AVSIRRAD-DLL

Dynamic Link Library for 32 bit Windows Applications

Version 3.2.0.0

Avantes B.V.

Oude Apeldoornseweg 28
NL-7333 NS Apeldoorn
The Netherlands
Tel: +31-313-670170
Fax: +31-313-670179

Web: www.avantes.com
Email: info@avantes.com

Microsoft, Visual C++, Visual Basic, Visual C# , Windows, Windows 95/98/Me, Windows NT/2000/XP and Microsoft Office are registered trademarks of the Microsoft Corporation. Windows Vista and Windows 7 are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
Delphi and C++Builder are trademarks of CodeGear, a subsidiary of Embarcadero Technologies
LabVIEW is a trademark of the National Instruments Corporation

Contents

0 INSTALLATION 5

Installation Dialogs 5

Launching the software 6

1 INTRODUCTION 7

2 HOW TO GET THE IRRADIANCE AND WAVELENGTH DATA 7

3 BACKGROUND 8

3.1 COLORIMETRY 8

3.2 COLOR RENDERING INDEX 10

3.3 RADIOMETRY 10

Peak Measurements 14

3.4 PHOTOMETRY 14

4 AVSIRRAD DLL EXPORTS 15

4.1 EXPORTED DATA TYPES 15

4.2 EXPORTED FUNCTIONS 15

4.2.1 Color_GetColorOfLightParam 16

4.2.2 Color_GetLedParamFromxy 17

4.2.3 Color_GetColorRenderingIndex 18

4.2.4 Color_GetCRIHires 19

4.2.5 Color_ InterpolateT4_5 20

4.2.6 Color_InterpolateCRI_CQStable 21

4.2.7 Radio_GetIrradiance 22

4.2.8 Radio_GetEnergyPerCM2 23
4.2.9 Radio_GetReceivedPower 24
4.2.10 Radio_GetReceivedEnergy 25
4.2.11 Radio_GetRadiantIntensity 26
4.2.12 Radio_GetRadiantEnergy 27
4.2.13 Radio_CalcEmittedPower 28
4.2.14 Radio_CalcEmittedEnergy 29
4.2.15 Radio_IspGetEmittedPower 30
4.2.16 Radio_IspGetEmittedEnergy 31
4.2.17 Radio_GetPhotonFluxPerM2 32
4.2.18 Radio_GetPhotonFlux 33
4.2.19 Radio_GetPhotonsPerM2 34
4.2.20 Radio_GetPhotons 35
4.2.21 Radio_GetPeak 36
4.2.22 Photo_GetIlluminance 38
4.2.23 Photo_GetReceivedLuminousFlux 39
4.2.24 Photo_GetLuminousIntensity 40
4.2.25 Photo_CalcLuminousFlux 41
4.2.26 Photo_IspGetLuminousFlux 42

5 EXAMPLE SOURCE CODE 43

0 Installation

AVSIRRAD-DLL is a 32-bit dynamic link library and can be installed under the following operating systems:

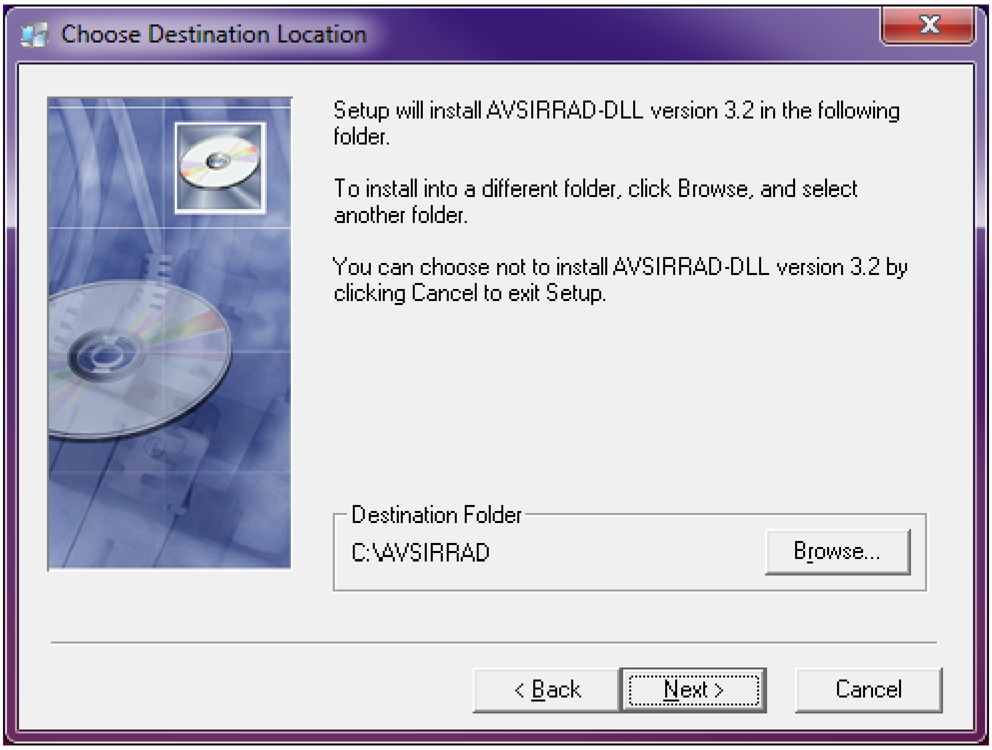
- Windows 95/98/Me
- Windows NT/2000
- XP/Vista/Windows7 x32 (32-bit O/S)
- XP/Vista/Windows7 x64 (64-bit O/S)

The installation program can be started by running the file “setup.exe” from the CD-ROM.

Installation Dialogs

The setup program will check the system configuration of the computer. If no problems are detected, the first dialog is the “Welcome” dialog with some general information.

In the next dialog, the destination directory for the AvaSoft software can be changed. The default destination directory is C:\AVSIRRAD. If you want to install the software to a different directory, click the Browse button, select a new directory and click OK. If the specified directory does not exist, it will be created.



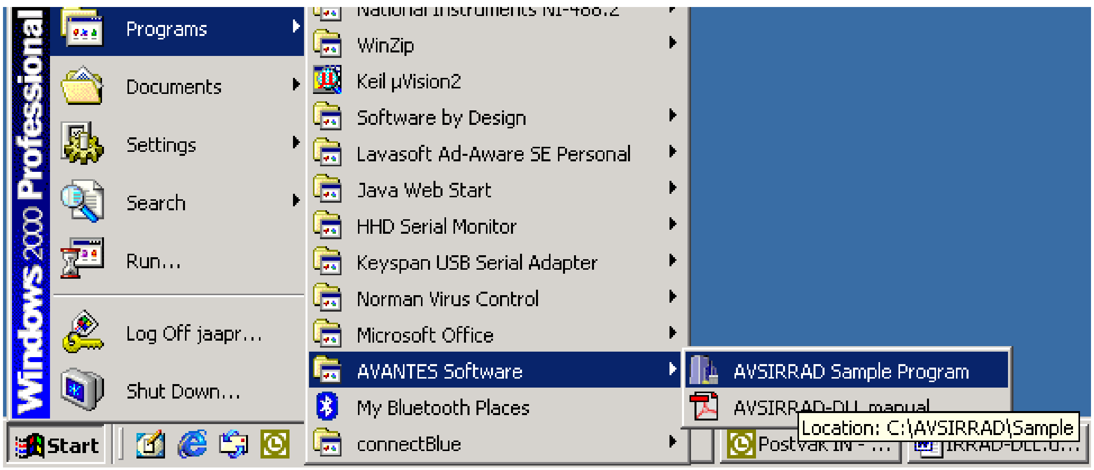
This installation is password protected. Enter the following password to proceed with the installation:

dwcdwpur

Launching the software

This AVSIRRAD-DLL manual and the full featured Borland C++ Builder example program can be launched from the Windows Start Menu, as shown in the figure below. The source code of the available example programs can be found in the “Examples” folder.

Included samples are:
- a full featured sample in Borland C++ Builder (version 5).
- A simple CRI sample in Borland Delphi (version 6).
- three simple samples in Labview 8.5, together with Labview vi’s for all functions in the DLL. Vi’s for a number of older versions of Labview are available on request.
- A simple CRI sample in Microsoft Visual C# (version 2008)
- A simple CRI sample in Microsoft C++ (version 2008) / Qt (version 4.8.2)



1 Introduction

The Irradiance-DLL includes functions for calculating radiometric, photometric and colorimetric parameters. Also LED specific parameters can be calculated such as: Dominant Wavelength, Purity, Central Wavelength, Centroïd, FWHM.

The input for most of the functions is an array of irradiance values (µWatt/cm²) per wavelength (nm):

```
const int MAX_PIXELS = 3648;

typedef struct
{
    double  wl[MAX_PIXELS] ;
    double  intensity[MAX_PIXELS];
} DataType;
```

2 How to get the irradiance and wavelength data

If the data comes from an AvaSpec spectrometer, the AVS_GetLambda function in the as161.dll (USB1 platform) or as5216.dll (USB2 platform) can be used to get the wavelength for each pixel. The function AVS_GetScopeData is used to collect the measured A/D Counts from the spectrometer. To be able to convert the intensity in A/D Counts into an intensity in µW/cm², an intensity calibration is required. The intensity calibration contains the data transfer function for each pixel. The data transfer function is used to convert the scope data (A/D Counts) into irradiance data (in µWatt/cm^2) . To be able to calculate the transfer function, a calibrated light source, with known output (in µWatt/cm^2/nm) needs to be available.

When saving the A/D Counts with the reference light source on and off, we know the relation between the A/D Counts and µWatt/cm^2:

$$\left(\frac{Caldata_n}{refcal_n - darkcal_n}\right)$$

$Caldata_n$	= Intensity of the calibrated light source at pixel n (in µWatt/cm^2) from lampfile
$refcal_n$	= A/D Counts at pixel n that were saved with the reference light source on
$darkcal_n$	= A/D Counts at pixel n that were saved with the reference light source off

When measuring the A/D counts received from a light source different from the calibrated light source (but of course with the same fiber optic cable and diffuser), this relation can be used to measure the intensity at every pixel n (in µWatt/cm^2). If $sample_n$ is the measured A/D counts at pixel n when looking at the sample light source, and $dark_n$ is the measured A/D Counts with the sample light source off, then the equation for intensity I_n (in µWatt/cm^2) becomes:

$$I_n = Caldata_n * \left(\frac{sample_n - dark_n}{refcal_n - darkcal_n}\right)$$

If during the intensity calibration a different integration time was used (e.g. 100ms) from the integration time during the sample measurements (e.g. 2 ms), a factor needs to be added to the equation to compensate for this. In the example the factor is 100/2 = 50.

Calculating the intensity (in µWatt/cm^2) from the measured sample spectrum (in A/D Counts) can therefore be done by the following equation:

$$I_n = Caldata_n * \left(\frac{sample_n - dark_n}{refcal_n - darkcal_n} \right) * factor$$

The irradiance (in µWatt/cm^2) and the wavelength (in nm) data for each pixel n will be the input for calculating the colorimetric, photometric, radiometric and LED parameters.

3 Background

3.1 Colorimetry

The color of light can be expressed by the chromaticity coordinates x, y and z. These chromaticity coordinates are obtained by taking the ratios of the tristimulus values (X, Y and Z) to their sum:

$$x = \frac{X}{(X+Y+Z)} \qquad y = \frac{Y}{(X+Y+Z)} \qquad z = \frac{Z}{(X+Y+Z)}$$

The tristimulus values X, Y and Z are computed by:

$$X = k * \sum I_\lambda * x_\lambda \qquad Y = k * \sum I_\lambda * y_\lambda \qquad Z = k * \sum I_\lambda * z_\lambda$$

where:

k	=	constant (= $1/(\Sigma y_\lambda)$)
$I\lambda$	=	Spectral irradiance at wavelength λ
$x\lambda$, $y\lambda$, $z\lambda$	=	CIE 1931 or 1964 Standard Observer value (2 or 10 degrees) at wavelength λ

The tristimulus values X, Y and Z and the spectral irradiance are computed in a wavelength range from 380 nm to 780 nm, using a 1 nm interval.

The CIE1960 UCS color coordinates u and v are calculated by:

$$u = \frac{4x}{(-2x+12y+3)} \qquad v = \frac{6y}{(-2x+12y+3)}$$

The equation that is used for calculating the color temperature is emperical and assumes a black body radiator:

p = ((x-0.332)/(y-0.1858))
Color Temp =5520.33-(6823.3*p)+(3525*p^2)-(449*p^3)

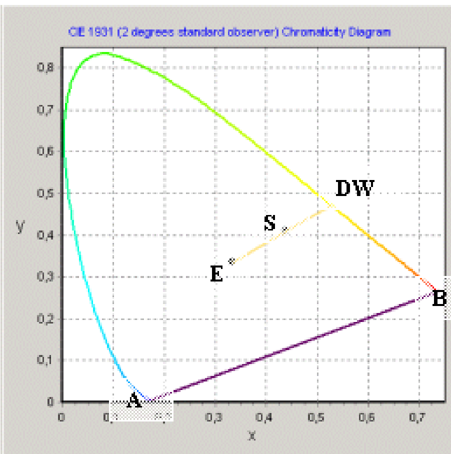
In LED measurements, the Dominant Wavelength and Purity (also known as HelmHolz coordinates) are often used to describe a color. The Dominant Wavelength can be calculated for a measured sample point S with chromaticity coordinates (Sx,Sy) by drawing a straight line from the midpoint in the chromaticity diagram (E with x=y=0.333) through S towards the edge of the Chromaticity Diagram (spectrum locus). The points at the spectrum locus correspond with a wavelength and the interception of the straight line through E and S with the locus is called the Dominant Wavelength (DW).

Purity, is the distance from the midpoint (E) to the sample point (S), divided by that from the midpoint (E) to the spectrum locus (DW):

Purity = (E-S) / (E-DW)

The method described above is used for all colors with a Dominant Wavelength from 380 to 699 nanometer. If the x,y coordinates are in the triangle area encompassed by the 3 points E, A and B, then the Dominant Wavelength can not be calculated because the interception point through E and S with the spectrum locus (between A and B) does not correspond with a wavelength. In that case the Complementary Dominant Wavelength (CDW) is used. The line from E through S is extended backward in order to determine the Complementary Dominant Wavelength (CDW).

The method described above is used for all colors with a Dominant Wavelength from 380 to 699 nanometer. If the x,y coordinates are in the triangle area encompassed by the 3 points E, A and B, then the Dominant Wavelength can not be calculated because the interception point through E and S with the spectrum locus (between A and B) does not correspond with a wavelength. In that case the Complementary Dominant Wavelength (CDW) is used. The line from E through S is extended backward in order to determine the Complementary Dominant Wavelength (CDW).



3.2 Color Rendering Index

The Color Rendering Index (CRI) is a measure of the color rendering properties of a light source. The CIE has defined 14 standard color samples and the CRI is defined as the mean value of the color rendering values of the first 8 of these standard CIE samples.
A CRI of 100 indicates that the sample sources have identical color coordinates under the light source to be tested and the reference light source. A CRI value of 50 is assigned to the CIE standard warm white fluorescent lamp.
The CRI calculation method use in the AVSIRRAD DLL is based on the CIE 13.3-1995 standard.
A reference light source is calculated on the basis of the color temperature of the light source to be tested. The color temperature calculation is done with the 'Color_GetColorOfLightParam' function. The algorithm used here is the one by McCamy.
For color temperatures under 5000K, a standard Planckian radiator is calculated with a color temperature that equals the color temperature of the sample light source. For color temperatures between 5000K and 1000000K, a standard daylight distribution is calculated, again with an identical color temperature.
The 14 standard CIE samples are then 'illuminated' with both light sources, and from the difference in chromaticity values the 14 Special Rendering Indices are calculated, the first 8 of which yield the CRI.
The DLL contains two functions that calculate a CRI:
The first one (Color_GetColorRenderingIndex) is meant for continuous spectra, like generated by halogen light sources. The input spectrum is normalized to 380-780 nm, 5nm increment, which is the way the tables from the CIE 13.3-1995 standard are also listed.
If you measure the spectrum of a compact fluorescent light source with a high-resolution spectrometer, this normalisation can completely miss some of the narrow peaks, resulting in a CRI value that is way off.
For this reason, we have added a second function (Color_GetCRIHires), which uses all pixels in the measured spectrum, and interpolates all CIE tables that are used in the calculation.
Since the wavelength values of the pixels of your spectrometer will not change during your measurement, the interpolations are not hidden in the DLL, but implemented as separate DLL calls. (Color_InterpolateT4_5 and Color_InterpolateCRI_CQStable)
This way, your program will not have to perform the same interpolations over and over again for each measurement, but can store the values needed in local parameters.

3.3 Radiometry

The radiometric parameters can be grouped into three categories:

- Radiant Flux [µWatt]: The radiant flux is the total optical power emitted from a source in all directions. The best way to measure the power emitted by a source is to measure the source inside an integrating sphere. This is often done when measuring LED's. It is also possible to calculate the flux of a source by measuring the irradiance at the surface of the diffuser (cosine corrector or integrating sphere sample port) at a certain distance from a light source. An important assumption in this calculation is that the source should be isotropic and the distance between diffuser and source should be greater than five times the largest dimension of the source (approximation of point source).
- Radiant Intensity [µWatt/sr]: The radiant intensity is the optical power per unit solid angle. It is used to quantify the optical power that is emitted by a source into a certain direction. Radiant intensity is calculated from the measured irradiance by multiplication with the square of the distance between source and diffuser surface. It is assumed that the source is a pointsource.

- Irradiance [µWatt/cm^2]: Irradiance is used to measure the power that is received by a surface.

Radiometric measurement can be done in different setups, like with fiber optic cosine corrector or integrating sphere. Both setups can be used to measure the irradiance spectrum received at the surface of the diffuser (cosine corrector or integrating sphere sample port) at a certain distance from a light source. When measuring at a certain distance from the source, the radiant intensity and flux can be calculated as described above. When measuring a light source inside an integrating sphere, the radiant flux can be measured, but radiant intensity and irradiance parameters cannot be measured.

Radiometric parameters calculated from the power distribution

The power distribution can be easily converted in an **energy distribution** by multiplying the power with the integration time. The result is the amount of energy that has been emitted or received during one integration time cycle.

Another radiometric parameter that can be calculated from the irradiance spectrum, is the **number of photons** that is received by a surface. Since the number of photons per nanometer is a huge number (even with very low light intensity), the number of Avogadro is used to express the number of photons in mols, or as in our application in µmols. The number of photons per nanometer can be calculated from the wavelength dependent photon energy, and the absolute light energy that is measured. A detailed description how this is done can be found at the next page.

The photon count distribution [µMol/(s.m^2.nm)] shows the photon flux received per square meter. Other photon count units that can be calculated from this are:

- [µMol/(s.nm)] photon flux received at diffuser surface
- [µMol/(m^2.nm)] photons received per square meter during one integration cycle
- [µMol/nm] photons received at diffuser surface during one integration cycle

How to convert a power distribution [µWatt/(cm^2.nm)] into a photon count distribution [µMol/(s.m^2.nm)]

Photon energy $E(\lambda) = h.c/\lambda$ e

Where:
h = Planck´s constant 6,626 068 76 x10^{-34}
c = velocity of light 2,998 x10^{8} m/s
λ = wavelength in meters

For example, the photon energy at 250 nm and at 1000 nm is:

$E(250) = (6{,}626 \times 10^{-34}.\ 2{,}998 \times 10^{8})/\ 250.10^{-9} = 7{,}946 \times 10^{-19}$ (Joule/photon) (1)

$E(1000) = (6{,}626 \times 10^{-34}.\ 2{,}998 \times 10^{8})/\ 1000.10^{-9} = 1{,}986 \times 10^{-19}$ (Joule/photon) (2)

1eV = 1,60207 x 10^{-19} Joule, so the photon energy expressed in eV/photon becomes:

$E(250) = 7{,}946 \times 10^{-19} / 1{,}60207 \times 10^{-19} = 4{,}9592$ (eV/photon) (3)

$E(1000) = 1{,}986 \times 10^{-19} / 1{,}60207 \times 10^{-19} = 1{,}2398$ (eV/photon) (4)

Suppose we measure 20 µWatt/cm^2 at a certain wavelength

20 µWatt/cm^2 = 20 µJoule/s/cm^2 = 0.2 Joule/s/m^2
= 0.2/(1,60207 x 10^{-19}) eV/s/m^2
= 1,248 . 10^{18} eV/s/m^2 (5)

Knowing the photon energy at 250 nm and at 1000 nm from (3) and (4), the number of photons that correspond with 20 µWatt/cm^2 at 250 nm and at 1000 nm can be calculated from (5) by:

for 250 nm : #photons = 1,248 . 10^{18} / 4,9592 = 2,517 . 10^{17} photons/s/m^2
for 1000 nm : #photons = 1,248 . 10^{18} / 1,2398 = 1,007 . 10^{18} photons/s/m^2

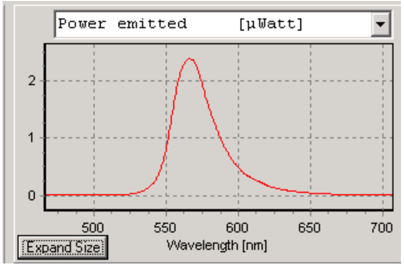
With 1 mol = 6,02308 x10^{23} (Number of Avogadro)
1 µmol = 6,02308 x10^{17}

So, the number of photons, expressed in µmol/s/m^2, when measuring 20 µWatt/cm^2 at wavelength 250 nm and also 20 µWatt/cm^2 at 1000 nm becomes:
for 250 nm : 2,517 . 10^{17} / 6,02308 .10^{17} = 0,418 µmol/s/m^2
for 1000 nm : 1,007 . 10^{18} / 6,02308 .10^{17} = 1,672 µmol/s/m^2

In the table below, the radiometric parameters that can be measured in AvaSoft are listed. Note that a wavelength range needs to be specified over which the parameter spectral output will be integrated. In the first column (Hardware setup), “inside sphere” refers to measurements that are done with the light source inside an integrating sphere, while “outside sphere or cc” refers to measurements that are done with a light source at a certain distance from the sphere or with a cosine corrector

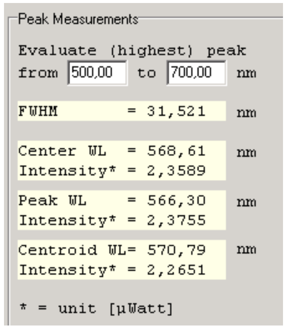
Hardware Setup	Parameter	Unit	Description
inside sphere	Radiant Flux (Power emitted)	µWatt	Total optical power emitted from a source
inside sphere	Energy emitted	µJoule	Total optical energy emitted from a source, calculated by multiplication of the power with the integration time
outside sphere or cc	Radiant Flux (Power emitted)	µWatt	Total optical power emitted from a source, calculated by multiplication of radiant intensity with the solid angle of the light source
outside sphere or cc	Energy emitted	µJoule	Total optical energy emitted from a source, calculated by multiplication of the power with the integration time
outside sphere or cc	Radiant Intensity	µWatt/sr	Optical power per unit solid angle, calculated by multiplication of irradiance with the square distance between point source and diffuser surface
outside sphere or cc	Radiant Energy	µJoule/sr	Total optical energy emitted from a source, calculated by multiplication of the radiant intensity with the integration time
outside sphere or cc	Power received	µWatt	Power received at diffuser surface
outside sphere or cc	Energy received	µJoule	Energy received at diffuser surface, calculated by multiplication of the power with the integration time
outside sphere or cc	Irradiance	µWatt/cm^2	Power received per square centimeter
outside sphere or cc	Energy/cm^2	µJoule/cm^2	Energy received per square centimeter, calculated by multiplication of irradiance with the integration time
outside sphere or cc	Photon Flux/m^2	µMol/(s.m^2)	Photons received per second and per square meter, see for calculation previous page
outside sphere or cc	Photon Flux	µMol/s	Photons received per second at diffuser surface
outside sphere or cc	Photons/m^2	µMol/m^2	Photons received per square meter during one integration time cycle
outside sphere or cc	Photons	µMol	Photons received at diffuser surface during one integration time cycle

Peak Measurements



A typical spectral power distribution of a (green) LED is shown in the figure at the right. A number of peak parameters can be calculated from this spectrum:

<u>FWHM and Center Wavelength</u>
The Full Width Half Maximum of a peak is the bandwidth (in nanometers) for which the intensity is higher than half of the maximum intensity of that peak. The Center Wavelength is the wavelength halfway between the left and right wavelength where the intensity is half of the maximum intensity.

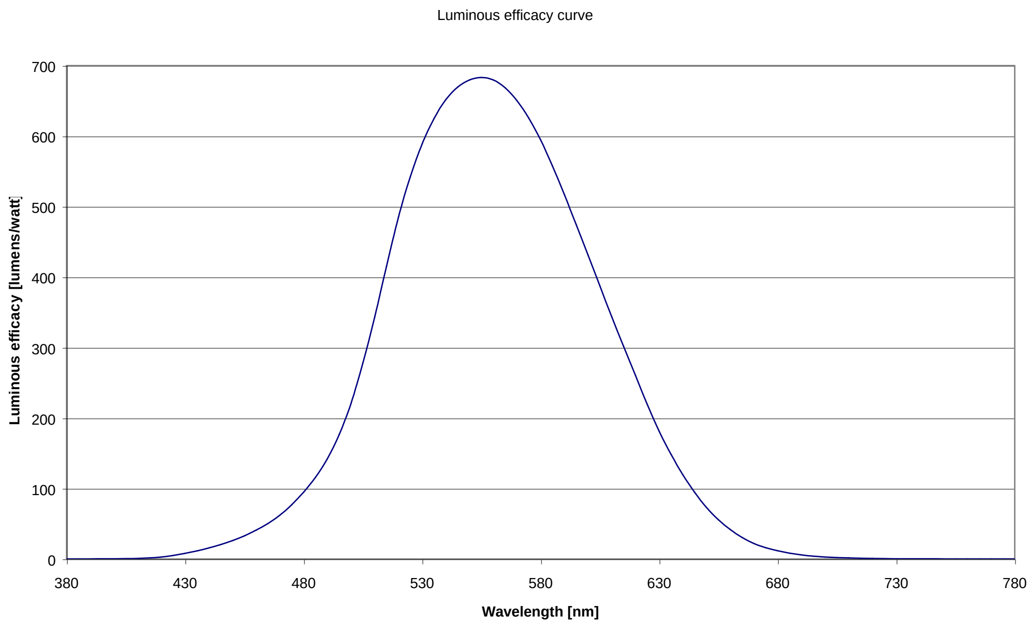


<u>Peak Wavelength</u>
Wavelength at the maximum spectral power

<u>Centroid Wavelength</u>
The total spectral power left and right from the centroid wavelength (integral) is the same.

3.4 Photometry

Photometry is the measurement of visible light. Unlike radiometry, it is not a purely physical measurement and is calculated considering a 'standard' human visual perception. This is attained by multiplying the radiometric data by the luminous efficacy curve (see figure below) and integrating the product over the visible range (380 – 780 nm).



The three categories that were defined for the radiometric parameters can also be used for the photometric parameters.

- The photometric equivalent for the Radiant Flux [µWatt] is the Luminous Flux, expressed in Lumens.
- The photometric equivalent for the Radiant Intensity [µWatt/sr] is the Luminous Intensity, expressed in Lumens/sr. This unit is equal to Candela.
- The photometric equivalent for Irradiance [$\mu Watt/cm^2$] is called Illuminance, expressed in $Lumens/m^2$. This unit is equal to Lux.

Since the geometry of the three categories is the same for radiometry and photometry, the same can be written about the hardware setup: luminous flux can be measured inside an integrating sphere. When measuring a source at a certain distance from the integrating sphere or cosine corrector, the luminous flux can be calculated, assuming that the source is an isotropic point source. The Luminous Intensity [Candela] and Illuminance [Lux] of a sphere can be measured outside the integrating sphere or with a cosine corrector.

4 AVSIRRAD dll exports

4.1 Exported data types

Several data-types used by the DLL that are necessary for the application interface are given below.

Type	Description	Remarks
DataType	typedef struct { double wl[3648] ; double intensity[3648]; } DataType;	Structure holding a 3648 array of double for wavelength and irradiance data
pixelarray	double pixelarray[3648]	array of 3648 doubles
unsigned char	unsigned character	8 bits value in the range 0...255
short int	signed integer	16 bits value in the range –32768…32767
double	double sized floating point number	64 bits value (15 digits precision)

4.2 Exported functions

The exported functions have been prefixed with Radio_ , Color_ or Photo_ to group them into functions for calculating respectively the radiometric, photometric or colorimetric parameters.

4.2.1 Color_GetColorOfLightParam

Function:	**short int Color_GetColorOfLightParam**		
	(		
In :	**DataType *a_data,**		
	unsigned char a_CIEResolution,		
	unsigned char a_Observer,		
Out :	**double *a_smallx,**		
	double *a_smally,		
	double *a_smallz,		
	double *a_bigX,		
	double *a_bigY,		
	double *a_bigZ,		
	double *a_u,		
	double *a_v,		
	double *a_Colortemperature		
	)		

Description	Calculates color parameters.		
Parameters:	a_data	Pointer to input data	See for background information section 3.1
	a_CIEResolution	0 = 1nm interval 1 = 5nm interval	
	a_Observer	0 = 2 degrees standard observer 1 = 10 degrees standard observer	
	a_smallx,	pointer to color parameter x	
	a_smally,	pointer to color parameter y	
	a_smallz,	pointer to color parameter z	
	a_bigX,	pointer to color parameter X	
	a_bigY,	pointer to color parameter Y	
	a_bigZ,	pointer to color parameter Z	
	a_u,	pointer to color parameter u	
	a_v,	pointer to color parameter v	
	a_Colortemperature	pointer to colortemperature	
Remark:	Input data must include irradiance data for wavelength range from 380nm to 780nm		
Return:	0 on SUCCESS -1 wavelength range incorrect: range 380 nm to 780 nm needs to be included in a_data -2 intensity in irradiance array zero for all elements		

4.2.2 Color_GetLedParamFromxy

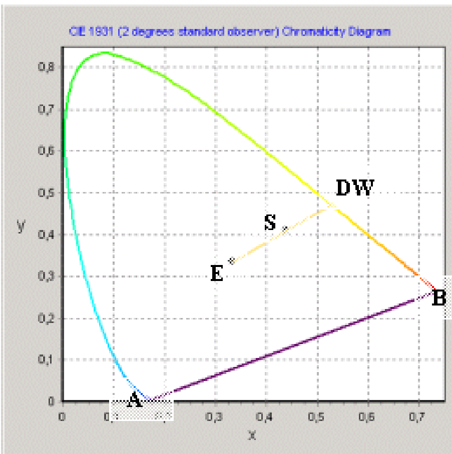
Function: **short int Color_GetLedParamFromxy**
(
In: **double a_smallx,**
double a_smally,
unsigned char a_Observer
Out: **double* a_dw,**
double* a_cdw,
double* a_purity
)

Description: Calculates LED specific color parameters from x,y. See section 3.1 for background information about (complementary) Dominant Wavelength and Purity

Parameters:

a_smallx	Color parameter x	
a_smally	Color parameter y	
a_dw	pointer to Dominant Wavelength = -1.0 if (x,y) in triangle (A,B,E) in figure below (see also section 3.1) In that case, the complementary dominant wavelength is calculated else dw within wavelength range 380 to 699 nm	
a_cdw	pointer to Complementary Dominant Wavelength = -1.0 if (x,y) not in triangle (A,B,E), in figure below (see also section 3.1). In that case, the dominant wavelength is calculated	
a_purity	pointer to purity	

Return: 0 if (x,y) inside chromaticity diagram resulting in a Dominant Wavelength in the visible (380-699nm) range
1 if (x,y) inside chromaticity diagram in triangle (A,B,E), resulting in a Complementary Dominant Wavelength
-3 if (x,y) outside chromaticity diagram



4.2.3 Color_GetColorRenderingIndex

Function:	**short int Color_ GetColorRenderingIndex**		
	(		
In :	**DataType *a_data,**		
	unsigned char a_CIEResolution,		
	unsigned char a_Observer,		
Out :	**double *R_Values,**		
	double *CRI,		
	)		
Description	Calculates CRI parameter		
Parameters:	a_data	Pointer to input data	See for background information section 3.1
	a_CIEResolution	0 = 1nm interval 1 = 5nm interval	
	a_Observer	0 = 2 degrees standard observer 1 = 10 degrees standard observer	
	R_Values,	pointer to array of 14 R values	
	CRI,	pointer to CIE Color Rendering Index	
Remark:	Input data must include irradiance data for wavelength range from 380nm to 780nm		
Return:	0 on SUCCESS -1 wavelength range incorrect: range 380 nm to 780 nm needs to be included in a_data -2 intensity in irradiance array zero for all elements -3 Calculated color temperature outside range 1..1000000 K -4 See -1 for calculated reference light source -5 See -2 for calculated reference light source -6 Input spectrum could not be normalized to 380-780nm, 5nm increments -7 See -1 for irradiation of CIE standard samples with test light source -8 See -2 for irradiation of CIE standard samples with test light source -9 See -1 for irradiation of CIE standard samples with calculated reference light source -10 See -2 for irradiation of CIE standard samples with calculated reference light source		

4.2.4 Color_GetCRIHires

Function:	**short int Color_ GetCRIHires**		
	(		
In :	**DataType *a_data,**		
	unsigned short a_NumPoints,		
	unsigned char a_Observer,		
	unsigned char a_samplenumber,		
	pixelarray *a_xarray,		
	pixelarray *a_yarray,		
	pixelarray *a_zarray,		
	pixelarray *a_CRItable,		
Out :	**double *a_Rvalue**		
	)		
Description	Calculates CRI parameter for spectra with narrow peaks using high res spectrometers		
Parameters:	a_data	Pointer to input data	See for background information section 3.1
	a_NumPoints	Number of points in a_data	
	a_Observer	0 = 2 degrees standard observer 1 = 10 degrees standard observer	
	a_samplenumber	Number of standard CRI sample (1-14)	
	a_xarray a_yarray a_zarray	Pointers to arrays that contain interpolated CIE tables T4 / T5	
	a_CRItable	Pointer to array that contains interpolated CIE CRI table	
	a_Rvalue	Pointer to calculated R value	
Remark:	Input data must include irradiance data for wavelength range from 380nm to 780nm		
Return:	0 on SUCCESS -1 wavelength range incorrect: range 380 nm to 780 nm needs to be included in a_data -2 intensity in irradiance array zero for all elements -3 Calculated color temperature outside range 1..1000000 K -4 See -1 for calculated reference light source -5 See -2 for calculated reference light source -7 See -1 for irradiation of CIE standard samples with test light source -8 See -2 for irradiation of CIE standard samples with test light source -9 See -1 for irradiation of CIE standard samples with calculated reference light source -10 See -2 for irradiation of CIE standard samples with calculated reference light source		

4.2.5 Color_ InterpolateT4_5

Function:	**short int Color_ InterpolateT4_5**
	(
In :	**DataType *a_data,**
	unsigned short a_NumInter,
	unsigned char a_Observer,
Out :	**pixelarray *a_xarray,**
	pixelarray *a_yarray,
	pixelarray *a_zarray
	)

Description Interpolates CIE table T4 /T5 for spectra with narrow peaks using high res spectrometers

Parameters:	a_data	Pointer to input data	See for background
	a_NumInter	Number of points in a_data	information section 3.1
	a_Observer	0 = 2 degrees standard observer	
		1 = 10 degrees standard observer	
	a_xarray	Pointers to arrays that contain	
	a_yarray	interpolated CIE tables T4 / T5	
	a_zarray		
Remark:			
Return:	0 on SUCCESS		
	1: X-values of the data points not unique		
	2: X-values of the data points not in ascending order		
	3: a_NumInter < 2		

4.2.6 Color_InterpolateCRI_CQStable

Function:	**short int Color_InterpolateCRI_CQStable**
	(
In :	**DataType *a_data,**
	unsigned short a_NumInter,
	unsigned char a_table,
	unsigned char a_CRI_CQSsample,
Out :	**pixelarray *a_tabelarray**
	)

Description	Interpolates CIE table T4 /T5 for spectra with narrow peaks using high res spectrometers		
Parameters:	a_data	Pointer to input data	See for background
	a_NumInter	Number of points in a_data	information section 3.1
	a_table	0 = CRI table	
		1 = CQS table (not implemented)	
	a_CRI_CQSsample	Number of standard sample	
		CRI: 1-14	
		CQS: 1-15	
	a_tablearray	interpolated CIE CRI table	
		(CQS not implemented yet)	
Remark:			
Return:	0 on SUCCESS		
	1: X-values of the data points not unique		
	2: X-values of the data points not in ascending order		
	3: a_NumInter < 2		

4.2.7 Radio_GetIrradiance

Function:	**short int Radio_GetIrradiance**		
	(		
In :	**DataType *a_data,**		
	double a_startwav,		
	double a_endwav,		
Out :	**double *a_integral**		
	)		
Description	Calculates Power received per square centimeter over the specified wavelength range by taking the integral of the irradiance input data over that range		
Parameters:	a_data	Pointer to input data	
	a_startwav	Start Wavelength for integral calculation.	nm
	a_endwav	End Wavelength for integral calculation.	nm
	a_integral	Pointer to Received Power per square centimeter	$\mu Watt/cm^2$
Return:	0 on SUCCESS		
	-4 if a_startwav > a_endwav		
	-5 if a_startwav >= end wavelength range in a_data		
	-6 if a_endwav <= start wavelength range in a_data		
	-7 if a_endwav > end wavelength range in a_data		
	-8 if a_startwav < start wavelength range in a_data		

4.2.8 Radio_GetEnergyPerCM2

Function:	**short int Radio_GetEnergyPerCM2**		
	(		
In :	**DataType *a_data,**		
	double a_startwav,		
	double a_endwav,		
	double a_inttime,		
Out :	**double *a_integral**		
	)		
Description	Calculates Energy received per square centimeter over the specified wavelength range by taking the integral of the irradiance input data over that range, and multiplying with the integration time cycle		
Parameters:	a_data	Pointer to input data	
	a_startwav	Start Wavelength for integral calculation.	nm
	a_endwav	End Wavelength for integral calculation.	nm
	a_inttime	Integration time cycle	ms
	a_integral	Pointer to Received Energy per square centimeter	µJoule/cm^2
Return:	0 on SUCCESS		
	-4 if a_startwav > a_endwav		
	-5 if a_startwav >= end wavelength range in a_data		
	-6 if a_endwav <= start wavelength range in a_data		
	-7 if a_endwav > end wavelength range in a_data		
	-8 if a_startwav < start wavelength range in a_data		

4.2.9 Radio_GetReceivedPower

Function:	**short int Radio_GetReceivedPower**		
	(		
In :	**DataType *a_data,**		
	double a_startwav,		
	double a_endwav,		
	double a_surfacecm2,		
Out :	**double *a_integral**		
	)		
Description	Calculates Power received by detector surface (cosine corrector or sample port integrating sphere) over the specified wavelength range by taking the integral of the irradiance input data over that range, and multiplying with the surface in cm2.		
Parameters:	a_data	Pointer to input data	
	a_startwav	Start Wavelength for integral calculation.	nm
	a_endwav	End Wavelength for integral calculation.	nm
	a_surfacecm2	Detector surface cosine corrector or sample port integrating sphere	cm^2
	a_integral	Pointer to Received Power	μWatt
Return:	0 on SUCCESS		
	-4 if a_startwav > a_endwav		
	-5 if a_startwav >= end wavelength range in a_data		
	-6 if a_endwav <= start wavelength range in a_data		
	-7 if a_endwav > end wavelength range in a_data		
	-8 if a_startwav < start wavelength range in a_data		

4.2.10 Radio_GetReceivedEnergy

Function:	**short int Radio_GetReceivedEnergy**		
	(		
In :	**DataType *a_data,**		
	double a_startwav,		
	double a_endwav,		
	double a_surfacecm2,		
	double a_inttime,		
Out :	**double *a_integral**		
	)		
Description	Calculates Energy received by detector surface (cosine corrector or sample port integrating sphere) over the specified wavelength range by taking the integral of the irradiance input data over that range, and multiplying with the detector surface and integration time.		
Parameters:	a_data	Pointer to input data	
	a_startwav	Start Wavelength for integral calculation.	nm
	a_endwav	End Wavelength for integral calculation.	nm
	a_surfacecm2	Detector surface cosine corrector or sample port integrating sphere	cm^2
	a_inttime	Integration time cycle	ms
	a_integral	Pointer to Received Energy	µJoule
Return:	0 on SUCCESS		
	-4 if a_startwav > a_endwav		
	-5 if a_startwav >= end wavelength range in a_data		
	-6 if a_endwav <= start wavelength range in a_data		
	-7 if a_endwav > end wavelength range in a_data		
	-8 if a_startwav < start wavelength range in a_data		

4.2.11 Radio_GetRadiantIntensity

Function:	**short int Radio_GetRadiantIntensity**		
	(		
In :	**DataType *a_data,**		
	double a_startwav,		
	double a_endwav,		
	double a_distancecm,		
Out :	**double *a_integral**		
	)		
Description	Get optical power per unit solid angle, calculated by multiplication of irradiance with the square distance between point source and detector surface (cosine corrector or integrating sphere sample port)		
Parameters:	a_data	Pointer to input data	
	a_startwav	Start Wavelength for integral calculation.	nm
	a_endwav	End Wavelength for integral calculation.	nm
	a_distancecm	Distance between point source and detector surface (cosine corrector or integrating sphere sample port)	cm
	a_integral	Pointer to Radiant Intensity	µWatt/sr
Return:	0 on SUCCESS -4 if a_startwav > a_endwav -5 if a_startwav >= end wavelength range in a_data -6 if a_endwav <= start wavelength range in a_data -7 if a_endwav > end wavelength range in a_data -8 if a_startwav < start wavelength range in a_data		

4.2.12 Radio_GetRadiantEnergy

Function: **short int Radio_GetRadiantEnergy**
(
In : **DataType *a_data,**
double a_startwav,
double a_endwav,
double a_inttime,
double a_distancecm,
Out : **double *a_integral**
)

Description: Get energy per unit solid angle, calculated by multiplication of irradiance with the square distance between point source and detector surface (cosine corrector or integrating sphere sample port), and multiplying with integration time.

Parameters:

a_data	Pointer to input data	
a_startwav	Start Wavelength for integral calculation.	nm
a_endwav	End Wavelength for integral calculation.	nm
a_inttime	Integration time cycle	ms
a_distancecm	Distance between point source and detector surface (cosine corrector or integrating sphere sample port)	cm
a_integral	Pointer to Radiant Intensity	µWatt/sr

Return:
0 on SUCCESS
-4 if a_startwav > a_endwav
-5 if a_startwav >= end wavelength range in a_data
-6 if a_endwav <= start wavelength range in a_data
-7 if a_endwav > end wavelength range in a_data
-8 if a_startwav < start wavelength range in a_data

4.2.13 Radio_CalcEmittedPower

Function:	**short int Radio_CalcEmittedPower**		
	(		
In :	**DataType *a_data,**		
	double a_startwav,		
	double a_endwav,		
	double a_distancecm,		
	double a_steradians,		
Out :	**double *a_integral**		
	)		
Description	Get total optical power emitted from a source, calculated by multiplication of radiant intensity with the solid angle of the light source (isotropic pointsource).		
Parameters:	a_data	Pointer to input data	
	a_startwav	Start Wavelength for integral calculation.	nm
	a_endwav	End Wavelength for integral calculation.	nm
	a_distancecm	Distance between point source and detector surface (cosine corrector or integrating sphere sample port)	cm
	a_steradians	Solid angle of the light source (isotropic pointsource).	sr
	a_integral	Pointer to Emitted Power	μWatt
Return:	0 on SUCCESS		
	-4 if a_startwav > a_endwav		
	-5 if a_startwav >= end wavelength range in a_data		
	-6 if a_endwav <= start wavelength range in a_data		
	-7 if a_endwav > end wavelength range in a_data		
	-8 if a_startwav < start wavelength range in a_data		

4.2.14 Radio_CalcEmittedEnergy

Function:	**short int Radio_CalcEmittedEnergy**		
	(		
In :	**DataType *a_data,**		
	double a_startwav,		
	double a_endwav,		
	double a_inttime,		
	double a_distancecm,		
	double a_steradians,		
Out :	**double *a_integral**		
	)		
Description	Get total energy emitted from a source, calculated by multiplication of radiant intensity with the solid angle of the light source (isotropic pointsource) and multiplying with integration time.		
Parameters:	a_data	Pointer to input data	
	a_startwav	Start Wavelength for integral calculation.	nm
	a_endwav	End Wavelength for integral calculation.	nm
	a_inttime	Integration time cycle	ms
	a_distancecm	Distance between point source and detector surface (cosine corrector or integrating sphere sample port)	cm
	a_steradians	Solid angle of the light source (isotropic pointsource).	sr
	a_integral	Pointer to Emitted Energy	µJoule
Return:	0 on SUCCESS -4 if a_startwav > a_endwav -5 if a_startwav >= end wavelength range in a_data -6 if a_endwav <= start wavelength range in a_data -7 if a_endwav > end wavelength range in a_data -8 if a_startwav < start wavelength range in a_data		

4.2.15 Radio_IspGetEmittedPower

Function:	**short int Radio_IspGetEmittedPower**		
	(		
In :	**DataType *a_data,**		
	double a_startwav,		
	double a_endwav,		
	double a_ispsurfacecm2,		
Out :	**double *a_integral**		
	)		
Description	Calculates emitted power of source inside an integrating sphere, over the specified wavelength range by taking the integral of the irradiance input data over that range, and multiplying with the surface of the sample port in cm^2.		
Parameters:	a_data	Pointer to input data	
	a_startwav	Start Wavelength for integral calculation.	nm
	a_endwav	End Wavelength for integral calculation.	nm
	a_ispsurfacecm2	Surface of sample port integrating sphere	cm^2
	a_integral	Pointer to Emitted Power	µWatt
Return:	0 on SUCCESS		
	-4 if a_startwav > a_endwav		
	-5 if a_startwav >= end wavelength range in a_data		
	-6 if a_endwav <= start wavelength range in a_data		
	-7 if a_endwav > end wavelength range in a_data		
	-8 if a_startwav < start wavelength range in a_data		

4.2.16 Radio_IspGetEmittedEnergy

Function:	**short int Radio_IspGetEmittedEnergy**		
	(		
In :	**DataType *a_data,**		
	double a_startwav,		
	double a_endwav,		
	double a_inttime,		
	double a_ispsurfacecm2,		
Out :	**double *a_integral**		
	)		
Description	Calculates emitted energy of source inside an integrating sphere during one integration time cycle, over the specified wavelength range by taking the integral of the irradiance input data over that range, and multiplying with the surface of the sample port in cm^2.		
Parameters:	a_data	Pointer to input data	
	a_startwav	Start Wavelength for integral calculation.	nm
	a_endwav	End Wavelength for integral calculation.	nm
	a_inttime	Integration time cycle	ms
	a_ispsurfacecm2	Surface of sample port integrating sphere	cm^2
	a_integral	Pointer to Emitted Energy	µJoule
Return:	0 on SUCCESS		
	-4 if a_startwav > a_endwav		
	-5 if a_startwav >= end wavelength range in a_data		
	-6 if a_endwav <= start wavelength range in a_data		
	-7 if a_endwav > end wavelength range in a_data		
	-8 if a_startwav < start wavelength range in a_data		

4.2.17 Radio_GetPhotonFluxPerM2

Function:	**short int Radio_GetPhotonFluxPerM2**		
	(		
In :	**DataType *a_data,**		
	double a_startwav,		
	double a_endwav,		
Out :	**double *a_integral**		
	)		
Description	Calculates Photon Flux (µMol Photons/second) received per square meter over the specified wavelength range. See also section 3.2: How to convert a power distribution [µWatt/(cm^2.nm)] into a photon count distribution [µMol/(s.m^2.nm)]		
Parameters:	a_data	Pointer to input data	
	a_startwav	Start Wavelength for integral calculation.	nm
	a_endwav	End Wavelength for integral calculation.	nm
	a_integral	Pointer to Photon Flux per square meter	µMol/(s.m^2)
Return:	0 on SUCCESS		
	-4 if a_startwav > a_endwav		
	-5 if a_startwav >= end wavelength range in a_data		
	-6 if a_endwav <= start wavelength range in a_data		
	-7 if a_endwav > end wavelength range in a_data		
	-8 if a_startwav < start wavelength range in a_data		

4.2.18 Radio_GetPhotonFlux

Function:	**short int Radio_GetPhotonFlux**		
	(		
In :	**DataType *a_data,**		
	double a_startwav,		
	double a_endwav,		
	double a_surfacecm2,		
Out :	**double *a_integral**		
	)		
Description	Calculates Photon Flux (µMol Photons/second) received by detector surface (cosine corrector or sample port integrating sphere) over the specified wavelength range by multiplying the Photon Flux per m^2 with the detector surface. See also section 3.2: How to convert a power distribution [µWatt/(cm^2.nm)] into a photon count distribution [µMol/(s.m^2.nm)]		
Parameters:	a_data	Pointer to input data	
	a_startwav	Start Wavelength for integral calculation.	nm
	a_endwav	End Wavelength for integral calculation.	nm
	a_surfacecm2	Detector surface cosine corrector or sample port integrating sphere	cm^2
	a_integral	Pointer to Photon Flux	µMol/s
Return:	0 on SUCCESS -4 if a_startwav > a_endwav -5 if a_startwav >= end wavelength range in a_data -6 if a_endwav <= start wavelength range in a_data -7 if a_endwav > end wavelength range in a_data -8 if a_startwav < start wavelength range in a_data		

4.2.19 Radio_GetPhotonsPerM2

Function:	**short int Radio_GetPhotonsPerM2**		
	(		
In :	**DataType *a_data,**		
	double a_startwav,		
	double a_endwav,		
	double a_inttime,		
Out :	**double *a_integral**		
	)		
Description	Calculates Photons received per square meter over the specified wavelength range during one integration time cycle by multiplying the Photon Flux per m^2 with the integration time cycle. See also section 3.2: How to convert a power distribution [µWatt/(cm^2.nm)] into a photon count distribution [µMol/(s.m^2.nm)]		
Parameters:	a_data	Pointer to input data	
	a_startwav	Start Wavelength for integral calculation.	nm
	a_endwav	End Wavelength for integral calculation.	nm
	a_inttime	Integration time cycle	ms
	a_integral	Pointer to Received Photons per square meter	µMol/m^2
Return:	0 on SUCCESS -4 if a_startwav > a_endwav -5 if a_startwav >= end wavelength range in a_data -6 if a_endwav <= start wavelength range in a_data -7 if a_endwav > end wavelength range in a_data -8 if a_startwav < start wavelength range in a_data		

4.2.20 Radio_GetPhotons

Function: **short int Radio_GetPhotons**
(
In : **DataType *a_data,**
double a_startwav,
double a_endwav,
double a_surfacecm2,
double a_inttime,
Out : **double *a_integral**
)

Description: Calculates Photons received by detector surface (cosine corrector or sample port integrating sphere) over the specified wavelength range during one integration time cycle by multiplying the Photon Flux per m^2 with the detector surface and integration time. See also section 3.2: How to convert a power distribution [µWatt/(cm^2.nm)] into a photon count distribution [µMol/(s.m^2.nm)]

Parameters:	a_data	Pointer to input data	
	a_startwav	Start Wavelength for integral calculation.	nm
	a_endwav	End Wavelength for integral calculation.	nm
	a_surfacecm2	Detector surface cosine corrector or sample port integrating sphere	cm^2
	a_inttime	Integration time cycle	ms
	a_integral	Pointer to Received Photons	µMol

Return: 0 on SUCCESS
-4 if a_startwav > a_endwav
-5 if a_startwav >= end wavelength range in a_data
-6 if a_endwav <= start wavelength range in a_data
-7 if a_endwav > end wavelength range in a_data
-8 if a_startwav < start wavelength range in a_data

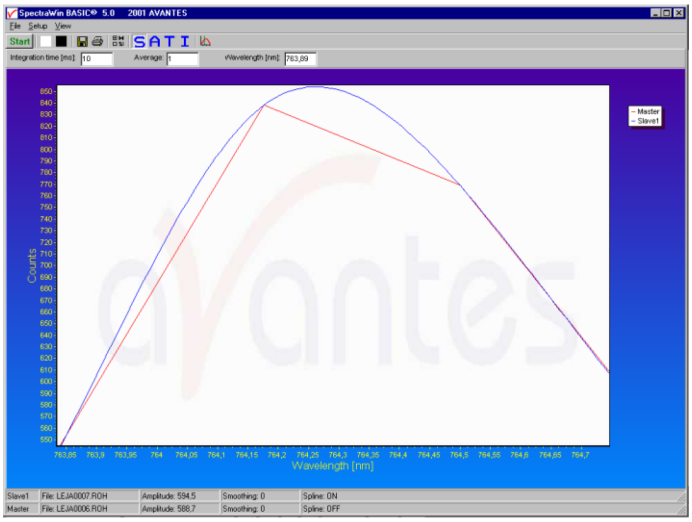
4.2.21 Radio_GetPeak

Function:	**short int Radio_GetPeak**
	(
In :	**DataType *a_data,**
	double a_startwav,
	double a_endwav,
	short int a_splinefactor,
Out :	**double *a_peaknm,**
	double *a_peakamp,
	double *a_fwhm,
	double *a_cwlnm,
	double *a_cwlamp,
	double *a_centroidnm,
	double *a_centroidamp
	)
Description	Calculates peak parameters (wavelength and amplitude) and full width half max for the highest peak in the wavelength range from a_startwav to a_endwav.

Parameters:	a_data	Pointer to input data	
	a_startwav	Start wavelength for peak search.	nm
	a_endwav	End wavelength for peak search.	nm
	a_splinefactor	Number of subpixels added with cubic spline algorithm. See comments below	
	a_peaknm	Wavelength for peak with highest intensity	nm
	a_peakamp	Intensity for peak with highest intensity	µWatt/cm^2
	a_fwhm	Full Width Half Max for peak with highest intensity	nm
	a_cwlnm	Center Wavelength for peak with highest intensity	nm
	a_cwlamp	Intensity at Center Wavelength	µWatt/cm^2
	a_centroidnm	Centroid Wavelength	nm
	a_centroidamp	Intensity at Centroid Wavelength	µWatt/cm^2

Return:
0 on SUCCESS
-4 if a_startwav > a_endwav
-5 if a_startwav >= end wavelength range in a_data
-6 if a_endwav <= start wavelength range in a_data
-7 if a_endwav > end wavelength range in a_data
-8 if a_startwav < start wavelength range in a_data

Cubic Spline Interpolation



In the figure at the right, the effect of spline interpolation is illustrated. The straight line shows the data for 4 pixels, connected by linear interpolation. The curved line is for these 4 pixels exactly the same as the straight line, but this time the cubic spline interpolation algorithm has been applied, resulting in data which is smooth in the first derivative and continuous in the second derivative.

The Splinefactor number is used to determine the number of subpixels added in the spline algorithm. For example, in the sample program, the “Load Simple Irrad Data” button will show 6 datapoints with 100nm interval at the x-axis. The point with the highest intensity without splining is found at 550nm (Intensity = 3.000). By recalculating the peak with splinefactor 20, the interval at the x-axis becomes 5 nm. The intensity for all new points is calculated by the spline algorithm. In the sample program, the peak wavelength has become 560nm and the intensity 3.011. By increasing the splinefactor to 100, all points with 1nm interval will be evaluated etc..

4.2.22 Photo_GetIlluminance

Function:	**short int Photo_GetIlluminance**
	(
In :	**DataType *a_data,**
	unsigned char a_CIEResolution,
Out :	**double *a_lux**
	)
Description	Calculates Illuminance in Lumens/m^2 (=Lux) by taking the integral of the irradiance input data between 380 nm and 780 nm and multiplying with the luminous efficacy curve (see section 3.3).

Parameters:	a_data	Pointer to input data	
	a_CIEResolution	0 = 1nm interval	
		1 = 5nm interval	
	a_lux	Pointer to Illuminance	Lumens/m^2 (=Lux)
Return:	0 on SUCCESS		
	-1 wavelength range incorrect: range 380 nm to 780 nm needs to be included in a_data		

Avantes website: http://www.avantes.com email: Info@avantes.com

4.2.23 Photo_GetReceivedLuminousFlux

Function:	**short int Photo_GetReceivedLuminousFlux**		
	(		
In :	**DataType *a_data,**		
	unsigned char a_CIEResolution,		
	double a_surfacecm2,		
Out :	**double *a_lumen**		
	)		
Description	Calculates Luminous Flux received by detector surface (cosine corrector or sample port integrating sphere) by multiplication of the illuminance (4.2.18) with the detector surface.		
Parameters:	a_data	Pointer to input data	
	a_CIEResolution	0 = 1nm interval 1 = 5nm interval	
	a_surfacecm2	surface cosine corrector or sample port integrating sphere	cm^2
	a_lumen	Pointer to Received Luminous Flux	Lumen
Return:	0 on SUCCESS -1 wavelength range incorrect: range 380 nm to 780 nm needs to be included in a_data		

4.2.24 Photo_GetLuminousIntensity

Function:	**short int Photo_GetLuminousIntensity**		
	(		
In :	**DataType *a_data,**		
	unsigned char a_CIEResolution,		
	double a_distancecm,		
Out :	**double *a_candela**		
	)		
Description	Get luminous flux per unit solid angle (Lumen/sr = Candela), calculated by multiplication of the illuminance (4.2.18) with the square distance between point source and detector surface (cosine corrector or integrating sphere sample port)		
Parameters:	a_data	Pointer to input data	
	a_CIEResolution	0 = 1nm interval 1 = 5nm interval	
	a_distancecm	Distance between point source and detector surface (cosine corrector or integrating sphere sample port)	cm
	a_candela	Pointer to Luminous Intensity	Lumen/sr = Candela
Return:	0 on SUCCESS -1 wavelength range incorrect: range 380 nm to 780 nm needs to be included in a_data		

4.2.25 Photo_CalcLuminousFlux

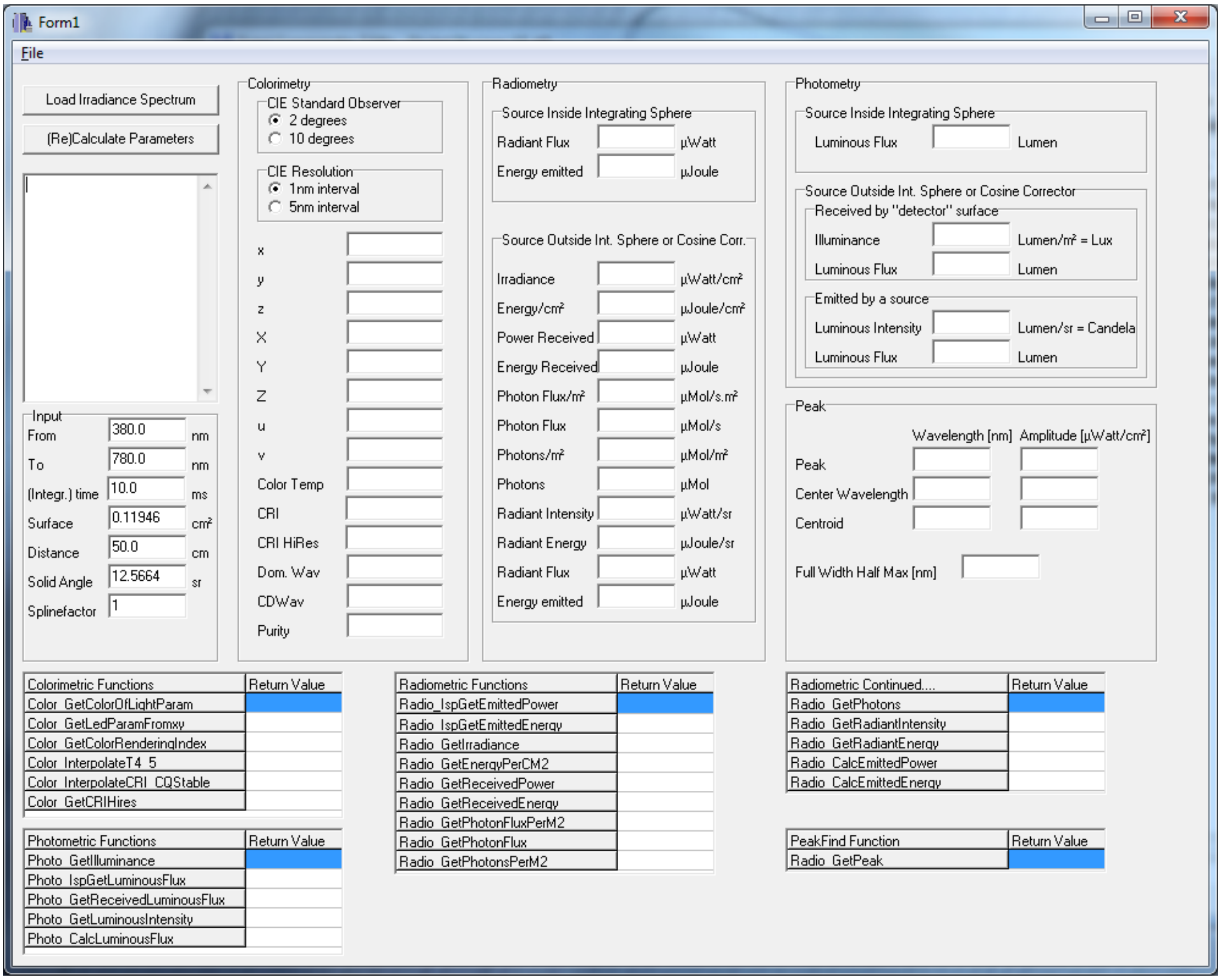
Function:	**short int Photo_CalcLuminousFlux**		
	(		
In :	**DataType *a_data,**		
	unsigned char a_CIEResolution,		
	double a_distancecm,		
	double a_steradians,		
Out :	**double *a_lumen**		
	)		
Description	Get total luminous flux emitted from a source, calculated by multiplication of luminous intensity with the solid angle of the light source (isotropic pointsource).		
Parameters:	a_data	Pointer to input data	
	a_CIEResolution	0 = 1nm interval 1 = 5nm interval	
	a_distancecm	Distance between point source and detector surface (cosine corrector or integrating sphere sample port)	cm
	a_steradians	Solid angle of the light source (isotropic pointsource).	sr
	a_lumen	Pointer to Emitted Luminous Flux	Lumen
Return:	0 on SUCCESS -1 wavelength range incorrect: range 380 nm to 780 nm needs to be included in a_data		

4.2.26 Photo_IspGetLuminousFlux

Function:	**short int Photo_IspGetLuminousFlux**		
	(		
In :	**DataType *a_data,**		
	unsigned char a_CIEResolution,		
	double a_ispsurfacecm2,		
Out :	**double *a_lumen**		
	)		
Description	Calculates emitted luminous flux of source inside an integrating sphere, by multiplication of the illuminance (4.2.18) with the surface of the integrating sphere sample port		
Parameters:	a_data	Pointer to input data	
	a_CIEResolution	0 = 1nm interval 1 = 5nm interval	
	a_ispsurfacecm2	Surface of sample port integrating sphere	cm^2
	a_lumen	Pointer to Emitted luminous flux	lumen
Return:	0 on SUCCESS -1 wavelength range incorrect: range 380 nm to 780 nm needs to be included in a_data		

5 Example source code

A full featured sample program with source code in Borland C++ Builder version 5 can be found in the folder “Examples\Borland C++”.



The sample program loads the irradiance data from a text file after pressing the “Load Irradiance Spectrum” button. The data is shown in two columns: the wavelength in nanometer in the first column and the irradiance data in µWatt/cm^2 in the second column.

By clicking the button “(Re)Calculate Parameters”, all functions described in section 4.2 will be called to calculate the parameters listed in the sample program. The input parameters at the left can be modified after which the output parameters can be recalculated.

Included in the “Examples” subdirectory are also:
- A simple CRI sample in Delphi, version 6.
- Some simple samples in Labview 8.5, complete with vi’s for all functions of the DLL. Vi’s for a number of older versions of Labview are available on request.

- A simple CRI sample in Visual c#, version 2008. Please note that not all function calls of the DLL have been translated to C#, just the ones used in the CRI sample.
- A simple CRI sample in Visual C++, version 2008, complete with complete header file and link library for Microsoft Visual Studio versions. The sample uses the Qt library, version 4.8.2, for its user interface.