OpenDriver**

Example: Query all connected source modules and open serial number "10101"

```
status = GetUSBDeviceCount(moduleCount);
for (index=0, i<moduleCount, i++) {
    status = GetUSBSerialNumber(index, snString);
    if Equal(snstring, '10101') then {
        displayString('opening device %i with sn %s');
        status = OpenUSBDevice(index, handle);
        status = OpenDriver(handle);
        exit;
    }
}
```

OP510.DLL Description

GetUSBSerialNumber

Returns the USB serial number of the device indicated to by *deviceNumber*.

Parameter: deviceNumber: integer
 snString: character pointer

Status OK =1

```
function GetUSBSerialNumber(deviceNumber: integer;  
                            var snString: string): integer;
```

OpenUSBDevice

This function call establishes and interfaces to one of the modules connected to any of the USB ports. The modules are differentiated with the *deviceNumber*. The USB device manager enumerates the connected modules starting at 0. The order of numeration is specific to each computer setup but always stays the same. Calling this function will return the *handle* to the USB driver for that port. That handle needs to be passed on to the source driver with the function *OpenDriver*.

Parameter: deviceNumber: integer [range: 0....]

Status OK =1

```
function OpenUSBDevice(deviceNumber: integer; var handle: integer): integer;
```

OP510.DLL Description

Example 1: Read the power from the one module that is connected.

```
status=OpenDriver(0);
status=ReadPower(powerDBM);
```

Example 2: Read the power from the second module that is connected. Set the calibration wavelength to 1310nm

```
Status=GetUSBDeviceCount(count);
If (count>=2) then {
    status=OpenDriver(1);
    status=SetWavelength(1310);
    status=ReadPower(powerDBM);
}
```

Example 3: Query the calibration wavelengths from a single module.

```
Status=GetUSBDeviceCount(count);
If (count>=1) then {
    status=OpenDriver(0);
    status=SetWavelength(0); // this sets the first available wavelength
    status=GetWavelength(wavelength, windex, count);
    for (i=0, i<=count,i++){
        status=NextWavelength(wavelength, windex,c);
        // do something with wavelength such as display
    }
}
```