Ocean Insight

OceanMathART

Programming Manual

For product: OceanMathART

Locations

Americas

Manufacturing & Logistics
3500 Quadrangle Blvd., Orlando, FL 32817, USA

Sales: info@oceaninsight.com
Orders: orders@oceaninsight.com
Support: techsupport@oceaninsight.com

Phone: +1 727.733.2447
Fax: +1 727.733.3962

Europe, Middle East & Africa

Sales & Support
Geograaf 24, 6921 EW Duiven, The Netherlands

Manufacturing & Logistics
Maybaachstrasse 11, 73760 Ostfildern, Germany

Email: info@oceaninsight.eu

Netherlands: +31 26-319-0500
Netherlands Fax: +31 26-319-0505
Germany: +49 711-341696-0
UK: +44 1865-819922
France: +33 442-386-588

Asia

Ocean Insight China
666 Gubei Rd., Kirin Tower Suite 601B
Changning District, Shanghai, PRC, 200336

Email: asiasales@oceaninsight.com

China: +86 21-6295-6600
China Fax: +86 21-6295-6708
Japan & Korea: +82 10-8514-3797

Ocean Insight India
Prestige Shantiniketan, Gate no.2
Tower C, 7th Floor
Whitefield main road, Mahadevpura
Bengaluru-560048 Karnataka, India

Phone: +91 80-67475336

www.oceaninsight.com

Table of Contents

Introduction

OceanMathART™ is a powerful Software Developer's Kit (SDK) that allows you to easily write custom software to process data from your Ocean Insight spectrometer. It provides complex functions such as calculating absorbance and reflectance. OceanMathART requires OceanMathCore to be installed for baseline functionality. Typically, OceanMathART is used with OceanDirect to simplify the ability to obtain spectral data from Ocean Insight spectrometers and then process it.

This document discusses the capabilities of OceanMathART and provides high level information on the data structures. Detailed information on the data structures is included in the SDK. In addition, sample code using the SDK may be found on the Ocean Insight website.

OceanMathART was developed in the C++ language and includes native libraries for Windows operating systems.

Operating System Support

- Windows - Windows 10 or higher

Multi-Language Support

You can develop OceanMathART-based applications in the following languages.

- C/C++/C#/Visual Basic (Microsoft Visual Studio environment)
- C (standard interface environment)
- LabVIEW (Windows only, Version 8 or greater)
- MATLAB
- Python (version 3 or later)

Development Environments

Windows Development

For purposes of programming in Windows, you can access OceanMathART functionality via two DLL files:

- OceanMathArt.dll contains the functions that allow you to manipulate spectra data. For example, you can calculate absorbance, convert wavelength to gigahertz, etc.

- NetOceanMathArt.dll is the same as above but specifically for development in the Microsoft .NET Framework.

LabVIEW Development

For LabVIEW developers, OceanMathART provides a set of VI files that expose its functionality in a fashion that is natural to the LabVIEW development environment. Behind the scenes, these VIs invoke the .NET methods contained in the DLL files that comprise OceanMathART.

MATLAB Development

For MATLAB developers, OceanMathART provides a set of 'm' file scripts that expose its functionality in a fashion that is natural to the MATLAB development environment. Behind the scenes, these scripts invoke the .NET methods contained in the DLL file for OceanMathART.

Installation

Upon purchasing the software, you will be provided a link for downloading the software and an associated license key. Click on the link to access the installation file. If you did not receive the link, or have misplaced it, request a replacement email via the technical support request form.

When the installation process is finished, the following subdirectories will be created beneath the OceanMath "home" directory:

Subdirectory	Contents
include	Header files for use with C/C++ application development
doc	Documentation relating to OceanMathART and its API
lib	Libraries for client applications

Once you have installed the software, you'll want to verify your installation, look at the samples provided on the OceanMath product page at OceanInsight.com to get an idea of how the objects and methods for OceanMathART are organized, and then run a sample program.

NOTE

OceanMathART uses functions in OceanMathCore. The order of installing the software (OceanMathCore and OceanMathART) does not matter, however both must be installed prior to using OceanMathART.

Installing OceanMathART Software

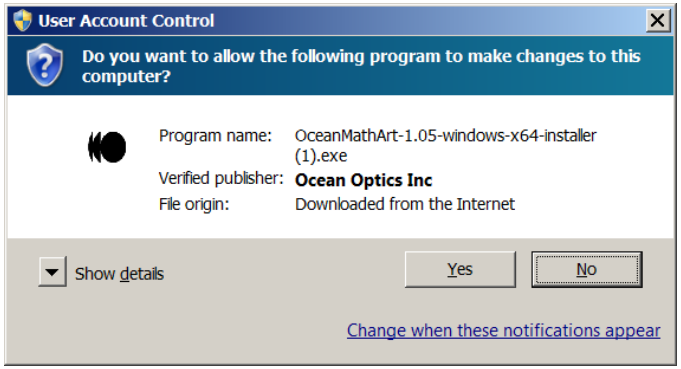
Simply download the file and double-click it in Windows Explorer to begin the installation procedure. The installer will guide you through the install process.

NOTE

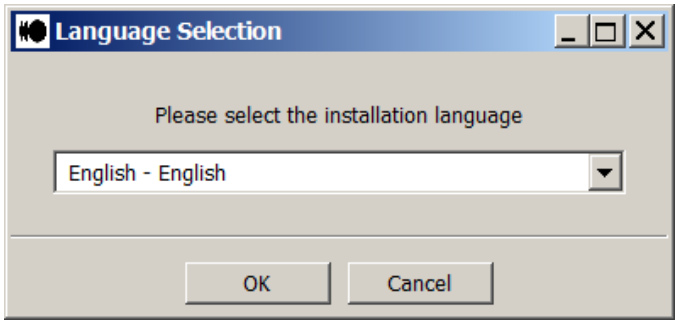
The computer on which you are installing the software must be connected to the Internet to validate the license key.

1. Start your Internet browser.
2. Navigate to the link provided to you and select the OceanMathART software (e.g., OceanMathART- x.xx-windows-64-installer.exe).
3. Save the software installer to the desired location on your computer.
4. Double-click on the file. The installer wizard guides you through the installation process. The default installation directory is c:\Program Files\Ocean Insight\OceanMath.

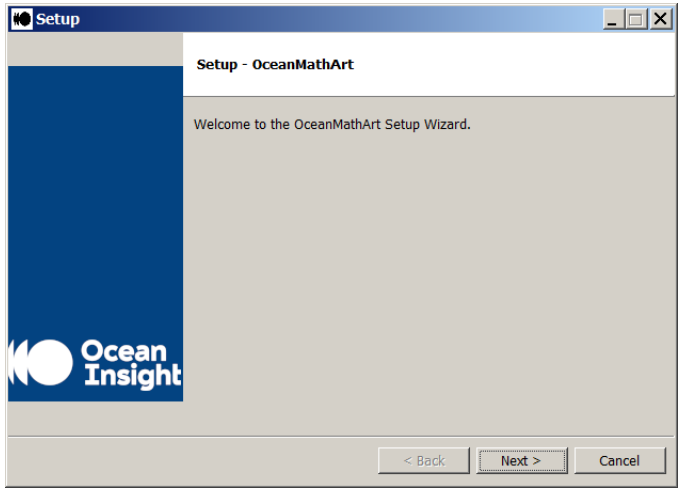
   a. Allow the installer access to your computer by clicking “Yes”.



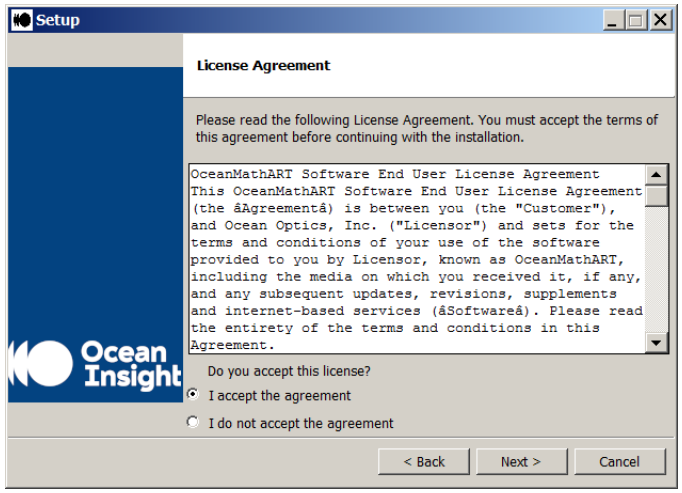
b. Choose the desired language from the drop-down list.



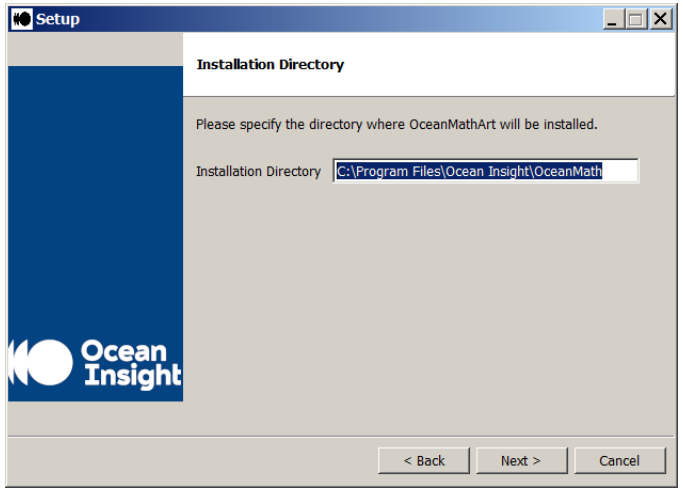
c. Allow the installation to begin by clicking “Next.”



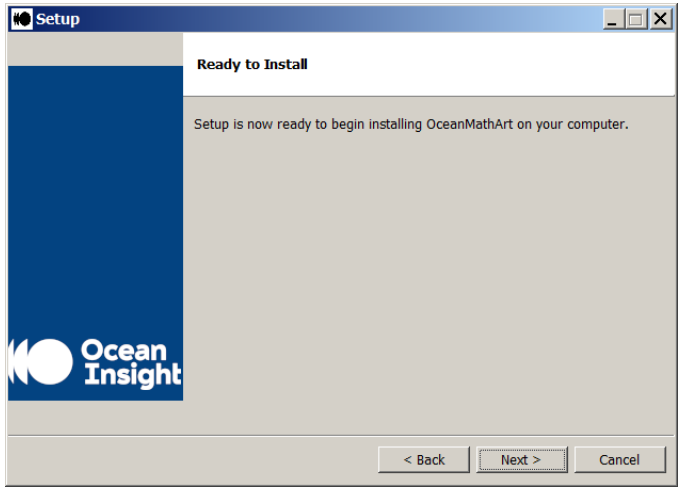
d. Review the license agreement. Select the “I accept the agreement” button, then click “Next”.



e. Choose your preferred file location. Click “Next” to continue.

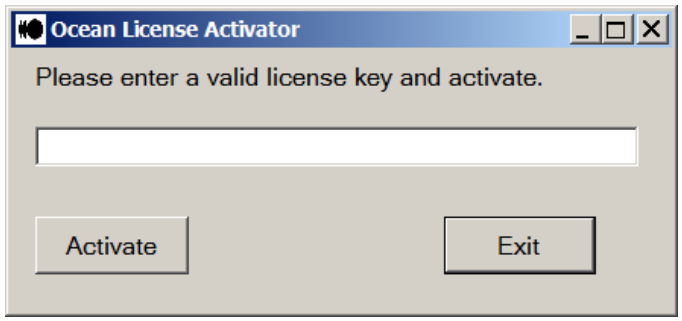


f. You are now ready to being the installation. Click “Next” to continue.

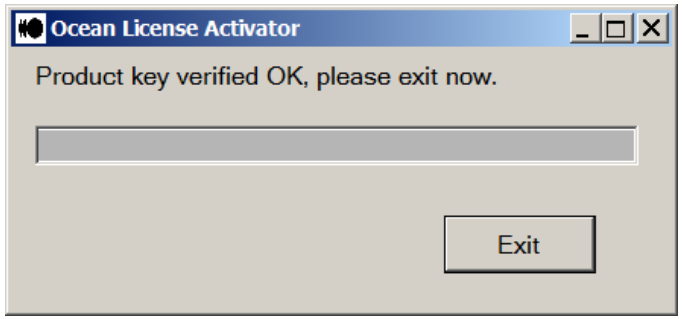


g. A progress bar is displayed showing the status of the installation.

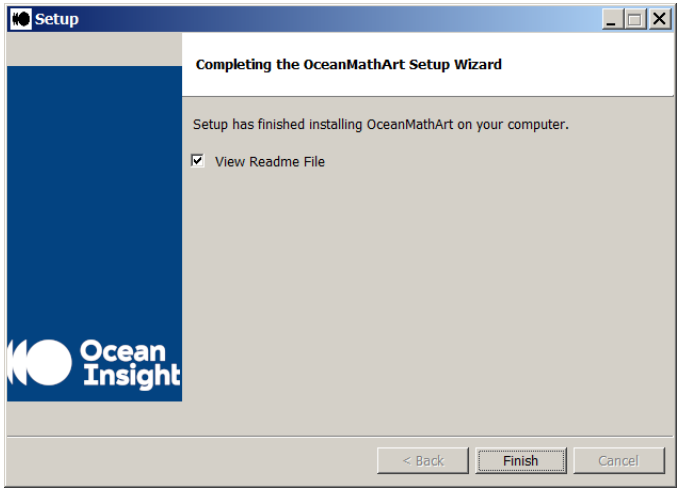
h. You will be prompted to enter your license key. Type the license number in and click on “Activate”.



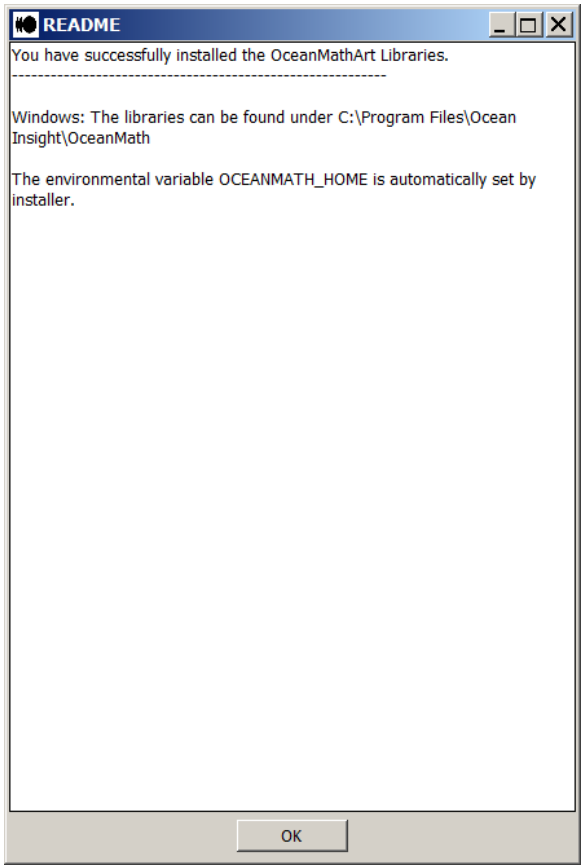
i. Once the license key has been verified, click on the “Exit” button.



j. Click “Finish” to complete the installation.



k. After reviewing the README file, click OK to exit.



NOTE

If you do not enter the license key during installation, you will not be able to successfully use the program. An error code will be returned for all function calls indicating that the license is not active.

To activate the license at a later time, use the license activator tool "OceanLicenseActivator.exe" located in the directory in which you installed the software. Run this tool and it will prompt you for the license key that will activate the software.

Using OceanMathART

Using OceanMathART with OceanDirect

The most convenient way of utilizing OceanMathART is to develop your application with OceanDirect. The output from OceanDIRECT can be used directly in OceanMathART to perform further processing – e.g., to compute absorbance and reflectance spectra.

Using OceanMathART Standalone

The user has the option to access the OceanMathART functions directly from their own application.

The following section discusses the various development environments that may be used. The data structures for developing in the .NET environment are provided in Appendix A. Additional information for developing in a C/C++ environment is available in the online documentation that is provided with the OceanMathART installation.

Development Environment

You may develop your application in any environment of your choosing. We will focus on Microsoft Visual Studio examples due to its availability and cross-platform capabilities.

Microsoft Visual Studio

To use the .NET assembly interface you must add a reference to the assembly as follows:

1. In your application, click on the **Project** menu item, and then choose **Add Reference**.
2. Click on the **Browse** tab.
3. Navigate to the OCEANMATH_HOME directory.
4. Highlight **NetOceanMathArt.dll** and click **OK**.

Creating a New C# Project that uses the .NET Interface to OceanMathART

1. Create new project of type C# “Windows Console”.
2. Add a reference to netOceanMathArt.dll:
   a. In Solution Explorer, right-click on **References** and choose **Add Reference…**
   b. Click the **Browse** tab.
   c. Navigate to the OCEANMATH_HOME folder.
   d. Highlight **NetOceanMathArt.dll** and click **OK**.
   e. Repeat Step d for the desired mathematical library (e.g., NetOceanMathArt.dll) if you are using any math functions.

Deploying Your C# Application

Normally, all that is needed to deploy your C# application is the EXE file containing the application itself. And you will also need to deploy the appropriate OceanMathART “redistributable” installer (no password required by this installer).

However, if your C# application uses the .NET assembly interface, you must also ensure that a copy of NetOceanMathArt.dll is placed in the same folder as your application’s EXE file. You can obtain this DLL file from the OCEANMATH_HOME directory.

LabVIEW

OceanMathART provides a .NET 4.0 interface. We recommend that all LabVIEW applications use the .NET 4.0 interface when accessing OceanDirect functions.

LabVIEW is able to show all the methods in a class as well as each method's inputs and outputs with default named variables. This graphical representation clearly documents the .NET interface within the LabVIEW environment.

The following steps must be performed when starting a new LabVIEW projects:

1. The first step is to place an "Invoke Node (.NET)" on the Block Diagram panel. Do this in the Functions window by selecting **Connectivity -> .NET -> Invoke Node (.NET)** and dragging it to the panel. Right-click the node and then choose **Select Class -> .NET -> Browse**. Navigate to the NetOceanMathArt.dll. The default location for the DLL is C:\Program Files\Ocean Insight\OceanMath\lib\.
2. In the **Select Object From Assembly** window, select the **NetOceanMathArt** object on the panel, click **OK**. This updates the node on the block diagram to ""NetOceanMathART". Right-click on the node and then choose **Select Method.** A list of all available methods is displayed. Select **[S]getInstance**. You can now call any of the methods found in the class e.g., `[S]calculateTransmission,` etc.

Visual Basic

Using the .NET Interface to OceanMathART

To use the .NET assembly interface to OceanMathART in your Visual Basic application, you must add a reference to your assembly as follows:

1. Click on the **Project** menu item and choose **Add Reference**.
2. Click on the **Browse** tab.
3. Navigate to the **OCEANMATH_HOME** directory.
4. Highlight the **NetOceanMathArt.dll** and click **OK.**

Sample Programs

A collection of sample programs for OceanMathART demonstrating basic functionality may be downloaded from the OceanMath product page at OceanInsight.com.

Appendix A - Data Structures

The OceanMathART API is the collection of objects and methods your application uses to control spectrometers and acquire data from them.

Depending on your development environment, you will use OceanMathArt.dll or NetOceanMathArt.dll (used in the Visual Studio environment). Public member functions for each class of the NetOceanMathART data structures are shown below. OceanMathART has similar names for the public functions.

The table below is a quick reference showing the public member functions for the NetOceanMathART class. The items in the table link to a description of the functions in this document. A more detailed description of the functions and the associated parameters may be found in the reference manual "NetOceanMathART_User_Manual.rtf" located at "c:\Program Files\Ocean Insight\OceanMath\Doc\net\rtf\. A similar manual for developing in the C/C++ environment, "OceanMathART_User_Manual.rtf" may be found at "c:\Program Files\Ocean Insight\OceanMath\doc\rtf\.

HTML versions of the reference manual may be found at "c:\Program Files\Ocean Insight\OceanMath\Doc\net\html\index.html" and "c:\Program Files\Ocean Insight\OceanMath\Doc\html\index.html".

Public Functions

Class	C# Public Functions
NetOceanMathART	calculateAbsorbance (1/2)
	calculateAbsorbance (2/2)
	calculateReflection *(1/2)*
	calculateReflection (2/2)
	calculateRelativeIrradiance (1/2)
	calculateRelativeIrradiance (2/2)
	calculateTransmission (1/2)

Class	C# Public Functions
	calculateTransmission *(2/2)*
	kubelkaMunkProcessReflectionSpectrum
	applyNonunityCorrectionalFactors
	getInterpolatedCorrectionFactors

static array<double> ***calculateAbsorbance*** (array< double >^ *dark*,

array< double >^ *reference*,

array< double >^ *sample*,

int% *errorCode*) [static]

- This function calculates the absorbance spectrum for a specified sample, reference and dark spectrum. This function should be used when the dark spectrum to be subtracted from the reference and the sample is the same -- i.e. the dark, reference and sample spectra have all been acquired sufficiently close in time that negligible drift in the dark spectrum will have occurred. The arrays used to store each of the spectra must be large enough to contain all of the pixels in each spectrum and the specified lengths for each spectrum must be identical.

static array<double> **calculateAbsorbance** (array< double >^ *referenceDark*,

array< double >^ *reference*,

array< double >^ *sampleDark*,

array< double >^ *sample*,

int% *errorCode*) [static]

- This function calculates the absorbance spectrum for a specified sample, reference spectrum with different dark spectra for the reference and sample. This function should be used when the dark spectra to be subtracted from the reference and the

sample are different -- e.g., when the reference and sample spectra have been taken sufficiently far apart in time to justify a dark spectrum to be taken for both the reference and sample.

static array<double> ***calculateReflection*** (array< double >^ *dark*,

array< double >^ *reference*,

array< double >^ *sample*,

int% *errorCode*) [static]

- This function calculates the reflectance spectrum for a specified sample, reference and dark spectrum. This function should be used when the dark spectrum to be subtracted from the reference and the sample is the same -- i.e., the dark, reference and sample spectra have all been acquired sufficiently close in time that negligible drift in the dark spectrum will have occurred. The arrays used to store each of the spectra must be large enough to contain all of the pixels in each spectrum and the specified lengths for each spectrum must be identical.

static array<double> ***calculateReflection*** (array< double >^ *referenceDark*,

array< double >^ *reference*,

array< double >^ *sampleDark*,

array< double >^ *sample*,

int% *errorCode*) [static]

- This function calculates the reflectance spectrum for a specified sample, reference spectrum with different dark spectra for the reference and sample. This function should be used when the dark spectra to be subtracted from the reference and the sample are different -- e.g., when the reference and sample spectra have been taken sufficiently far apart in time to justify a dark spectrum to be taken for both the reference and sample.

static array<double> ***calculateRelativeIrradiance*** (double *colorTemperature*,

array< double >^ *dark*,

array< double >^ *reference*,

array< double >^ *sample*,

array< double >^ *wavelengths*,

int% *errorCode*) [static]

- This function computes Relative Irradiance. Irradiance is the amount of energy at each wavelength emitted from a radiant sample. Before you can compute Relative Irradiance you must take a reference spectrum of a known color temperature. Additionally, you must obtain a dark spectrum by removing the fiber from the reference lamp and preventing light from entering it. Relative irradiance is a comparison of the fraction of energy the sample emits and the energy the sampling system collects from a lamp with a blackbody energy distribution (normalized to 1 at the energy maximum). This function should be used when the dark spectrum to be subtracted from the reference and the sample is the same -- i.e., the dark, reference and sample spectra have all been acquired sufficiently close in time that negligible drift in the dark spectrum will have occurred.

static array<double> **calculateRelativeIrradiance** (double *colorTemperature*,

array< double >^ *referenceDark*,

array< double >^ *reference*,

array< double >^ *sampleDark*,

array< double >^ *sample*,

array< double >^ *wavelengths*,

int% *errorCode*) [static]

- This function computes Relative Irradiance. Irradiance is the amount of energy at each wavelength emitted from a radiant sample. Before you can compute Relative Irradiance you must take a reference spectrum of a known color temperature. Additionally, you must obtain a dark spectrum by removing the fiber from the reference lamp and preventing light from entering it. Relative irradiance is a comparison of the fraction of energy the sample emits and the energy the sampling system collects from a lamp with a blackbody energy distribution (normalized to 1 at the energy maximum).

static array<double> ***calculateTransmission*** (array< double >^ *dark*,

array< double >^ *reference*,

array< double >^ *sample*,

int% *errorCode*) [static]

- This function calculates the transmission spectrum for a specified sample, reference and dark spectrum. This function should be used when the dark spectrum to be subtracted from the reference and the sample is the same -- i.e., the dark, reference and sample spectra have all been acquired sufficiently close in time that negligible drift in the dark spectrum will have occurred. The arrays used to store each of the spectra must be large enough to contain all of the pixels in each spectrum and the specified lengths for each spectrum must be identical.

static array<double> ***calculateTransmission*** (array< double >^ *referenceDark*,

array< double >^ *reference*,

array< double >^ *sampleDark*,

array< double >^ *sample*,

int% *errorCode*) [static]

- This function calculates the transmission spectrum for a specified sample spectrum and a specified reference spectrum with different dark correction spectra for the reference and sample. This function should be used when the dark spectra to be subtracted from the reference and the sample are different e.g. when the reference and sample spectra have been taken sufficiently far apart in time to justify a dark spectrum to be taken for both the reference and sample.

static array<double> ***kubelkaMunkProcessReflectionSpectrum*** (array< double >^ *inputReflectionValuesArray*,

int% *errorCode*) [static]

- Processes a supplied reflection spectrum into an array of K/S values as per the Kubelka-Munk calculation.

static cli::array<double> ***applyNonunityCorrectionFactors*** (cli::array< double > ^ *inputPixelsArray*,

cli::array< double > ^ *inputInterpolatedFactorsArray*,

cli::array< double > ^ *inputDarkSpectrumArray*,

int% *errorCode*) [static]

- Calculates the nonuniformity correction to compensate for a reflection standard that is not perfectly white. The correction factors should be obtained from the function omapi_pnuc_get_interpolated_correction_factors.

static cli::array<double> ***getInterpolatedCorrectionFactors*** (cli::array< double > ^ *inputCorrectionFactorsArray*,
cli::array< double > ^ *inputCorrectionFactorWavelengthArray*,
cli::array< double > ^ *inputSourceWavelengthArray*,
int% *errorCode*) [static]

- Returns the interpolated correction factors for a spectrum by performing a cubic spline on the standard white wavelength, the white source wavelengths, and the correction factors.

Appendix B - Error Codes

The following are error codes that may be returned from a function call.

Error Code	Description	Error Code	Description
0	Successful/no error	5	Divide by zero
1	Null pointer	6	License not checked
2	Array size error	7	Full license exceeded
3	Array length error	8	Trial license expired
4	Invalid argument	9	Perpetual license version mismatch

Appendix C - Improving Performance

Windows is not a real-time operating system and cannot guarantee a level of responsiveness. But there are a few things you can do to improve the performance of your application.

- Often the problem is that some *other* application (or Windows service) is performing activity that interferes with the speed of your application. Try setting the priority of your OceanMathART application to "RealTime". To do this:
  1. Type control+alt+delete to bring up the Windows Task Manager.
  2. Select the **Processes** tab.
  3. Right-click on your OceanMathART application and choose **Set Priority** | **RealTime**.
- Determine what applications and services are running on your PC and shut down all unnecessary applications. If you have a backup utility such as Carbonite, you should pause it. Check your anti-virus application to see if it is configured in some way that might result in bursts of disk I/O.
- If bursts of disk I/O are interfering with your application, there is a good chance this disk I/O is due to "page faults". Page faults occur when Windows does not have enough RAM/memory to run all applications that are currently active. So Windows “borrows” some disk space to "simulate" additional RAM. If this causes your OceanMathART application a problem, then the solution is to either shut down as many applications as possible, or to install additional RAM.
- Try running your application on another PC that has nothing else installed.

Unlock the Unknown

Ocean Insight exists to end guessing. We equip humanity with technology and data to make precisely informed decisions providing transformational clarity for human advancement in health, safety, and the environment.

Questions?
Chat with us at **OceanInsight.com**.
info@oceaninsight.com • **US** +1 727-733-2447
EUROPE +31 26-3190500 • **ASIA** +86 21-6295-6600