Ghent University, Belgium

Diamond Light Source

xraylib

The definitive manual

Version 3.2.0

Bruno Golosio
Antonio Brunetti
Manuel Sanchez del Rio
Teemu Ikonen
Tom Schoonjans
Matthew Wormington
David Sagan

July 5, 2016

Contents

1 Introduction

Quantitative estimate of elemental composition by spectroscopic and imaging techniques using X-ray fluorescence requires the availability of accurate data of X-ray interaction with matter. Although a wide number of computer codes and data sets are reported in literature, none of them is presented in the form of freely available library functions which can be easily included in software applications for X-ray fluorescence. This work presents a compilation of data sets from different published works and an xraylib interface in the form of callable functions. Although the target applications are on X-ray fluorescence, cross sections of interactions like photoionization, coherent scattering and Compton scattering, as well as form factors and anomalous scattering functions, are also available.
xraylib provides access to some of the most respected databases of physical data in the field of X-rays. The core of xraylib is a library, written in ANSI C, containing over 40 functions to be used to retrieve data from these databases. This C library can be directly linked with any program written in C, C++ or Objective-C. Furthermore, the xraylib package contains bindings to several popular programming languages: Fortran 2003, Perl, Python, Java, IDL, Lua, Ruby, PHP and .NET, as well as a command-line utility which can be used as a pocket-calculator. Although not officially supported, xraylib has been reported to be useable from within Matlab and LabView.
The source code is known to compile and run on the following platforms: Linux, Mac OS X, Solaris, FreeBSD and Windows. It is very likely that xraylib will also work on other platforms: we would be grateful if you would report your successes in this regard. Please note that not all of the bindings are currently working on all platforms.
A paper was published concerning xraylib by A. Brunetti, M. Sanchez del Rio, B. Golosio, A. Simionovici and A. Somogyi, “A library for X-ray matter interaction cross sections for X-ray fluorescence applications”, Spectrochimica Acta B 59 (2004) 1725-1731.
This paper was recently superseded by a new manuscript, covering all features of xraylib upto version 2.15.0, written by T. Schoonjans, A. Brunetti, B. Golosio, M. Sanchez del Rio, V. A. Solé, C. Ferrero and L. Vincze, named The xraylib library for X-ray—matter interactions. Recent developments
You are kindly requested to include this paper in the reference list of your published work when you would decide to use xraylib for scientific purposes.
The xraylib source code is distributed under the BSD license. A copy of this license can be found in the source tarball.
New: we are currently building a website that allows users to perform simple queries from xraylib. This is possible through the PHP bindings that were added to the version 3.0.0. The code of the website can be found at http://github.com/tschoonj/xraylib-web. If you want to try out the website itself, visit http://lvserver.ugent.be/xraylib-web for a demo.

2 Installation instructions

2.1 Compilation from source

You can download the latest version of the *xraylib* source code from our Downloads repository.
I would recommend NOT to download the tagged version that is offered by Github, since this version does not come with the configure file. It is still possible though to compile the software based on this package, if you run `autoreconf -i` after unpacking the tarball, and then following the rest of the steps outlined in the installation section. This requires a complete installation of the GNU build tools Autoconf, Automake and Libtool (at least version 2.0!).
After downloading the tarball, unpack and compile the source code using the following commands:

```
gunzip xraylib-version.tar.gz

tar xvf xraylib-version.tar

cd xraylib-version

./configure

make
```

Optional, but recommended is to check if the compilation went well:

```
make check
```

Depending on the requested install location, it may be necessary to perform the installation as a user with administrative privileges

```
make install
```

configure and make will try to build as many bindings as possible (except for the PHP and Pascal bindings, which will only be built if the `--enable-php` and/or `--enable-pascal` options are passed). If some of these would be considered as redundant, it is possible to disable their compilation using `--disable-binding` options to configure. See `./configure --help` for a list of all options. Keep in mind though that for the command-line utility to work, the python bindings are essential.
The default behavior is for the libraries and the bindings to be installed in subdirectories of `--prefix=<your_installation_root>` (default is `/usr/local`). This could cause the user to be forced to define language specific environment variables before the bindings become usable. This problem can be circumvented by forcing

the bindings to be installed in the default locations as set when the specific interpreter (Perl, Python etc) was built. For this purpose, the `--enable-perl-integration`, `--enable-python-integration`, `--enable-ruby-integration` and `--enable-php-integration` options were added to the configure script. This is considered an advanced feature and is not recommended for standard installations.
More information can be found in the README and INSTALL files that are included in the package.

2.2 Linux

2.2.1 Fedora, Centos, Scientific Linux

To facilitate the installation on RPM based Linux distributions, the package includes a spec file which can be used to produce RPM packages for linux distributions that support them (Fedora, Red Hat etc). The developers have built 64-bit RPM packages of *xraylib* for the officially supported Fedora and Redhat EL/CentOS/Scientific Linux 6/7 distributions. These can be downloaded from the RPM repository that is hosted by the X-ray Microspectroscopy and Imaging research group of Ghent University. Access to this repository can be obtained as follows for Fedora distros:

```
su -c 'rpm -Uvh http://lvserver.ugent.be/yum/xmi-repo-key-fedora.noarch.rpm'
```

for Red Hat EL 6 based distributions:

```
su -c 'rpm -Uvh http://lvserver.ugent.be/yum/xmi-repo-key-6.0-1.el6.noarch.rpm'
```

and for Red Hat EL 7 based distributions:

```
su -c 'rpm -Uvh http://lvserver.ugent.be/yum/xmi-repo-key-7.0-1.el7.noarch.rpm'
```

The *xraylib* packages themselves can then be downloaded using yum:

```
su -c 'yum install xraylib xraylib-python xraylib-devel
xraylib-fortran xraylib-idl xraylib-lua xraylib-perl
xraylib-ruby'
```

Updates can be installed in a similar way:

```
su -c 'yum update xraylib xraylib-python xraylib-devel
xraylib-fortran xraylib-idl xraylib-lua xraylib-perl
xraylib-ruby'
```

The python packages for Fedora contain bindings to both python2 and python3 interpreters. Since neither RHEL 6 nor 7 offers python3 in its default repository, no python3 bindings are available in the rpm packages.

2.2.2 Debian and Ubuntu

Packages were created for Debian and Ubuntu. Currently the following flavors are supported: Debian Wheezy and Jessie, as well as a number of Ubuntu releases. For now only the python, fortran and perl bindings are supported. More bindings will be packaged in the future.
In order to access these packages using your favorite package manager, execute the following command to import our public key:

```
curl http://lvserver.ugent.be/apt/xmi.packages.key |
sudo apt-key add -
```

Next, add the package download location corresponding to your distribution to the /etc/apt/sources.list file (as root):
Debian Wheezy:

```
deb http://lvserver.ugent.be/apt/debian wheezy stable
deb-src http://lvserver.ugent.be/apt/debian wheezy stable
```

Debian Jessie:

```
deb http://lvserver.ugent.be/apt/debian jessie stable
deb-src http://lvserver.ugent.be/apt/debian jessie stable
```

Ubuntu Precise 12.04:

```
deb [arch=amd64] http://lvserver.ugent.be/apt/ubuntu precise stable
deb-src http://lvserver.ugent.be/apt/ubuntu precise stable
```

Ubuntu Trusty 14.04:

```
deb [arch=amd64] http://lvserver.ugent.be/apt/ubuntu trusty stable
deb-src http://lvserver.ugent.be/apt/ubuntu trusty stable
```

Ubuntu Wily 15.10:

```
deb [arch=amd64] http://lvserver.ugent.be/apt/ubuntu wily stable
deb-src http://lvserver.ugent.be/apt/ubuntu wily stable
```

Ubuntu Xenial 16.04:

```
deb [arch=amd64] http://lvserver.ugent.be/apt/ubuntu xenial stable
deb-src http://lvserver.ugent.be/apt/ubuntu xenial stable
```

When the sources.list file contains the correct download locations, update the apt cache by running:

```
sudo apt-get update
```

After this, one can install *xraylib* by executing the following command:

```
sudo apt-get install libxrl7 xraylib libxrl7-dev libxrlf03-7
libxrl-perl
```

2.3 Mac OS X

The recommended way to install *xraylib* on Mac OS X is through Homebrew. In a terminal type in the following command:

```
brew install homebrew/science/xraylib
```

Most bindings are supported, as can be seen when invoking the `info` option:

```
brew info homebrew/science/xraylib
```

We have also released a universal framework (32/64-bit Intel) for inclusion in Xcode projects. A zipfile containing the framework can be found in the Downloads folder. After unpacking, either drag it straight into your Xcode project or copy it into /Library/Frameworks.

2.4 Windows SDK

We have created two xraylib SDKs (32- and 64-bit) for the Windows operating system. To install, download the xraylib-{version}-win32.exe or xraylib-{version}-win64.exe file from the **Downloads repository** and double-click to activate the installation wizard.
Depending on the selected options, this will install the headers, examples, static libraries and the dynamic-link library (dll) into the appropriate folders. The SDK has been verified to work with several compilers: mingw32, Microsoft Visual C++, Intel C++ and Borland C++ (32-bit only). For this last compiler use the libxrl-{version}-bcc.lib file to link.
The SDK also contains bindings for IDL and Python , as well as bindings for the .NET framework.
The PYTHONPATH, IDL_PATH and IDL_DLM_PATH environment variables will be set correctly during installation making the bindings immediately available to the user.
When activating the Python bindings, the xraylib command-line utility will also be installed (called `xraylib-cli.py`), and its location will be added to the PATH environment variable.
The README.TXT file in the Lib subdirectory of the xraylib SDK explains the steps we used to create this package, as well as some advice on how to use it. The Visual C++ 2008 and 2010 Redistributable libraries will be installed if necessary.
It should be noted that both 32-bit and 64-bit SDKs can be installed alongside each other.

2.5 Windows command-line utility

With the release of version 2.14.0, a special version of the *xraylib* command-line utility was created for the Windows platform. The installer xraylib-{version}-win32-cli.exe can be found in the **Downloads repository**. It will install the executable

xraylib.exe, as well as some auxiliary files and the Visual C++ 2008 Redistributable libraries (if necessary). This executable was constructed using py2exe and contains the *xraylib* dll, Python bindings and scripts as well as the Python interpreter version 2.7.
After the installation, typing in: xraylib.exe at the command prompt will launch the application. The PATH environment variable should pick it up.

2.6 Java JAR with OSGi support

xraylib release 3.2.0 introduces a completely rewritten implementation of the Java bindings. Unlike the previously provided SWIG generated JNI bindings, this new version is rewritten in pure Java and is not linked to the xraylib shared library. The jar comes with an OSGi manifest and can therefore easily be integrated into software projects that support the OSGi standard (e.g. Eclipse).
The jar is distributed separately and requires Java Runtime Enviroment 7 or later. It depends on Apache Commons Math 3 for the use of its Complex class, which is used by the diffraction API.
This dependency will be downloaded automatically when building the jar with the configure script: add `--enable-java` to the argument list to do this. It is not shipped with the jar though!

3 Using xraylib

3.1 Using the C library with pkg-config

The source code contains a pkg-config file that will facilitate the use of xraylib in combination with your own software, written in C, C++ or Objective-C.
For example, you can compile program.c as follows:

```
gcc `pkg-config --cflags libxrl` program.c `pkg-config --libs libxrl`
```

This approach makes sure that the preprocessor finds the right headers and that the linker finds the library.
Include the xraylib headers with:

```
#include <xraylib.h>
```

Depending on the installation location of the library, it may be necessary to change the `LD_LIBRARY_PATH` and/or `DYLD_LIBRARY_PATH` environment variables. On some Linux systems it is also possible to add a file containing the xraylib library location to the `/etc/ld.so.conf.d` directory and running `ldconfig` if linking errors appear at runtime.
If xraylib was installed using our RPM or DEB packages, the required pkg-config file will only be available if the development package was also installed.
Use the `PKG_CONFIG_PATH` variable if the pkg-config file is not picked up: this will

probably be the case when compiling from source and using the default installation prefix.
pkg-config may also be used on the Windows platform, provided xraylib is compiled from source (usually with the MinGW compiler or using Cygwin).

3.2 Using the C library in Visual Studio (Windows)

Open the property pages of your project. Go to *Configuration Properties* → *C/C++* → *General*.
Add the Include folder of your xraylib installation to the *Additional IncludeDirectories* key.
With this, you should have access to the headers of xraylib when adding the following line to your C/C++ program:

```
#include <xraylib.h>
```

In order to have successful linking to the xraylib dll, go in the property pages of your project to *Configuration Properties* → *Linker* → *General*, and add the location of the Lib folder of your xraylib installation to the *Additional Library Directories* key.
The library itself can be specified from *Configuration Properties* → *Linker* → *Input*: add libxrl-7.lib to *Additional Dependencies*.

3.3 Using the framework in Mac OS X

After installing the framework in `/Library/Frameworks`, you may link to it from within your Xcode application, or from the commandline:

```
clang -framework xraylib -o test test.c
```

Include the header as:

```
#import <xraylib/xraylib.h>
```

3.4 Fortran bindings

In order to compile the xraylib Fortran bindings, it is necessary that the configure script finds a Fortran compiler on the system that supports the C interoperability features of the Fortran 2003 standard. Compilers known to do the job include: gfortran (starting with 4.4) , Intel Fortran and Sun Studio 12 .
Similar to the C library, it is possible to link in the Fortran bindings using a pkg-config file. Use the following syntax:

```
gfortran `pkg-config --cflags libxrlf03` program.f90 `pkg-config --libs libxrlf03`
```

In order to gain access to the xraylib module, include the following line in your Fortran source code:

```
USE :: xraylib
```

3.5 Python bindings

To include the xraylib python module in your own scripts add the following line:

```
import xraylib
```

xraylib 3.1.0 introduced an alternative implementation of the Python bindings, in which the arguments and return values are in fact numpy arrays (currently not supported on Windows). This is recommended whenever large amounts of calls to xraylib are required and performance becomes critical. This module, generated with Cython may be loaded from your script with:

```
import xraylib_np
```

Our testresults have shown that using this module one can obtain virtually identical performance as when using native C-code, or around 60 % faster compared to the default SWIG-generated bindings. The numpy based module does not export all functions from xraylib: those using strings and structs are not covered.
Both the default (SWIG generated) and the numpy (Cython generated) modules can be used alongside each other.
Depending on the install location of xraylib, it may be necessary to adjust the `PYTHONPATH` environment variable.

3.6 Perl bindings (Linux and Mac OS X)

To include the xraylib perl module in your own scripts add the following line:

```
use xraylib;
```

Depending on the install location of xraylib, it may be necessary to adjust the `PERL5LIB` environment variable.

3.7 IDL bindings

To use the IDL bindings, update the `IDL_PATH` and `IDL_DLM_PATH` environment variables to include `${prefix}/pro` and `${prefix}/dlm` respectively. If you want to take advantage of the xraylib constants, add the line

```
@xraylib
```

to your IDL scripts or your IDL startup script .

3.8 Lua bindings (Linux and Mac OS X)

In order to use the Lua bindings, update the `LUA_CPATH` environment variable to include `${libdir}/lua/x.y` (with x.y corresponding to your version of lua), and include the following line in your lua code:

```
require("xraylib")
```

3.9 Java bindings

To import the xraylib jar, include the lines

```
import com.github.tschoonj.xraylib.*;
import org.apache.commons.math3.complex.Complex;
```

in your java code. You may have to update the `CLASSPATH` environment variable to make sure that it points to the location of the xraylib class variables at compile and runtime.
In case of an error when calling an xraylib function (e.g. with invalid arguments) an`XraylibException` will be thrown instead of returning 0.0. The diffraction API can be used in two ways: using static methods (identical to the other bindings) or using an object oriented API.

3.10 .NET bindings (Windows only)

The .NET Framework can be installed on computers running Microsoft Windows operating systems. It supports multiple programming languages, including C#, VB.NET, C++/CLI, Pascal, Fortran and includes a large class library for that solves many common programming problems. Microsoft offers free versions of its Express Edition compilers from http://www.microsoft.com/express/
These were compiled using Visual Studio 2008 Standard and was built against .NET Framework Version 2. The binding consists of a single, mixed-mode assembly XrayLib.NET.dll written in C++/CLI. The assembly provides the interop between a managed XrayLib class and the native functions and types exposed by libxrl-7.dll. This combines the language interoperability of .NET with the performance of the native underlying functions.
A pre-built Release version of the assembly and an example program can be found in the bin folder together with a HTML Help v1 file.
To use XrayLib.NET.dll in Visual Studio:

1. Right-click on your project in the Solution Explorer
2. Click the References… menu item
3. Click the Add New Reference… button in the dialog box
4. Click the Browse tab of the Add Reference dialog box
5. Navigate to the xraylib Lib folder and select the XrayLib.NET.dll file
6. Click the OK buttons to add the assembly reference and close the dialog boxes
7. Assuming you are using C#, add the following line to the top of your source code file

```
using Science;
```

1. To create a instance of the XrayLib class that provides access to the XrayLib functionality, use the make the following call

```
XrayLib xl = XrayLib.Instance;
```

The class uses the static property Instance to implement a singleton pattern so that only a single instance of the XrayLib class is created and can be used throughout your program.

3.11 Ruby bindings (Linux and Mac OS X)

To include the xraylib ruby module in your own scripts add the following line:

```
require 'xraylib'
```

Depending on the install location of xraylib, it may be necessary to adjust the RUBYLIB environment variable.

3.12 PHP bindings (Linux and Mac OS X)

To include the xraylib PHP module in your own scripts add the following line:

```
include("xraylib.php");
```

Furthermore, you may have to modify your php.ini file by updating it with a new extension:

```
extension=<path_to_xraylib.so>/xraylib.so
```

and by modifying the include_path:

```
include_path=<path_to_xraylib.php>
```

The PHP bindings have been used to generate the online calculator.

3.13 Matlab bindings (Linux and Windows)

It was reported that xraylib can be called on Linux and Windows by defining functions as in the following examples.

3.13.1 Linux:

```
function A = atomicweight(Z)
    if libisloaded('libxrl') == 0
    loadlibrary('/usr/local/lib/libxrl.so',...
                '/usr/local/include/xraylib/xraylib.h')
    end
    A = calllib('libxrl','AtomicWeight',Z);
end
```

3.13.2 Windows

```
function A = atomicweight(Z)
    if libisloaded('libxrl') == 0
    loadlibrary('C:\Windows\System32\libxrl-7.dll',...
                'C:\Program Files\xraylib 64-bit\Include/xraylib.h',...
                'alias','libxrl')
    end
    A = calllib('libxrl','AtomicWeight',Z);
end
```

On both platforms one can query the available functions using:

```
libfunctions('libxrl', '-full')
```

xraylib can be unloaded using:

```
unloadlibrary('libxrl')
```

Obviously, the exact path to the xraylib library and the header will depend on the installation.
Since the developers do not have access to a Matlab installation, these bindings are currently not officially supported.

3.14 Mathematica bindings

xraylib has been reported to be callable from Mathematica (on Windows) after executing the following series of commands:

```
Needs["NETLink`"];
UninstallNET[];
InstallNET["Force32Bit"->True];
dllLocation="C:\\SomeFolder\\PathToDLL\\libxrl-7.dll";
```

To load an xraylib function one needs to define each function separately, according to the prototypes defined in the xraylib API

```
CSPhoto = DefineDLLFunction["CS_Photo", dllLocation, "double", {"int", "double"}];
CSFluorLineKissel = DefineDLLFunction["CS_FluorLine_Kissel", dllLocation,"double", 
FluorYield = DefineDLLFunction["FluorYield", dllLocation,"double", {"int", "int"}];
```

These functions may then be called as:

```
CSPhoto[33, 15.00];
CSFluorLineKissel[33, 0, 15.00];
FluorYield[33, 0];
```

The Mathematica bindings are currently not officially supported.

3.15 Command-line interface

The command-line utility called `xraylib` will be installed in the bin subfolder of the requested installation path (default: /usr/local/bin). If your shell does not pick this up after the installation, update your PATH environment variable accordingly. An example of usage (fluorescence line energy of Ca-Kalpha):

```
xraylib "LineEnergy(20,KA1_LINE)"
```

Help can be requested:

```
xraylib -h
```

3.16 Using xraylib: examples

Examples demonstrating the usage of the C library and the bindings can be found in the example folder of the xraylib package. These examples will be compiled and executed when `make check` is invoked after `make` during the installation.
Two complete typical usage examples written in Fortran and Ruby can be found here

4 The xraylib API: list of all functions

Important: with the release of version 3.0.0, we have changed all float datatypes to double! This may require users to make modifications to their existing C/C++/Obj-C and Fortran codes, as well as recompile their software that depends on xraylib. Apart from this datatype transition, we also changed the CompoundParser prototype.
Unless explicitly stated otherwise, all energies are assumed to be expressed in keV, whether used as input argument or return value.
Similarly, all angles are assumed to be expressed in radians.

4.1 Atomic weights

```
double AtomicWeight(int Z);
```

Given an element `Z`, returns its atomic weight in g/mol.

4.2 Element densities

```
double ElementDensity(int Z);
```

Given an element `Z`, returns its density at room temperature in g/cm^3.

4.3 Cross sections

```
double CS_Total(int Z, double E);
double CS_Photo(int Z, double E);
double CS_Rayl(int Z, double E);
double CS_Compt(int Z, double E);
```

Given an element Z and an energy E , these functions will return respectively the total absorption cross section, the photoionization cross section, the Rayleigh scattering cross section and the Compton scattering cross section, expressed in cm^2/g.

```
double CSb_Total(int Z, double E);
double CSb_Photo(int Z, double E);
double CSb_Rayl(int Z, double E);
double CSb_Compt(int Z, double E);
```

Identical to the last four functions, but the cross section is returned expressed in barn/atom.

```
double CS_KN(double E);
```

Given an energy E , returns the total Klein-Nishina cross section expressed in barn.

```
double CS_Photo_Partial(int Z, int shell, double E);
```

Given an element Z, shell-type macro `shell` and energy E, returns the partial photoionization cross section expressed in cm^2/g.

```
double CSb_Photo_Partial(int Z, int shell, double E);
```

Identical to the last function, but the cross section is returned in barn/electron.

```
double CS_Total_Kissel(int Z, double E);
double CSb_Total_Kissel(int Z, double E);
```

Identical to CS_Total and CSb_Total, but the photoionization contribution is calculated using the Kissel cross sections.

```
double CS_Energy(int Z, double E);
```

Given an element Z and an energy E, returns the mass-energy absorption cross section in $cm^2 g$

```
double CS_Total_CP(const char compound[], double E);
double CS_Photo_CP(const char compound[], double E);
double CS_Rayl_CP(const char compound[], double E);
double CS_Compt_CP(const char compound[], double E);
```

```
double CSb_Total_CP(const char compound[], double E);
double CSb_Photo_CP(const char compound[], double E);
double CSb_Rayl_CP(const char compound[], double E);
double CSb_Compt_CP(const char compound[], double E);
double CS_Total_Kissel_CP(const char compound[], double E);
double CSb_Total_Kissel_CP(const char compound[], double E);
double CS_Energy(const char compound[], double E);
```

Identical to the earlier mentioned functions, but require a chemical formula or a NIST compound name `compound` as first argument.

4.4 Unpolarized differential scattering cross sections

```
double DCS_Thoms(double theta);
```

Given a scattering polar angle `theta`, returns the Thomson differential cross section expressed in barn.

```
double DCS_KN(double E, double theta);
```

Given an energy `E` and a scattering polar angle `theta`, returns the Klein-Nishina differential scattering cross section expressed in barn.

```
double DCS_Rayl(int Z, double E, double theta);
double DCS_Compt(int Z, double E, double theta);
```

Given an element `Z`, energy `E` and a scattering polar angle `theta` , returns respectively the differential Rayleigh and the differential Compton scattering cross section expressed in cm^2/g/sterad.

```
double DCSb_Rayl(int Z, double E, double theta);
double DCSb_Compt(int Z, double E, double theta);
```

Identical to the last two functions, but the cross section is returned expressed in barn/atom/sterad.

```
double DCS_Rayl_CP(const char compound[], double E, double theta);
double DCS_Compt_CP(const char compound[], double E, double theta);
double DCSb_Rayl_CP(const char compound[], double E, double theta);
double DCSb_Compt_CP(const char compound[], double E, double theta);
```

Identical to the earlier mentioned functions, but require a chemical formula or a NIST compound name `compound` as first argument.

4.5 Polarized differential scattering cross sections

```
double DCSP_Thoms(double theta, double phi);
```

Given a scattering polar angle theta and scattering azimuthal angle phi, returns the Thomson differential cross section for a polarized beam expressed in barn.

```
double DCSP_KN(double E, double theta, double phi);
```

Given an energy E, a scattering polar angle theta and scattering azimuthal angle phi, returns the Klein-Nishina differential cross section for a polarized beam expressed in barn.

```
double DCSP_Rayl(int Z, double E, double theta, double phi);
double DCSP_Compt(int Z, double E, double theta, double phi);
```

Given an element Z, an energy E , a scattering polar angle theta and scattering azimuthal angle phi, returns respectively the Rayleigh differential and Compton differential cross sections for a polarized beam expressed in cm^2/g/sterad.

```
double DCSPb_Rayl(int Z, double E, double theta, double phi);
double DCSPb_Compt(int Z, double E, double theta, double phi);
```

Identical to the last two functions, but the cross section is returned expressed in barn/atom/sterad.

```
double DCSP_Rayl_CP(const char compound[], double E, double theta, double phi);
double DCSP_Compt_CP(const char compound[], double E, double theta, double phi);
double DCSPb_Rayl_CP(const char compound[], double E, double theta, double phi);
double DCSPb_Compt_CP(const char compound[], double E, double theta, double phi);
```

Identical to the earlier mentioned functions, but require a chemical formula or a NIST compound name compound as first argument.

4.6 Scattering factors

In this section, we introduce the momentum transfer parameter q, which is used in several of the following functions. It should be noted that several definitions can be found of this parameter throughout the scientific literature, which vary mostly depending on the community where it is used. The crystallography and diffraction community for example, use the following definition:

$$q = 4\pi \times \sin(\theta)/\lambda$$

with θ the angle between the incident X-ray and the crystal scattering planes according to Bragg's law, and λ the wavelength.
xraylib uses however, a different definition, in which θ corresponds to the scattering angle, which in case of Bragg scattering is equal to twice the angle from the previous definition. This new definition has the advantage of being useful when working with amorphous materials, as well as with incoherent scattering. Furthermore, our definition drops the 4π scale factor, in line with the definition by Hubbell et al in *Atomic form factors, incoherent scattering functions, and photon scattering cross sections*, J. Phys. Chem. Ref. Data, Vol.4, No. 3, 1975:

$$q = E \times \sin(\theta/2) \times h \times c \times 10^8$$

with E the energy of the photon, h Planck's constant and c the speed of light. The unit of the returned momentum transfer is then Å^{-1}.

```
double  FF_Rayl(int Z, double q);
```

Given an element `Z` and a momentum transfer `q` (expressed in Å^{-1}), returns the atomic form factor for Rayleigh scattering.

```
double  SF_Compt(int Z, double q);
```

Given an element `Z` and a momentum transfer `q` (expressed in Å^{-1}), returns the incoherent scattering function for Compton scattering.

```
double  MomentTransf(double E, double theta);
```

Given an energy `E` and a scattering polar angle `theta`, returns the momentum transfer for X-ray photon scattering expressed in Å^{-1}.

```
double Fi(int Z, double E);
double Fii(int Z, double E);
```

Given an element `Z` and and energy `E`, returns respectively the anomalous scattering factors $\Delta f'$ and $\Delta f''$.

4.7 X-ray fluorescence line energies

```
double LineEnergy(int Z, int line);
```

Given an element `Z` and line-type macro `line`, returns the energy of the requested XRF line expressed in keV.

4.8 X-ray fluorescence yields

```
double  FluorYield(int Z, int shell);
```

Given an element `Z` and shell-type macro `shell`, returns the corresponding fluorescence yield .

4.9 Auger yields

```
double  AugerYield(int Z, int shell);
```

Given an element Z and shell-type macro shell, returns the corresponding Auger yield .

4.10 Coster-Kronig transition probabilities

```
double  CosKronTransProb(int Z, int trans);
```

Given an element Z and transition-type macro trans, returns the corresponding Coster-Kronig transition probability.

4.11 Absorption edge energies

```
double EdgeEnergy(int Z, int shell);
```

Given an element Z and shell-type macro shell, returns the absorption edge energy expressed in keV.

4.12 Jump factors

```
double JumpFactor(int Z, int shell);
```

Given an element Z and shell-type macro shell, returns the jump factor.

4.13 X-ray fluorescence cross sections

```
double CS_FluorLine(int Z, int line, double E);
```

Given an element Z, a line-type macro line and an energy E, returns the XRF cross section expressed in cm^2/g

```
double CSb_FluorLine(int Z, int line, double E);
```

Identical to the previous function, but returns the cross section expressed in barn/atom. These last two functions calculate the XRF cross sections assuming the jump factor approximation. We also offer XRF cross sections calculated using the partial photoelectric effect cross sections calculated by Kissel et al. The corresponding functions are:

```
double CS_FluorLine_Kissel(int Z, int line, double E);
double CSb_FluorLine_Kissel(int Z, int line, double E);
```

Recently we introduced XRF cross sections that take into account cascade effects, both those coming from radiative transitions and those from non-radiative transitions :

```
double CS_FluorLine_Kissel_Cascade(int Z, int line, double E);
double CSb_FluorLine_Kissel_Cascade(int Z, int line, double E);
double CS_FluorLine_Kissel_Nonradiative_Cascade(int Z, int line, double E);
double CSb_FluorLine_Kissel_Nonradiative_Cascade(int Z, int line, double E);
double CS_FluorLine_Kissel_Radiative_Cascade(int Z, int line, double E);
double CSb_FluorLine_Kissel_Radiative_Cascade(int Z, int line, double E);
double CS_FluorLine_Kissel_no_Cascade(int Z, int line, double E);
double CSb_FluorLine_Kissel_no_Cascade(int Z, int line, double E);
```

`CS_FluorLine_Kissel` and `CbS_FluorLine_Kissel` are mapped to resp. `CS_FluorLine_Kissel_Cascad` and `CSb_FluorLine_Kissel_Cascade`.
Using these functions, it is possible to examine the influence of the two different cascade types separately, but keep in mind that in reality they will always be occurring simultaneously.
All `CS_FluorLine*` functions offer XRF cross sections for both K- and L-lines (provided the corresponding shells can be excited), but only the `CS_FluorLine_Kissel*` functions offer also the M-line XRF cross sections!

4.14 Radiative rates

```
double RadRate(int Z, int line);
```

Given an element `Z` and a line-type macro `line`, returns the radiative rate.

4.15 Non-radiative rates

```
double AugerRate(int Z, int auger);
```

Given an element `Z` and an Auger-type macro `auger` corresponding with the electrons involved, returns the non-radiative rate.

4.16 Atomic level widths

```
double AtomicLevelWidth(int Z, int shell);
```

Given an element `Z` and a shell-type macro `shell`, returns the atomic level width in keV.

4.17 Compton energy

```
double ComptonEnergy(double E0, double theta);
```

Given an initial photon energy `E0` and a scattering polar angle `theta`, returns the photon energy after Compton scattering.

4.18 Refractive indices

```
double Refractive_Index_Re(const char compound[], double E, double rho);
double Refractive_Index_Im(const char compound[], double E, double rho);
xrlComplex Refractive_Index(const char compound[], double E, double rho);
```

Given a chemical formula `compound`, energy `E` and a density `rho`, return respectively the real, the imaginary or both parts of the refractive index. For a definition of `xrlComplex`, see the crystal diffraction section.

4.19 Compton profiles

```
double ComptonProfile(int Z, double pz);
```

Given an element `Z` and a momentum `pz` (expressed in atomic units), returns the Compton scattering profile summed over all shells.

```
double ComptonProfile_Partial(int Z, int shell, double pz);
```

Given an element `Z`, a shell-type macro `shell` and a momentum `pz`, returns the Compton scattering profile for a particular subshell.

4.20 Electronic configurations

```
double ElectronConfig(int Z, int shell);
```

Given an element `Z` and a shell-type macro `shell` , returns the number of electrons the shell possesses.

4.21 Crystal diffraction

```
Crystal_Struct* Crystal_GetCrystal(const char* material, Crystal_Array* c_array);
```

Get a pointer to a Crystal_Struct of a given crystal `material` from `c_array`.
If `c_array` is `NULL` then the internal array of crystals is searched.
If not found, `NULL` is returned.
The `c_array` argument is only used in C, C++ and Fortran. The other bindings support only the internal array.

```
double Bragg_angle (Crystal_Struct* cryst, double E, int i_miller, int j_miller, int
```

Computes the Bragg angle in radians for a given crystal `cryst`, energy `E` and Miller indices `i_miller`, `j_miller` and `k_miller`.

```
double Q_scattering_amplitude(Crystal_Struct* cryst, double E, int i_miller, int j_m
```

Computes the Q scattering amplitude for a given crystal `cryst`, incident energy `E`, Miller indices (`i_miller`, `j_miller` and `k_miller`) and relative angle `rel_angle`.

```
void Atomic_Factors (int Z, double E, double q, double debye_factor, double* f0, dou
```

Computes the atomic factors f_0 `f0`, Δf′ `f_prime` and Δf″ `f_prime2` for a given element `Z`, incident energy `E`, momentum transfer `q` and Debye factor `debye_factor`. `f0`, `f_prime` and `f_prime2` are pointers to doubles in C, C++, Fortran and IDL BUT return values in Perl, Python and Lua!!

```
xrlComplex Crystal_F_H_StructureFactor (Crystal_Struct* cryst, double E, int i_mille
```

Computes the F_H Structure factor for a given crystal `cryst`, incident energy `E`, Miller indices (`i_miller`, `j_miller` and `k_miller`), Debye factor `debye_factor` and relative angle `rel_angle`. The return value is a complex number.

```
xrlComplex Crystal_F_H_StructureFactor_Partial (Crystal_Struct* crystal, double ener
```

with:

```
typedef struct {
    double re;
    double im;
} xrlComplex;
```

with:

- `re`: real part
- `im`: imaginary part

Wherever possible for the bindings (Python, IDL, Perl, Ruby, Fortran, C#), we have tried using the native complex number datatype in favor of a direct analogue of the xrlComplex struct.
See also `Crystal_F_H_StructureFactor`.
The Atomic structure factor has three terms:

$$F_H = f_0 + \Delta f' + \Delta f''.$$

For each of these three terms, there is a corresponding `*_flag` argument which controls the numerical value used in computing F_H:
`*_flag` = 0 → Set this term to 0.
`*_flag` = 1 → Set this term to 1. Only used for f_0.
`*_flag` = 2 → Set this term to the value given.

```
double Crystal_UnitCellVolume (Crystal_Struct* cryst);
```

Computes the unit cell volume for a crystal `cryst`. Structures obtained from the official array will have their volume in .volume

```
double Crystal_dSpacing (Crystal_Struct* cryst, int i_miller, int j_miller, int k_mi
```

Computes the d-spacing for a given crystal `cryst` and Miller indices (`i_miller`, `j_miller` and `k_miller`).
The routine assumes that if `cryst->volume` is nonzero then it holds a valid value.
If (i,j,k) = (0,0,0) then zero is returned.

4.22 Compound parser

```
struct compoundData {
    int nElements;
    double nAtomsAll;
    int *Elements;
    double *massFractions;
};
```

with:

- `nElements`: number of different elements in the compound
- `nAtomsAll`: number of atoms in the formula. Since indices may be real numbers, this member variable is of type double
- `Elements`: a dynamically allocated array (length = `nElements`) containing the elements, in ascending order
- `massFractions`: a dynamically allocated array (length = `nElements`) containing the mass fractions of the elements in Elements

```
struct compoundData *CompoundParser(const char compoundString[])
```

The CompoundParser function will parse a chemical formula `compoundString` and will allocate a compoundData structure with the results if successful, otherwise `NULL` is returned.
Chemical formulas may contain (nested) brackets, followed by an integer or real number (with a dot) subscript.
Examples of accepted formulas are: `H2O`, `Ca5(PO4)3F`, `Ca5(PO4)F0.33Cl0.33(OH)0.33`.
The allocated memory should be freed with

```
FreeCompoundData(struct compoundData *cd); (C/C++/Obj-C and Fortran only)
```

```
char * AtomicNumberToSymbol(int Z);
```

The AtomicNumberToSymbol function returns a pointer to a string containing the element for atomic number `Z`. If an error occurred, the `NULL` string is returned. The string should be freed after usage with the xrlFree function (C/C++/Obj-C and Fortran only).

```
int SymbolToAtomicNumber(char *symbol);
```

The SymbolToAtomicNumber function returns the atomic number that corresponds with element `symbol`. If the element does not exist, 0 is returned.

4.23 NIST compound catalog

```
struct compoundDataNIST {
    char *name;
    int nElements;
    int *Elements;
    double *massFractions;
    double density;
};
```

with:

- `name`: a string containing the full name of the compound, as retrieved from the NIST database
- `nElements`: number of different elements in the compound
- `Elements`: a dynamically allocated array (length = `nElements`) containing the elements, in ascending order
- `massFractions`: a dynamically allocated array (length = `nElements`) containing the mass fractions of the elements in Elements
- `density`: the density of the compound, expressed in g/cm^3

```
struct compoundDataNIST *GetCompoundDataNISTByName(const char compoundString[]);
```

```
struct compoundDataNIST *GetCompoundDataNISTByIndex(int compoundIndex);
```

Using these two functions it is possible to query the contents of NISTs catalog of compound compositions and densities. The former takes a compound name `compoundString` and if a match is found, the corresponding newly allocated compoundDataNIST structure is returned, while the latter takes an index `compoundIndex` in the form of a NIST compound-type macro. The list of compound names can be queried using:

```
char **GetCompoundDataNISTList(int *nCompounds);
```

which returns a `NULL` terminated array of strings. Optionally, pass a pointer to an integer `nCompounds` to obtain the number of strings in the array (pass `NULL` if value is not required). This option is only present in the C/C++/Obj-C implementation.
The list can also be obtained at our online xraylib calculator.
After usage, the returned compoundDataNIST structures should be freed with (C/C++/Obj-C and Fortran only):

```
void FreeCompoundDataNIST(struct compoundDataNIST *compoundData);
```

4.24 Radionuclides

```
struct radioNuclideData {
    char *name;
    int Z;
    int A;
    int N;
    int Z_xray;
    int nXrays;
    int *XrayLines;
    double *XrayIntensities;
    int nGammas;
    double *GammaEnergies;
    double *GammaIntensities;
};
```

with:

- `name`: a string containing the mass number (A), followed by the chemical element (e.g. 55Fe)
- `Z`: atomic number of the radionuclide
- `A`: mass number of the radionuclide
- `N`: number of neutrons of the radionuclide
- `Z_xray`: atomic number of the nuclide after decay, which should be used in calculating the energy of the emitted X-ray lines using `LineEnergy`
- `nXrays`: number of emitted characteristic X-rays
- `XrayLines`: a dynamically allocated array (length = `nXrays`) of line-type macros, identifying the emitted X-rays
- `XrayIntensities`: a dynamically allocated array (length = `nXrays`) of photons per disintegration, one value per emitted X-ray
- `nGammas`: number of emitted gamma-rays
- `GammaEnergies`: a dynamically allocated array (length = `nGammas`) of emitted gamma-ray energies
- `GammaIntensities`: a dynamically allocated array (length = `nGammas`) of emitted gamma-ray photons per disintegration

```
struct radioNuclideData *GetRadioNuclideDataByName(const char radioNuclideString[]);

struct radioNuclideData *GetRadioNuclideDataByIndex(int radioNuclideIndex);
```

Use these two functions to query xraylib's database of X-ray emission profiles for several important radionuclides. The former expects the name `radioNuclideString` of a radionuclide, while the latter takes a radionuclide-type macro `radioNuclideIndex`. When successful, a freshly allocated radioNuclideData structure is returned. Query the list of names using:

```
char **GetRadioNuclideDataList(int *nRadioNuclides);
```

which returns a `NULL` terminated array of strings. Optionally, pass a pointer to an integer `nRadioNuclides` to obtain the number of strings in the array (pass `NULL` if value is not required). This option is only present in the C/C++/Obj-C implementation.
The list can also be obtained at our online xraylib calculator.
After usage, the returned radioNuclideData structures should be freed with (C/C++/Obj-C and Fortran only):

```
void FreeRadioNuclideData(struct radioNuclideData *radioNuclideData);
```

4.25 Error handling

xraylib's error handling consists of error messages that are presented to the user whenever necessary, most commonly when invalid arguments were detected. Internally a global variable `ExitStatus` will be set to 1 at the first occurrence of an error. It is possible to force the calling program to exit in this case, by using the `SetHardExit` function, before calling the function that could trigger the error. If you are merely interested in knowing whether an error occurred or not, check the return value of `GetExitStatus`, after calling the function that could trigger the error. Consider setting `SetExitStatus(0)` before to ensure that no previous errors would influence the outcome of `GetExitStatus`. The following functions determine the error handling behavior:

```
void SetHardExit(int hard_exit);
```

Pass 1 to exit the program at the first error.

```
void SetExitStatus(int exit_status);
```

```
int GetExitStatus(void);
```

These two functions allow for the setting and getting of the `ExitStatus`. If it is equal to 1, then an error has occurred, while 0 indicates normal operation.
The authors of xraylib strongly discourage the use of these three functions. Due to the internal use of a global variable, they are NOT thread-safe. Furthermore, some functions will set the `ExitStatus` variable to 1, although no error has occurred! This is considered a bug and will be fixed in a future update. For now, the authors recommend to check the return value of the functions: 0 indicates either an error or an unavailable quantity.
Since some of the error messages are in fact meaningless (in particular those produced by the XRF cross section functions), it may be useful to turn them off completely. Use the following functions for this purpose:

```c
void SetErrorMessages(int status);

int GetErrorMessages(void);
```

Passing 0 to `SetErrorMessages` suppresses the output of the error messages, while 1 restores it.

5 Code examples

In this section we will demonstrate how to use xraylib in some real-world situations. You should be able to compile (if necessary) and run these examples after installing xraylib with the required bindings.

5.1 Fundamental parameter method (Fortran 2003)

This first example, written in Fortran 2003, demonstrates how one can determine the expected first order net-line intensity of a particular XRF line after irradiating a sample (apatite) with an X-ray beam. Compilation instructions can be found in the Fortran bindings section.
The equation that has been used here can be found in every handbook on quantitative X-ray fluorescence and many scientific articles. Try for example *Spectrochim. Acta Part B, 67:32–42, 2012.*

```fortran
PROGRAM fpm

USE :: xraylib
USE, INTRINSIC :: ISO_C_BINDING

IMPLICIT NONE

REAL (C_DOUBLE) :: flux = 1E9 !photons/s
REAL (C_DOUBLE) :: G = 1E-5
REAL (C_DOUBLE) :: density = 3.19 !g/cm3
REAL (C_DOUBLE) :: thickness = 0.1 !cm
REAL (C_DOUBLE) :: xrf_intensity, chi
REAL (C_DOUBLE) :: mu_0, mu_1, w_Ca, A_corr, Q
REAL (C_DOUBLE) :: alpha = 45.0, beta = 45.0 !degrees
REAL (C_DOUBLE) :: beam_energy = 20.0 !keV
TYPE (compoundData), POINTER :: cd
CHARACTER (len=50) :: apatite = 'Ca5(PO4)3(OH)0.33F0.33Cl0.33'
REAL (C_DOUBLE), PARAMETER :: deg2rad = 3.14159265359/180.0

cd => compoundParser(apatite)
w_Ca = cd%massFractions(6) ! fortran array indexing starts at 1!!!!
```

```
mu_0 = CS_Total_CP(apatite, beam_energy)
mu_1 = CS_Total_CP(apatite, LineEnergy(20, KL3_LINE))
chi = mu_0/SIN(deg2rad*alpha) + mu_1/SIN(deg2rad*beta)
A_corr = (1.0-EXP(-chi*density*thickness))/(chi*density*thickness)
Q = CS_FluorLine_Kissel(20, KL3_LINE, beam_energy)

xrf_intensity = flux*G*Q*w_Ca*density*thickness*A_corr

CALL FreeCompoundData(cd)

WRITE (*, '(A, ES12.4)') 'xrf_intensity: ', xrf_intensity
END PROGRAM fpm
```

Save as `fpm.f90` and compile with:

```
gfortran -o fpm `pkg-config --cflags libxrlf03` fpm.f90 `pkg-config --libs libxrlf03
```

Executing `fpm` should produce the following output:

```
xrf_intensity:   1.0849E+01
```

5.2 Transmission efficiency using Monte Carlo method (Ruby)

The Monte Carlo method is often used in the field of X-rays as it can be used to predict the outcome of experiments, provided that all the relevant physical datasets are present. xraylib can be used to this effect as is shown in the following example, written in Ruby, in which one determines the fraction of photons than manages to penetrate through a sample of a particular thickness along the beampath.

```
require 'xraylib'

compound = "Uranium Monocarbide"

Xraylib.SetErrorMessages(0)

cdn = Xraylib.GetCompoundDataNISTByName(compound)
density = cdn['density'] #g/cm3
thickness = 0.01 #cm
energy = 50.0 #keV

mu_rho = Xraylib.CS_Total_CP(compound, energy)*density

transmitted = 0
total = 100000
```

```
total.times {|i|
x = -Math.log(rand())/mu_rho
transmitted += 1 if x > thickness
}

printf("transmitted: %i\n", transmitted)
printf("MC fraction: %f\n", Float(transmitted)/total)
printf("True fraction: %f\n", Math.exp(-mu_rho*thickness))
```

Save as `mc.rb` and execute with:

```
ruby mc.rb
```

This should produce something similar (remember: it's a Monte Carlo simulation!) to the following output:

```
transmitted: 23438
MC fraction: 0.234380
True fraction: 0.233201
```

5.3 More examples

The xraylib example folder contains example files for all the languages that are officially supported. Click on the following to links to access them directly.

- C
- Perl
- Fortran 2003
- IDL
- Python
- C++
- Java
- C#
- Lua
- Ruby
- PHP
- Python-Numpy

6 References and additional resources

Our work on *xraylib* has been published in two peer-reviewed journals and has been presented at several conferences. Additional information can be found in the following resources.

6.1 Papers

- A library for X-ray matter interaction cross sections for X-ray fluorescence applications. Antonio Brunetti, Manuel Sanchez del Rio, Bruno Golosio, Alexandre Simionovici and Andrea Somogyi. Spectrochimica Acta B, 59(10-11), 1725-1731, 2004. DOI
- The xraylib library for X-ray–matter interactions. Recent developments. T. Schoonjans, A. Brunetti, B. Golosio, M. Sanchez del Rio, V. A. Solé, C. Ferrero and L. Vincze. Spectrochimica Acta Part B, 66(11-12), 776-784, 2011. DOI

6.2 Posters

- The xraylib library for X-ray matter interaction cross-sections: New developments and applications. Tom Schoonjans, Manuel Sanchez del Rio, Laszlo Vincze. Denver X-ray Conference, Colorado Springs, CO, USA, 27-31 July 2009 – Best XRF poster award PDF
- The xraylib library for X-ray matter interaction cross sections: New developments and applications. Tom Schoonjans, Antonio Brunetti, Bruno Golosio, Manuel Sanchez del Rio, Vicente Armando Solé, Claudio Ferrero and Laszlo Vincze. SPIE Optics and Photonics, San Diego, CA, USA, 21-25 August 2011
- The xraylib library for X- ray—matter interactions: recent developments. Tom Schoonjans, Antonio Brunetti, Bruno Golosio, Manuel Sanchez Del Rio, Vicente Armando Solé, Claudio Ferrero and Laszlo Vincze. European Conference on X-Ray Spectrometry, IAEA, Vienna, Austria, 18-22 June 2012
- The xraylib library for X-ray-matter interactions. Tom Schoonjans, Antonio Brunetti, Bruno Golosio, Manuel Sanchez del Rio, Vicente Armando Solé, Claudio Ferrero and Laszlo Vincze. European Conference on X-Ray Spectrometry, Alma Mater Studiorum Università di Bologna, Italy, 15-20 June 2014
- The xraylib library for X-ray-matter interactions. Tom Schoonjans, Antonio Brunetti, Bruno Golosio, Manuel Sanchez del Rio, Vicente Armando Solé, Claudio Ferrero and Laszlo Vincze. European Conference on X-Ray Spectrometry, University of Gothenburg, Sweden, 19-24 June 2016 PDF

6.3 Oral presentation

- The xraylib library for X-ray matter interaction cross sections: New developments. Tom Schoonjans, Manuel Sanchez del Rio, Antonio Brunetti, Bruno Golosio, Alexandre Simionovici, Claudio Ferrero and Laszlo Vincze. European Conference on X-Ray Spectrometry, Figueira da Foz, Coimbra, Portugal, 20-25 June 2010 PDF
- The xraylib library for X-ray-matter interaction cross sections. Tom Schoonjans. Monte Carlo simulation tools for X-ray imaging and fluorescence work-

shop, ESRF, Grenoble, France, 24-25 February 2014. PDF

A Tables of xraylib macros

This appendix contains tables of all macros that are used by the xraylib API. These will be substituted at compile-time (C, C++, Objective-C) or run-time with a corresponding integer. These integers have intentionally not been included in these tables as their direct usage is strongly discouraged: it is possible that these integers will change in future versions!

A.1 Physical constants

xraylib provides several physical constants, which are commonly used in the field of X-ray physics. They are implemented as macros that are mapped to the corresponding values as provided by NIST.

Constant	Value	Unit	Macro
Avogadro constant	0.602214129	$mol^{-1} \times barn^{-1} \times cm^2$	`AVOGNUM`
Electron rest mass energy	510.998928	keV	`MEC2`
Square of classical electron radius	0.079407877	barn	`RE2`
Classical electron radius	2.8179403267e-15	m	`R_E`
keV to Å^{-1} conversion	12.39841930	-	`KEV2ANGST`

A.2 X-ray fluorescence line macros

In this section an overview is presented of all macros that are being used by such functions as `LineEnergy`, `RadRate` and the `CS_FluorLine` family. Three types of macros are offered: IUPAC, Siegbahn and obsolete. The first two categories refer to individual XRF lines while the third category refers to groups of XRF lines. Usage of this last group is deprecated as explained in our manuscript The xraylib library for X-ray—matter interactions. Recent developments:

Access to these functions from the xraylib API is managed through a number of macros which allow for the extraction of specific information, such as a particular XRF line or atomic shell. A number of examples can be found in our earlier work [13]. However, care should be taken when using the macros related to the XRF lines. Despite the long-standing recommendation by the International Union of Pure and Applied Chemistry (IUPAC) to designate an XRF line based on the atomic levels involved in the transition (e.g. K−L_3), the Siegbahn notation is still widely used in X-ray spectroscopy. This notation (e.g. $K_{\alpha 1}$), which classifies lines on the basis of their relative intensities, is very often used in a way that overlapping lines are referred to using a single notation: e.g. K_α corresponds to both $K_{\alpha 1}$ and $K_{\alpha 2}$. Another example concerns the L_β line which actually indicates about twenty individual lines. This situation has an impact on how several functions from the xraylib API are used: Firstly, when retrieving an XRF cross section for a line group, the function must return the sum of the XRF cross sections of all lines belonging to that group. Secondly, a problem arises when calling the XRF line energy of a line group. One solution is returning an (weighted) average of the individual line energies. In the case of elements with a high atomic number, however, this would result in a line energy that would be absent from a real experimental spectrum (e.g. Pb K-L_2: 72.8045 keV, K-L_3: 74.9693). Thirdly, in the case of transition probabilities, it makes no sense to request e.g. this value for L_β since it corresponds to several lines, originating from the L_1, L_2 and L_3 shells! These issues were circumvented at an earlier stage by defining 4 Siegbahn macros `KA_LINE`, `KB_LINE`, `LA_LINE` and `LB_LINE` which correspond to $K_{\alpha 1}$+$K_{\alpha 2}$, $K_{\beta 1}$ + $K_{\beta 2}$, $L_{\alpha 1}$ and $L_{\beta 1}$, respectively. These 4 macros were accompanied by a large number of IUPAC macros, each associated with an individual line. Recent xraylib versions have, however, been augmented with about 45 new Siegbahn macros retrieving individual XRF lines [33]. Their usage will resolve the above mentioned issues entirely. The original 4 Siegbahn macros are kept for backward compatibility but their usage is deprecated. When invoking functions with these macros, the returned values will be valid for the complete line group, which means that they will in some cases be different from older xraylib versions.

A.2.1 IUPAC X-ray fluorescence line macros

Transition	IUPAC macro
$K \leftarrow L_1$	`KL1_LINE`
$K \leftarrow L_2$	`KL2_LINE`
$K \leftarrow L_3$	`KL3_LINE`

Transition	IUPAC macro
$K \leftarrow M_1$	KM1_LINE
$K \leftarrow M_2$	KM2_LINE
$K \leftarrow M_3$	KM3_LINE
$K \leftarrow M_4$	KM4_LINE
$K \leftarrow M_5$	KM5_LINE
$K \leftarrow N_1$	KN1_LINE
$K \leftarrow N_2$	KN2_LINE
$K \leftarrow N_3$	KN3_LINE
$K \leftarrow N_4$	KN4_LINE
$K \leftarrow N_5$	KN5_LINE
$K \leftarrow N_6$	KN6_LINE
$K \leftarrow N_7$	KN7_LINE
$K \leftarrow O$	KO_LINE
$K \leftarrow O_1$	KO1_LINE
$K \leftarrow O_2$	KO2_LINE
$K \leftarrow O_3$	KO3_LINE
$K \leftarrow O_4$	KO4_LINE
$K \leftarrow O_5$	KO5_LINE
$K \leftarrow O_6$	KO6_LINE
$K \leftarrow O_7$	KO7_LINE
$K \leftarrow P$	KP_LINE
$K \leftarrow P_1$	KP1_LINE
$K \leftarrow P_2$	KP2_LINE
$K \leftarrow P_3$	KP3_LINE
$K \leftarrow P_4$	KP4_LINE
$K \leftarrow P_5$	KP5_LINE
$L_1 \leftarrow L_2$	L1L2_LINE
$L_1 \leftarrow L_3$	L1L3_LINE
$L_1 \leftarrow M_1$	L1M1_LINE
$L_1 \leftarrow M_2$	L1M2_LINE
$L_1 \leftarrow M_3$	L1M3_LINE
$L_1 \leftarrow M_4$	L1M4_LINE
$L_1 \leftarrow M_5$	L1M5_LINE
$L_1 \leftarrow N_1$	L1N1_LINE
$L_1 \leftarrow N_2$	L1N2_LINE
$L_1 \leftarrow N_3$	L1N3_LINE
$L_1 \leftarrow N_4$	L1N4_LINE
$L_1 \leftarrow N_5$	L1N5_LINE
$L_1 \leftarrow N_6$	L1N6_LINE
$L_1 \leftarrow N_{67}$	L1N67_LINE
$L_1 \leftarrow N_7$	L1N7_LINE
$L_1 \leftarrow O_1$	L1O1_LINE

Transition	IUPAC macro
$L_1 \leftarrow O_2$	L1O2_LINE
$L_1 \leftarrow O_3$	L1O3_LINE
$L_1 \leftarrow O_4$	L1O4_LINE
$L_1 \leftarrow O_{45}$	L1O45_LINE
$L_1 \leftarrow O_5$	L1O5_LINE
$L_1 \leftarrow O_6$	L1O6_LINE
$L_1 \leftarrow O_7$	L1O7_LINE
$L_1 \leftarrow P_1$	L1P1_LINE
$L_1 \leftarrow P_2$	L1P2_LINE
$L_1 \leftarrow P_{23}$	L1P23_LINE
$L_1 \leftarrow P_3$	L1P3_LINE
$L_1 \leftarrow P_4$	L1P4_LINE
$L_1 \leftarrow P_5$	L1P5_LINE
$L_2 \leftarrow L_3$	L2L3_LINE
$L_2 \leftarrow M_1$	L2M1_LINE
$L_2 \leftarrow M_2$	L2M2_LINE
$L_2 \leftarrow M_3$	L2M3_LINE
$L_2 \leftarrow M_4$	L2M4_LINE
$L_2 \leftarrow M_5$	L2M5_LINE
$L_2 \leftarrow N_1$	L2N1_LINE
$L_2 \leftarrow N_2$	L2N2_LINE
$L_2 \leftarrow N_3$	L2N3_LINE
$L_2 \leftarrow N_4$	L2N4_LINE
$L_2 \leftarrow N_5$	L2N5_LINE
$L_2 \leftarrow N_6$	L2N6_LINE
$L_2 \leftarrow N_7$	L2N7_LINE
$L_2 \leftarrow O_1$	L2O1_LINE
$L_2 \leftarrow O_2$	L2O2_LINE
$L_2 \leftarrow O_3$	L2O3_LINE
$L_2 \leftarrow O_4$	L2O4_LINE
$L_2 \leftarrow O_5$	L2O5_LINE
$L_2 \leftarrow O_6$	L2O6_LINE
$L_2 \leftarrow O_7$	L2O7_LINE
$L_2 \leftarrow P_1$	L2P1_LINE
$L_2 \leftarrow P_2$	L2P2_LINE
$L_2 \leftarrow P_{23}$	L2P23_LINE
$L_2 \leftarrow P_3$	L2P3_LINE
$L_2 \leftarrow P_4$	L2P4_LINE
$L_2 \leftarrow P_5$	L2P5_LINE
$L_2 \leftarrow Q_1$	L2Q1_LINE
$L_3 \leftarrow M_1$	L3M1_LINE
$L_3 \leftarrow M_2$	L3M2_LINE

Transition	IUPAC macro
$L_3 \leftarrow M_3$	L3M3_LINE
$L_3 \leftarrow M_4$	L3M4_LINE
$L_3 \leftarrow M_5$	L3M5_LINE
$L_3 \leftarrow N_1$	L3N1_LINE
$L_3 \leftarrow N_2$	L3N2_LINE
$L_3 \leftarrow N_3$	L3N3_LINE
$L_3 \leftarrow N_4$	L3N4_LINE
$L_3 \leftarrow N_5$	L3N5_LINE
$L_3 \leftarrow N_6$	L3N6_LINE
$L_3 \leftarrow N_7$	L3N7_LINE
$L_3 \leftarrow O_1$	L3O1_LINE
$L_3 \leftarrow O_2$	L3O2_LINE
$L_3 \leftarrow O_3$	L3O3_LINE
$L_3 \leftarrow O_4$	L3O4_LINE
$L_3 \leftarrow O_{45}$	L3O45_LINE
$L_3 \leftarrow O_5$	L3O5_LINE
$L_3 \leftarrow O_6$	L3O6_LINE
$L_3 \leftarrow O_7$	L3O7_LINE
$L_3 \leftarrow P_1$	L3P1_LINE
$L_3 \leftarrow P_2$	L3P2_LINE
$L_3 \leftarrow P_{23}$	L3P23_LINE
$L_3 \leftarrow P_3$	L3P3_LINE
$L_3 \leftarrow P_4$	L3P4_LINE
$L_3 \leftarrow P_{45}$	L3P45_LINE
$L_3 \leftarrow P_5$	L3P5_LINE
$L_3 \leftarrow Q_1$	L3Q1_LINE
$M_1 \leftarrow M_2$	M1M2_LINE
$M_1 \leftarrow M_3$	M1M3_LINE
$M_1 \leftarrow M_4$	M1M4_LINE
$M_1 \leftarrow M_5$	M1M5_LINE
$M_1 \leftarrow N_1$	M1N1_LINE
$M_1 \leftarrow N_2$	M1N2_LINE
$M_1 \leftarrow N_3$	M1N3_LINE
$M_1 \leftarrow N_4$	M1N4_LINE
$M_1 \leftarrow N_5$	M1N5_LINE
$M_1 \leftarrow N_6$	M1N6_LINE
$M_1 \leftarrow N_7$	M1N7_LINE
$M_1 \leftarrow O_1$	M1O1_LINE
$M_1 \leftarrow O_2$	M1O2_LINE
$M_1 \leftarrow O_3$	M1O3_LINE
$M_1 \leftarrow O_4$	M1O4_LINE
$M_1 \leftarrow O_5$	M1O5_LINE

Transition	IUPAC macro
$M_1 \leftarrow O_6$	M1O6_LINE
$M_1 \leftarrow O_7$	M1O7_LINE
$M_1 \leftarrow P_1$	M1P1_LINE
$M_1 \leftarrow P_2$	M1P2_LINE
$M_1 \leftarrow P_3$	M1P3_LINE
$M_1 \leftarrow P_4$	M1P4_LINE
$M_1 \leftarrow P_5$	M1P5_LINE
$M_2 \leftarrow M_3$	M2M3_LINE
$M_2 \leftarrow M_4$	M2M4_LINE
$M_2 \leftarrow M_5$	M2M5_LINE
$M_2 \leftarrow N_1$	M2N1_LINE
$M_2 \leftarrow N_2$	M2N2_LINE
$M_2 \leftarrow N_3$	M2N3_LINE
$M_2 \leftarrow N_4$	M2N4_LINE
$M_2 \leftarrow N_5$	M2N5_LINE
$M_2 \leftarrow N_6$	M2N6_LINE
$M_2 \leftarrow N_7$	M2N7_LINE
$M_2 \leftarrow O_1$	M2O1_LINE
$M_2 \leftarrow O_2$	M2O2_LINE
$M_2 \leftarrow O_3$	M2O3_LINE
$M_2 \leftarrow O_4$	M2O4_LINE
$M_2 \leftarrow O_5$	M2O5_LINE
$M_2 \leftarrow O_6$	M2O6_LINE
$M_2 \leftarrow O_7$	M2O7_LINE
$M_2 \leftarrow P_1$	M2P1_LINE
$M_2 \leftarrow P_2$	M2P2_LINE
$M_2 \leftarrow P_3$	M2P3_LINE
$M_2 \leftarrow P_4$	M2P4_LINE
$M_2 \leftarrow P_5$	M2P5_LINE
$M_3 \leftarrow M_4$	M3M4_LINE
$M_3 \leftarrow M_5$	M3M5_LINE
$M_3 \leftarrow N_1$	M3N1_LINE
$M_3 \leftarrow N_2$	M3N2_LINE
$M_3 \leftarrow N_3$	M3N3_LINE
$M_3 \leftarrow N_4$	M3N4_LINE
$M_3 \leftarrow N_5$	M3N5_LINE
$M_3 \leftarrow N_6$	M3N6_LINE
$M_3 \leftarrow N_7$	M3N7_LINE
$M_3 \leftarrow O_1$	M3O1_LINE
$M_3 \leftarrow O_2$	M3O2_LINE
$M_3 \leftarrow O_3$	M3O3_LINE
$M_3 \leftarrow O_4$	M3O4_LINE

Transition	IUPAC macro
$M_3 \leftarrow O_5$	M3O5_LINE
$M_3 \leftarrow O_6$	M3O6_LINE
$M_3 \leftarrow O_7$	M3O7_LINE
$M_3 \leftarrow P_1$	M3P1_LINE
$M_3 \leftarrow P_2$	M3P2_LINE
$M_3 \leftarrow P_3$	M3P3_LINE
$M_3 \leftarrow P_4$	M3P4_LINE
$M_3 \leftarrow P_5$	M3P5_LINE
$M_3 \leftarrow Q_1$	M3Q1_LINE
$M_4 \leftarrow M_5$	M4M5_LINE
$M_4 \leftarrow N_1$	M4N1_LINE
$M_4 \leftarrow N_2$	M4N2_LINE
$M_4 \leftarrow N_3$	M4N3_LINE
$M_4 \leftarrow N_4$	M4N4_LINE
$M_4 \leftarrow N_5$	M4N5_LINE
$M_4 \leftarrow N_6$	M4N6_LINE
$M_4 \leftarrow N_7$	M4N7_LINE
$M_4 \leftarrow O_1$	M4O1_LINE
$M_4 \leftarrow O_2$	M4O2_LINE
$M_4 \leftarrow O_3$	M4O3_LINE
$M_4 \leftarrow O_4$	M4O4_LINE
$M_4 \leftarrow O_5$	M4O5_LINE
$M_4 \leftarrow O_6$	M4O6_LINE
$M_4 \leftarrow O_7$	M4O7_LINE
$M_4 \leftarrow P_1$	M4P1_LINE
$M_4 \leftarrow P_2$	M4P2_LINE
$M_4 \leftarrow P_3$	M4P3_LINE
$M_4 \leftarrow P_4$	M4P4_LINE
$M_4 \leftarrow P_5$	M4P5_LINE
$M_5 \leftarrow N_1$	M5N1_LINE
$M_5 \leftarrow N_2$	M5N2_LINE
$M_5 \leftarrow N_3$	M5N3_LINE
$M_5 \leftarrow N_4$	M5N4_LINE
$M_5 \leftarrow N_5$	M5N5_LINE
$M_5 \leftarrow N_6$	M5N6_LINE
$M_5 \leftarrow N_7$	M5N7_LINE
$M_5 \leftarrow O_1$	M5O1_LINE
$M_5 \leftarrow O_2$	M5O2_LINE
$M_5 \leftarrow O_3$	M5O3_LINE
$M_5 \leftarrow O_4$	M5O4_LINE
$M_5 \leftarrow O_5$	M5O5_LINE
$M_5 \leftarrow O_6$	M5O6_LINE

Transition	IUPAC macro
$M_5 \leftarrow O_7$	M5O7_LINE
$M_5 \leftarrow P_1$	M5P1_LINE
$M_5 \leftarrow P_2$	M5P2_LINE
$M_5 \leftarrow P_3$	M5P3_LINE
$M_5 \leftarrow P_4$	M5P4_LINE
$M_5 \leftarrow P_5$	M5P5_LINE
$N_1 \leftarrow N_2$	N1N2_LINE
$N_1 \leftarrow N_3$	N1N3_LINE
$N_1 \leftarrow N_4$	N1N4_LINE
$N_1 \leftarrow N_5$	N1N5_LINE
$N_1 \leftarrow N_6$	N1N6_LINE
$N_1 \leftarrow N_7$	N1N7_LINE
$N_1 \leftarrow O_1$	N1O1_LINE
$N_1 \leftarrow O_2$	N1O2_LINE
$N_1 \leftarrow O_3$	N1O3_LINE
$N_1 \leftarrow O_4$	N1O4_LINE
$N_1 \leftarrow O_5$	N1O5_LINE
$N_1 \leftarrow O_6$	N1O6_LINE
$N_1 \leftarrow O_7$	N1O7_LINE
$N_1 \leftarrow P_1$	N1P1_LINE
$N_1 \leftarrow P_2$	N1P2_LINE
$N_1 \leftarrow P_3$	N1P3_LINE
$N_1 \leftarrow P_4$	N1P4_LINE
$N_1 \leftarrow P_5$	N1P5_LINE
$N_2 \leftarrow N_3$	N2N3_LINE
$N_2 \leftarrow N_4$	N2N4_LINE
$N_2 \leftarrow N_5$	N2N5_LINE
$N_2 \leftarrow N_6$	N2N6_LINE
$N_2 \leftarrow N_7$	N2N7_LINE
$N_2 \leftarrow O_1$	N2O1_LINE
$N_2 \leftarrow O_2$	N2O2_LINE
$N_2 \leftarrow O_3$	N2O3_LINE
$N_2 \leftarrow O_4$	N2O4_LINE
$N_2 \leftarrow O_5$	N2O5_LINE
$N_2 \leftarrow O_6$	N2O6_LINE
$N_2 \leftarrow O_7$	N2O7_LINE
$N_2 \leftarrow P_1$	N2P1_LINE
$N_2 \leftarrow P_2$	N2P2_LINE
$N_2 \leftarrow P_3$	N2P3_LINE
$N_2 \leftarrow P_4$	N2P4_LINE
$N_2 \leftarrow P_5$	N2P5_LINE
$N_3 \leftarrow N_4$	N3N4_LINE

Transition	IUPAC macro
$N_3 \leftarrow N_5$	N3N5_LINE
$N_3 \leftarrow N_6$	N3N6_LINE
$N_3 \leftarrow N_7$	N3N7_LINE
$N_3 \leftarrow O_1$	N3O1_LINE
$N_3 \leftarrow O_2$	N3O2_LINE
$N_3 \leftarrow O_3$	N3O3_LINE
$N_3 \leftarrow O_4$	N3O4_LINE
$N_3 \leftarrow O_5$	N3O5_LINE
$N_3 \leftarrow O_6$	N3O6_LINE
$N_3 \leftarrow O_7$	N3O7_LINE
$N_3 \leftarrow P_1$	N3P1_LINE
$N_3 \leftarrow P_2$	N3P2_LINE
$N_3 \leftarrow P_3$	N3P3_LINE
$N_3 \leftarrow P_4$	N3P4_LINE
$N_3 \leftarrow P_5$	N3P5_LINE
$N_4 \leftarrow N_5$	N4N5_LINE
$N_4 \leftarrow N_6$	N4N6_LINE
$N_4 \leftarrow N_7$	N4N7_LINE
$N_4 \leftarrow O_1$	N4O1_LINE
$N_4 \leftarrow O_2$	N4O2_LINE
$N_4 \leftarrow O_3$	N4O3_LINE
$N_4 \leftarrow O_4$	N4O4_LINE
$N_4 \leftarrow O_5$	N4O5_LINE
$N_4 \leftarrow O_6$	N4O6_LINE
$N_4 \leftarrow O_7$	N4O7_LINE
$N_4 \leftarrow P_1$	N4P1_LINE
$N_4 \leftarrow P_2$	N4P2_LINE
$N_4 \leftarrow P_3$	N4P3_LINE
$N_4 \leftarrow P_4$	N4P4_LINE
$N_4 \leftarrow P_5$	N4P5_LINE
$N_5 \leftarrow N_6$	N5N6_LINE
$N_5 \leftarrow N_7$	N5N7_LINE
$N_5 \leftarrow O_1$	N5O1_LINE
$N_5 \leftarrow O_2$	N5O2_LINE
$N_5 \leftarrow O_3$	N5O3_LINE
$N_5 \leftarrow O_4$	N5O4_LINE
$N_5 \leftarrow O_5$	N5O5_LINE
$N_5 \leftarrow O_6$	N5O6_LINE
$N_5 \leftarrow O_7$	N5O7_LINE
$N_5 \leftarrow P_1$	N5P1_LINE
$N_5 \leftarrow P_2$	N5P2_LINE
$N_5 \leftarrow P_3$	N5P3_LINE

Transition	IUPAC macro
$N_5 \leftarrow P_4$	N5P4_LINE
$N_5 \leftarrow P_5$	N5P5_LINE
$N_6 \leftarrow N_7$	N6N7_LINE
$N_6 \leftarrow O_1$	N6O1_LINE
$N_6 \leftarrow O_2$	N6O2_LINE
$N_6 \leftarrow O_3$	N6O3_LINE
$N_6 \leftarrow O_4$	N6O4_LINE
$N_6 \leftarrow O_5$	N6O5_LINE
$N_6 \leftarrow O_6$	N6O6_LINE
$N_6 \leftarrow O_7$	N6O7_LINE
$N_6 \leftarrow P_1$	N6P1_LINE
$N_6 \leftarrow P_2$	N6P2_LINE
$N_6 \leftarrow P_3$	N6P3_LINE
$N_6 \leftarrow P_4$	N6P4_LINE
$N_6 \leftarrow P_5$	N6P5_LINE
$N_7 \leftarrow O_1$	N7O1_LINE
$N_7 \leftarrow O_2$	N7O2_LINE
$N_7 \leftarrow O_3$	N7O3_LINE
$N_7 \leftarrow O_4$	N7O4_LINE
$N_7 \leftarrow O_5$	N7O5_LINE
$N_7 \leftarrow O_6$	N7O6_LINE
$N_7 \leftarrow O_7$	N7O7_LINE
$N_7 \leftarrow P_1$	N7P1_LINE
$N_7 \leftarrow P_2$	N7P2_LINE
$N_7 \leftarrow P_3$	N7P3_LINE
$N_7 \leftarrow P_4$	N7P4_LINE
$N_7 \leftarrow P_5$	N7P5_LINE
$O_1 \leftarrow O_2$	O1O2_LINE
$O_1 \leftarrow O_3$	O1O3_LINE
$O_1 \leftarrow O_4$	O1O4_LINE
$O_1 \leftarrow O_5$	O1O5_LINE
$O_1 \leftarrow O_6$	O1O6_LINE
$O_1 \leftarrow O_7$	O1O7_LINE
$O_1 \leftarrow P_1$	O1P1_LINE
$O_1 \leftarrow P_2$	O1P2_LINE
$O_1 \leftarrow P_3$	O1P3_LINE
$O_1 \leftarrow P_4$	O1P4_LINE
$O_1 \leftarrow P_5$	O1P5_LINE
$O_2 \leftarrow O_3$	O2O3_LINE
$O_2 \leftarrow O_4$	O2O4_LINE
$O_2 \leftarrow O_5$	O2O5_LINE
$O_2 \leftarrow O_6$	O2O6_LINE

Transition	IUPAC macro
$O_2 \leftarrow O_7$	O2O7_LINE
$O_2 \leftarrow P_1$	O2P1_LINE
$O_2 \leftarrow P_2$	O2P2_LINE
$O_2 \leftarrow P_3$	O2P3_LINE
$O_2 \leftarrow P_4$	O2P4_LINE
$O_2 \leftarrow P_5$	O2P5_LINE
$O_3 \leftarrow O_4$	O3O4_LINE
$O_3 \leftarrow O_5$	O3O5_LINE
$O_3 \leftarrow O_6$	O3O6_LINE
$O_3 \leftarrow O_7$	O3O7_LINE
$O_3 \leftarrow P_1$	O3P1_LINE
$O_3 \leftarrow P_2$	O3P2_LINE
$O_3 \leftarrow P_3$	O3P3_LINE
$O_3 \leftarrow P_4$	O3P4_LINE
$O_3 \leftarrow P_5$	O3P5_LINE
$O_4 \leftarrow O_5$	O4O5_LINE
$O_4 \leftarrow O_6$	O4O6_LINE
$O_4 \leftarrow O_7$	O4O7_LINE
$O_4 \leftarrow P_1$	O4P1_LINE
$O_4 \leftarrow P_2$	O4P2_LINE
$O_4 \leftarrow P_3$	O4P3_LINE
$O_4 \leftarrow P_4$	O4P4_LINE
$O_4 \leftarrow P_5$	O4P5_LINE
$O_5 \leftarrow O_6$	O5O6_LINE
$O_5 \leftarrow O_7$	O5O7_LINE
$O_5 \leftarrow P_1$	O5P1_LINE
$O_5 \leftarrow P_2$	O5P2_LINE
$O_5 \leftarrow P_3$	O5P3_LINE
$O_5 \leftarrow P_4$	O5P4_LINE
$O_5 \leftarrow P_5$	O5P5_LINE
$O_6 \leftarrow O_7$	O6O7_LINE
$O_6 \leftarrow P_4$	O6P4_LINE
$O_6 \leftarrow P_5$	O6P5_LINE
$O_7 \leftarrow P_4$	O7P4_LINE
$O_7 \leftarrow P_5$	O7P5_LINE
$P_1 \leftarrow P_2$	P1P2_LINE
$P_1 \leftarrow P_3$	P1P3_LINE
$P_1 \leftarrow P_4$	P1P4_LINE
$P_1 \leftarrow P_5$	P1P5_LINE
$P_2 \leftarrow P_3$	P2P3_LINE
$P_2 \leftarrow P_4$	P2P4_LINE
$P_2 \leftarrow P_5$	P2P5_LINE

Transition	IUPAC macro
$P_3 \leftarrow P_4$	P3P4_LINE
$P_3 \leftarrow P_5$	P3P5_LINE

A.2.2 Siegbahn X-ray fluorescence line macros

The following table contains the Siegbahn macros: these are all mapped to a corresponding IUPAC macro from the previous table.

Siegbahn notation	Siegbahn macro	IUPAC macro
$K_{\alpha 1}$	KA1_LINE	KL3_LINE
$K_{\alpha 2}$	KA2_LINE	KL2_LINE
$K_{\alpha 3}$	KA3_LINE	KL1_LINE
$K_{\beta 1}$	KB1_LINE	KM3_LINE
$K_{\beta 2}$	KB2_LINE	KN3_LINE
$K_{\beta 3}$	KB3_LINE	KM2_LINE
$K_{\beta 4}$	KB4_LINE	KN5_LINE
$K_{\beta 5}$	KB5_LINE	KM5_LINE
$L_{\alpha 1}$	LA1_LINE	L3M5_LINE
$L_{\alpha 2}$	LA2_LINE	L3M4_LINE
$L_{\beta 1}$	LB1_LINE	L2M4_LINE
$L_{\beta 2}$	LB2_LINE	L3N5_LINE
$L_{\beta 3}$	LB3_LINE	L1M3_LINE
$L_{\beta 4}$	LB4_LINE	L1M2_LINE
$L_{\beta 5}$	LB5_LINE	L3O45_LINE
$L_{\beta 6}$	LB6_LINE	L3N1_LINE
$L_{\beta 7}$	LB7_LINE	L3O1_LINE
$L_{\beta 9}$	LB9_LINE	L1M5_LINE
$L_{\beta 10}$	LB10_LINE	L1M4_LINE
$L_{\beta 15}$	LB15_LINE	L3N4_LINE
$L_{\beta 17}$	LB17_LINE	L2M3_LINE
$L_{\gamma 1}$	LG1_LINE	L2N4_LINE
$L_{\gamma 2}$	LG2_LINE	L1N2_LINE
$L_{\gamma 3}$	LG3_LINE	L1N3_LINE
$L_{\gamma 4}$	LG4_LINE	L1O3_LINE
$L_{\gamma 5}$	LG5_LINE	L2N1_LINE
$L_{\gamma 6}$	LG6_LINE	L2O4_LINE
$L_{\gamma 8}$	LG8_LINE	L2O1_LINE
L_{η}	LE_LINE	L2M1_LINE
L_{h}	LH_LINE	L2M1_LINE
L_{l}	LL_LINE	L3M1_LINE
L_{s}	LS_LINE	L3M3_LINE
L_{t}	LT_LINE	L3M2_LINE

Siegbahn notation	Siegbahn macro	IUPAC macro
L_u	LU_LINE	L3N6_LINE
L_v	LV_LINE	L2N6_LINE
$M_{\alpha 1}$	MA1_LINE	M5N7_LINE
$M_{\alpha 2}$	MA2_LINE	M5N6_LINE
M_β	MB_LINE	M4N6_LINE
M_γ	MG_LINE	M3N5_LINE

A.2.3 Obsolete X-ray fluorescence macros

The following macros based on the Siegbahn notation are presented here strictly for the sake of completeness: though it is quite unlikely that they will be removed in future versions, their usage is deprecated. Since they refer to groups of lines you will up receiving values that are calculated as averages or sums over a number of individual lines, which may not have an obvious counterpart in experimental data.

Siegbahn notation	Obsolete macro
K_α	KA_LINE
K_β	KB_LINE
L_α	LA_LINE
L_β	LB_LINE

A.3 Shell macros

These macros denote X-ray levels ('shells') conforming with the many-electron description of electronic structure as is commonly used in X-ray spectrometry. They are used in the `CS_Photo_Partial`, `FluorYield`, `AugerYield`, `EdgeEnergy`, `JumpFactor`, `AtomicLevelWidth`, `ComptonProfile_Partial` and `ElectronConfig`.

Level	Electron configuration	Shell macro
K	$1s^{-1}$	K_SHELL
L_1	$2s^{-1}$	L1_SHELL
L_2	${2p_{1/2}}^{-1}$	L2_SHELL
L_3	${2p_{3/2}}^{-1}$	L3_SHELL
M_1	$3s^{-1}$	M1_SHELL
M_2	${3p_{1/2}}^{-1}$	M2_SHELL
M_3	${3p_{3/2}}^{-1}$	M3_SHELL
M_4	${3d_{3/2}}^{-1}$	M4_SHELL
M_5	${3d_{5/2}}^{-1}$	M5_SHELL
N_1	$4s^{-1}$	N1_SHELL
N_2	${4p_{1/2}}^{-1}$	N2_SHELL
N_3	${4p_{3/2}}^{-1}$	N3_SHELL

Level	Electron configuration	Shell macro
N_4	${4d_{3/2}}^{-1}$	N4_SHELL
N_5	${4d_{5/2}}^{-1}$	N5_SHELL
N_6	${4f_{5/2}}^{-1}$	N6_SHELL
N_7	${4f_{7/2}}^{-1}$	N7_SHELL
O_1	$5s^{-1}$	O1_SHELL
O_2	${5p_{1/2}}^{-1}$	O2_SHELL
O_3	${5p_{3/2}}^{-1}$	O3_SHELL
O_4	${5d_{3/2}}^{-1}$	O4_SHELL
O_5	${5d_{5/2}}^{-1}$	O5_SHELL
O_6	${5f_{5/2}}^{-1}$	O6_SHELL
O_7	${5f_{7/2}}^{-1}$	O7_SHELL
P_1	$6s^{-1}$	P1_SHELL
P_2	${6p_{1/2}}^{-1}$	P2_SHELL
P_3	${6p_{3/2}}^{-1}$	P3_SHELL
P_4	${6d_{3/2}}^{-1}$	P4_SHELL
P_5	${6d_{5/2}}^{-1}$	P5_SHELL
Q_1	$7s^{-1}$	Q1_SHELL
Q_2	${7p_{1/2}}^{-1}$	Q2_SHELL
Q_3	${7p_{3/2}}^{-1}$	Q3_SHELL

A.4 Coster-Kronig transition macros

The following macros are used to select a specific Coster-Kronig (CK) transition with the `CosKronTransProb` function. Currently we are only supporting L- and M-shell CK transitions.

Transition	CK macro
$L_1 \leftarrow L_2$	FL12_TRANS
$L_1 \leftarrow L_3$	FL13_TRANS
$L_2 \leftarrow L_3$	FL23_TRANS
$M_1 \leftarrow M_2$	FM12_TRANS
$M_1 \leftarrow M_3$	FM13_TRANS
$M_1 \leftarrow M_4$	FM14_TRANS
$M_1 \leftarrow M_5$	FM15_TRANS
$M_2 \leftarrow M_3$	FM23_TRANS
$M_2 \leftarrow M_4$	FM24_TRANS
$M_2 \leftarrow M_5$	FM25_TRANS
$M_3 \leftarrow M_4$	FM34_TRANS
$M_3 \leftarrow M_5$	FM35_TRANS
$M_4 \leftarrow M_5$	FM45_TRANS

A.5 Auger transition macros

The `AugerRate` function requires one of the following macros as argument. Some of these macros (e.g. `L2_L3M1_AUGER`) correspond to Coster-Kronig transitions (a special case of the Auger effect), these will return zero. This list contains only those transitions that are relevant for X-ray fluorescence production involving the K-, L- and M-shells: do not expect to find macros like `K_N2O3_AUGER` although these Auger transitions certainly exist!

Excited shell	Ionized shells	Auger macro
K	L_1 & L_1	K_L1L1_AUGER
K	L_1 & L_2	K_L1L2_AUGER
K	L_1 & L_3	K_L1L3_AUGER
K	L_1 & M_1	K_L1M1_AUGER
K	L_1 & M_2	K_L1M2_AUGER
K	L_1 & M_3	K_L1M3_AUGER
K	L_1 & M_4	K_L1M4_AUGER
K	L_1 & M_5	K_L1M5_AUGER
K	L_1 & N_1	K_L1N1_AUGER
K	L_1 & N_2	K_L1N2_AUGER
K	L_1 & N_3	K_L1N3_AUGER
K	L_1 & N_4	K_L1N4_AUGER
K	L_1 & N_5	K_L1N5_AUGER
K	L_1 & N_6	K_L1N6_AUGER
K	L_1 & N_7	K_L1N7_AUGER
K	L_1 & O_1	K_L1O1_AUGER
K	L_1 & O_2	K_L1O2_AUGER
K	L_1 & O_3	K_L1O3_AUGER
K	L_1 & O_4	K_L1O4_AUGER
K	L_1 & O_5	K_L1O5_AUGER
K	L_1 & O_6	K_L1O6_AUGER
K	L_1 & O_7	K_L1O7_AUGER
K	L_1 & P_1	K_L1P1_AUGER
K	L_1 & P_2	K_L1P2_AUGER
K	L_1 & P_3	K_L1P3_AUGER
K	L_1 & P_4	K_L1P4_AUGER
K	L_1 & P_5	K_L1P5_AUGER
K	L_1 & Q_1	K_L1Q1_AUGER
K	L_1 & Q_2	K_L1Q2_AUGER
K	L_1 & Q_3	K_L1Q3_AUGER
K	L_2 & L_2	K_L2L2_AUGER
K	L_2 & L_3	K_L2L3_AUGER
K	L_2 & M_1	K_L2M1_AUGER
K	L_2 & M_2	K_L2M2_AUGER

Excited shell	Ionized shells	Auger macro
K	L_2 & M_3	K_L2M3_AUGER
K	L_2 & M_4	K_L2M4_AUGER
K	L_2 & M_5	K_L2M5_AUGER
K	L_2 & N_1	K_L2N1_AUGER
K	L_2 & N_2	K_L2N2_AUGER
K	L_2 & N_3	K_L2N3_AUGER
K	L_2 & N_4	K_L2N4_AUGER
K	L_2 & N_5	K_L2N5_AUGER
K	L_2 & N_6	K_L2N6_AUGER
K	L_2 & N_7	K_L2N7_AUGER
K	L_2 & O_1	K_L2O1_AUGER
K	L_2 & O_2	K_L2O2_AUGER
K	L_2 & O_3	K_L2O3_AUGER
K	L_2 & O_4	K_L2O4_AUGER
K	L_2 & O_5	K_L2O5_AUGER
K	L_2 & O_6	K_L2O6_AUGER
K	L_2 & O_7	K_L2O7_AUGER
K	L_2 & P_1	K_L2P1_AUGER
K	L_2 & P_2	K_L2P2_AUGER
K	L_2 & P_3	K_L2P3_AUGER
K	L_2 & P_4	K_L2P4_AUGER
K	L_2 & P_5	K_L2P5_AUGER
K	L_2 & Q_1	K_L2Q1_AUGER
K	L_2 & Q_2	K_L2Q2_AUGER
K	L_2 & Q_3	K_L2Q3_AUGER
K	L_3 & L_3	K_L3L3_AUGER
K	L_3 & M_1	K_L3M1_AUGER
K	L_3 & M_2	K_L3M2_AUGER
K	L_3 & M_3	K_L3M3_AUGER
K	L_3 & M_4	K_L3M4_AUGER
K	L_3 & M_5	K_L3M5_AUGER
K	L_3 & N_1	K_L3N1_AUGER
K	L_3 & N_2	K_L3N2_AUGER
K	L_3 & N_3	K_L3N3_AUGER
K	L_3 & N_4	K_L3N4_AUGER
K	L_3 & N_5	K_L3N5_AUGER
K	L_3 & N_6	K_L3N6_AUGER
K	L_3 & N_7	K_L3N7_AUGER
K	L_3 & O_1	K_L3O1_AUGER
K	L_3 & O_2	K_L3O2_AUGER
K	L_3 & O_3	K_L3O3_AUGER
K	L_3 & O_4	K_L3O4_AUGER

Excited shell	Ionized shells	Auger macro
K	L_3 & O_5	K_L3O5_AUGER
K	L_3 & O_6	K_L3O6_AUGER
K	L_3 & O_7	K_L3O7_AUGER
K	L_3 & P_1	K_L3P1_AUGER
K	L_3 & P_2	K_L3P2_AUGER
K	L_3 & P_3	K_L3P3_AUGER
K	L_3 & P_4	K_L3P4_AUGER
K	L_3 & P_5	K_L3P5_AUGER
K	L_3 & Q_1	K_L3Q1_AUGER
K	L_3 & Q_2	K_L3Q2_AUGER
K	L_3 & Q_3	K_L3Q3_AUGER
K	M_1 & M_1	K_M1M1_AUGER
K	M_1 & M_2	K_M1M2_AUGER
K	M_1 & M_3	K_M1M3_AUGER
K	M_1 & M_4	K_M1M4_AUGER
K	M_1 & M_5	K_M1M5_AUGER
K	M_1 & N_1	K_M1N1_AUGER
K	M_1 & N_2	K_M1N2_AUGER
K	M_1 & N_3	K_M1N3_AUGER
K	M_1 & N_4	K_M1N4_AUGER
K	M_1 & N_5	K_M1N5_AUGER
K	M_1 & N_6	K_M1N6_AUGER
K	M_1 & N_7	K_M1N7_AUGER
K	M_1 & O_1	K_M1O1_AUGER
K	M_1 & O_2	K_M1O2_AUGER
K	M_1 & O_3	K_M1O3_AUGER
K	M_1 & O_4	K_M1O4_AUGER
K	M_1 & O_5	K_M1O5_AUGER
K	M_1 & O_6	K_M1O6_AUGER
K	M_1 & O_7	K_M1O7_AUGER
K	M_1 & P_1	K_M1P1_AUGER
K	M_1 & P_2	K_M1P2_AUGER
K	M_1 & P_3	K_M1P3_AUGER
K	M_1 & P_4	K_M1P4_AUGER
K	M_1 & P_5	K_M1P5_AUGER
K	M_1 & Q_1	K_M1Q1_AUGER
K	M_1 & Q_2	K_M1Q2_AUGER
K	M_1 & Q_3	K_M1Q3_AUGER
K	M_2 & M_2	K_M2M2_AUGER
K	M_2 & M_3	K_M2M3_AUGER
K	M_2 & M_4	K_M2M4_AUGER
K	M_2 & M_5	K_M2M5_AUGER

Excited shell	Ionized shells	Auger macro
K	M_2 & N_1	K_M2N1_AUGER
K	M_2 & N_2	K_M2N2_AUGER
K	M_2 & N_3	K_M2N3_AUGER
K	M_2 & N_4	K_M2N4_AUGER
K	M_2 & N_5	K_M2N5_AUGER
K	M_2 & N_6	K_M2N6_AUGER
K	M_2 & N_7	K_M2N7_AUGER
K	M_2 & O_1	K_M2O1_AUGER
K	M_2 & O_2	K_M2O2_AUGER
K	M_2 & O_3	K_M2O3_AUGER
K	M_2 & O_4	K_M2O4_AUGER
K	M_2 & O_5	K_M2O5_AUGER
K	M_2 & O_6	K_M2O6_AUGER
K	M_2 & O_7	K_M2O7_AUGER
K	M_2 & P_1	K_M2P1_AUGER
K	M_2 & P_2	K_M2P2_AUGER
K	M_2 & P_3	K_M2P3_AUGER
K	M_2 & P_4	K_M2P4_AUGER
K	M_2 & P_5	K_M2P5_AUGER
K	M_2 & Q_1	K_M2Q1_AUGER
K	M_2 & Q_2	K_M2Q2_AUGER
K	M_2 & Q_3	K_M2Q3_AUGER
K	M_3 & M_3	K_M3M3_AUGER
K	M_3 & M_4	K_M3M4_AUGER
K	M_3 & M_5	K_M3M5_AUGER
K	M_3 & N_1	K_M3N1_AUGER
K	M_3 & N_2	K_M3N2_AUGER
K	M_3 & N_3	K_M3N3_AUGER
K	M_3 & N_4	K_M3N4_AUGER
K	M_3 & N_5	K_M3N5_AUGER
K	M_3 & N_6	K_M3N6_AUGER
K	M_3 & N_7	K_M3N7_AUGER
K	M_3 & O_1	K_M3O1_AUGER
K	M_3 & O_2	K_M3O2_AUGER
K	M_3 & O_3	K_M3O3_AUGER
K	M_3 & O_4	K_M3O4_AUGER
K	M_3 & O_5	K_M3O5_AUGER
K	M_3 & O_6	K_M3O6_AUGER
K	M_3 & O_7	K_M3O7_AUGER
K	M_3 & P_1	K_M3P1_AUGER
K	M_3 & P_2	K_M3P2_AUGER
K	M_3 & P_3	K_M3P3_AUGER

Excited shell	Ionized shells	Auger macro
K	M_3 & P_4	K_M3P4_AUGER
K	M_3 & P_5	K_M3P5_AUGER
K	M_3 & Q_1	K_M3Q1_AUGER
K	M_3 & Q_2	K_M3Q2_AUGER
K	M_3 & Q_3	K_M3Q3_AUGER
K	M_4 & M_4	K_M4M4_AUGER
K	M_4 & M_5	K_M4M5_AUGER
K	M_4 & N_1	K_M4N1_AUGER
K	M_4 & N_2	K_M4N2_AUGER
K	M_4 & N_3	K_M4N3_AUGER
K	M_4 & N_4	K_M4N4_AUGER
K	M_4 & N_5	K_M4N5_AUGER
K	M_4 & N_6	K_M4N6_AUGER
K	M_4 & N_7	K_M4N7_AUGER
K	M_4 & O_1	K_M4O1_AUGER
K	M_4 & O_2	K_M4O2_AUGER
K	M_4 & O_3	K_M4O3_AUGER
K	M_4 & O_4	K_M4O4_AUGER
K	M_4 & O_5	K_M4O5_AUGER
K	M_4 & O_6	K_M4O6_AUGER
K	M_4 & O_7	K_M4O7_AUGER
K	M_4 & P_1	K_M4P1_AUGER
K	M_4 & P_2	K_M4P2_AUGER
K	M_4 & P_3	K_M4P3_AUGER
K	M_4 & P_4	K_M4P4_AUGER
K	M_4 & P_5	K_M4P5_AUGER
K	M_4 & Q_1	K_M4Q1_AUGER
K	M_4 & Q_2	K_M4Q2_AUGER
K	M_4 & Q_3	K_M4Q3_AUGER
K	M_5 & M_5	K_M5M5_AUGER
K	M_5 & N_1	K_M5N1_AUGER
K	M_5 & N_2	K_M5N2_AUGER
K	M_5 & N_3	K_M5N3_AUGER
K	M_5 & N_4	K_M5N4_AUGER
K	M_5 & N_5	K_M5N5_AUGER
K	M_5 & N_6	K_M5N6_AUGER
K	M_5 & N_7	K_M5N7_AUGER
K	M_5 & O_1	K_M5O1_AUGER
K	M_5 & O_2	K_M5O2_AUGER
K	M_5 & O_3	K_M5O3_AUGER
K	M_5 & O_4	K_M5O4_AUGER
K	M_5 & O_5	K_M5O5_AUGER

Excited shell	Ionized shells	Auger macro
K	M_5 & O_6	`K_M5O6_AUGER`
K	M_5 & O_7	`K_M5O7_AUGER`
K	M_5 & P_1	`K_M5P1_AUGER`
K	M_5 & P_2	`K_M5P2_AUGER`
K	M_5 & P_3	`K_M5P3_AUGER`
K	M_5 & P_4	`K_M5P4_AUGER`
K	M_5 & P_5	`K_M5P5_AUGER`
K	M_5 & Q_1	`K_M5Q1_AUGER`
K	M_5 & Q_2	`K_M5Q2_AUGER`
K	M_5 & Q_3	`K_M5Q3_AUGER`
L_1	L_2 & L_2	`L1_L2L2_AUGER`
L_1	L_2 & L_3	`L1_L2L3_AUGER`
L_1	L_2 & M_1	`L1_L2M1_AUGER`
L_1	L_2 & M_2	`L1_L2M2_AUGER`
L_1	L_2 & M_3	`L1_L2M3_AUGER`
L_1	L_2 & M_4	`L1_L2M4_AUGER`
L_1	L_2 & M_5	`L1_L2M5_AUGER`
L_1	L_2 & N_1	`L1_L2N1_AUGER`
L_1	L_2 & N_2	`L1_L2N2_AUGER`
L_1	L_2 & N_3	`L1_L2N3_AUGER`
L_1	L_2 & N_4	`L1_L2N4_AUGER`
L_1	L_2 & N_5	`L1_L2N5_AUGER`
L_1	L_2 & N_6	`L1_L2N6_AUGER`
L_1	L_2 & N_7	`L1_L2N7_AUGER`
L_1	L_2 & O_1	`L1_L2O1_AUGER`
L_1	L_2 & O_2	`L1_L2O2_AUGER`
L_1	L_2 & O_3	`L1_L2O3_AUGER`
L_1	L_2 & O_4	`L1_L2O4_AUGER`
L_1	L_2 & O_5	`L1_L2O5_AUGER`
L_1	L_2 & O_6	`L1_L2O6_AUGER`
L_1	L_2 & O_7	`L1_L2O7_AUGER`
L_1	L_2 & P_1	`L1_L2P1_AUGER`
L_1	L_2 & P_2	`L1_L2P2_AUGER`
L_1	L_2 & P_3	`L1_L2P3_AUGER`
L_1	L_2 & P_4	`L1_L2P4_AUGER`
L_1	L_2 & P_5	`L1_L2P5_AUGER`
L_1	L_2 & Q_1	`L1_L2Q1_AUGER`
L_1	L_2 & Q_2	`L1_L2Q2_AUGER`
L_1	L_2 & Q_3	`L1_L2Q3_AUGER`
L_1	L_3 & L_2	`L1_L3L2_AUGER`
L_1	L_3 & L_3	`L1_L3L3_AUGER`
L_1	L_3 & M_1	`L1_L3M1_AUGER`

Excited shell	Ionized shells	Auger macro
L_1	L_3 & M_2	L1_L3M2_AUGER
L_1	L_3 & M_3	L1_L3M3_AUGER
L_1	L_3 & M_4	L1_L3M4_AUGER
L_1	L_3 & M_5	L1_L3M5_AUGER
L_1	L_3 & N_1	L1_L3N1_AUGER
L_1	L_3 & N_2	L1_L3N2_AUGER
L_1	L_3 & N_3	L1_L3N3_AUGER
L_1	L_3 & N_4	L1_L3N4_AUGER
L_1	L_3 & N_5	L1_L3N5_AUGER
L_1	L_3 & N_6	L1_L3N6_AUGER
L_1	L_3 & N_7	L1_L3N7_AUGER
L_1	L_3 & O_1	L1_L3O1_AUGER
L_1	L_3 & O_2	L1_L3O2_AUGER
L_1	L_3 & O_3	L1_L3O3_AUGER
L_1	L_3 & O_4	L1_L3O4_AUGER
L_1	L_3 & O_5	L1_L3O5_AUGER
L_1	L_3 & O_6	L1_L3O6_AUGER
L_1	L_3 & O_7	L1_L3O7_AUGER
L_1	L_3 & P_1	L1_L3P1_AUGER
L_1	L_3 & P_2	L1_L3P2_AUGER
L_1	L_3 & P_3	L1_L3P3_AUGER
L_1	L_3 & P_4	L1_L3P4_AUGER
L_1	L_3 & P_5	L1_L3P5_AUGER
L_1	L_3 & Q_1	L1_L3Q1_AUGER
L_1	L_3 & Q_2	L1_L3Q2_AUGER
L_1	L_3 & Q_3	L1_L3Q3_AUGER
L_1	M_1 & L_2	L1_M1L2_AUGER
L_1	M_1 & L_3	L1_M1L3_AUGER
L_1	M_1 & M_1	L1_M1M1_AUGER
L_1	M_1 & M_2	L1_M1M2_AUGER
L_1	M_1 & M_3	L1_M1M3_AUGER
L_1	M_1 & M_4	L1_M1M4_AUGER
L_1	M_1 & M_5	L1_M1M5_AUGER
L_1	M_1 & N_1	L1_M1N1_AUGER
L_1	M_1 & N_2	L1_M1N2_AUGER
L_1	M_1 & N_3	L1_M1N3_AUGER
L_1	M_1 & N_4	L1_M1N4_AUGER
L_1	M_1 & N_5	L1_M1N5_AUGER
L_1	M_1 & N_6	L1_M1N6_AUGER
L_1	M_1 & N_7	L1_M1N7_AUGER
L_1	M_1 & O_1	L1_M1O1_AUGER
L_1	M_1 & O_2	L1_M1O2_AUGER

Excited shell	Ionized shells	Auger macro
L_1	M_1 & O_3	L1_M1O3_AUGER
L_1	M_1 & O_4	L1_M1O4_AUGER
L_1	M_1 & O_5	L1_M1O5_AUGER
L_1	M_1 & O_6	L1_M1O6_AUGER
L_1	M_1 & O_7	L1_M1O7_AUGER
L_1	M_1 & P_1	L1_M1P1_AUGER
L_1	M_1 & P_2	L1_M1P2_AUGER
L_1	M_1 & P_3	L1_M1P3_AUGER
L_1	M_1 & P_4	L1_M1P4_AUGER
L_1	M_1 & P_5	L1_M1P5_AUGER
L_1	M_1 & Q_1	L1_M1Q1_AUGER
L_1	M_1 & Q_2	L1_M1Q2_AUGER
L_1	M_1 & Q_3	L1_M1Q3_AUGER
L_1	M_2 & L_2	L1_M2L2_AUGER
L_1	M_2 & L_3	L1_M2L3_AUGER
L_1	M_2 & M_1	L1_M2M1_AUGER
L_1	M_2 & M_2	L1_M2M2_AUGER
L_1	M_2 & M_3	L1_M2M3_AUGER
L_1	M_2 & M_4	L1_M2M4_AUGER
L_1	M_2 & M_5	L1_M2M5_AUGER
L_1	M_2 & N_1	L1_M2N1_AUGER
L_1	M_2 & N_2	L1_M2N2_AUGER
L_1	M_2 & N_3	L1_M2N3_AUGER
L_1	M_2 & N_4	L1_M2N4_AUGER
L_1	M_2 & N_5	L1_M2N5_AUGER
L_1	M_2 & N_6	L1_M2N6_AUGER
L_1	M_2 & N_7	L1_M2N7_AUGER
L_1	M_2 & O_1	L1_M2O1_AUGER
L_1	M_2 & O_2	L1_M2O2_AUGER
L_1	M_2 & O_3	L1_M2O3_AUGER
L_1	M_2 & O_4	L1_M2O4_AUGER
L_1	M_2 & O_5	L1_M2O5_AUGER
L_1	M_2 & O_6	L1_M2O6_AUGER
L_1	M_2 & O_7	L1_M2O7_AUGER
L_1	M_2 & P_1	L1_M2P1_AUGER
L_1	M_2 & P_2	L1_M2P2_AUGER
L_1	M_2 & P_3	L1_M2P3_AUGER
L_1	M_2 & P_4	L1_M2P4_AUGER
L_1	M_2 & P_5	L1_M2P5_AUGER
L_1	M_2 & Q_1	L1_M2Q1_AUGER
L_1	M_2 & Q_2	L1_M2Q2_AUGER
L_1	M_2 & Q_3	L1_M2Q3_AUGER

Excited shell	Ionized shells	Auger macro
L_1	M_3 & L_2	`L1_M3L2_AUGER`
L_1	M_3 & L_3	`L1_M3L3_AUGER`
L_1	M_3 & M_1	`L1_M3M1_AUGER`
L_1	M_3 & M_2	`L1_M3M2_AUGER`
L_1	M_3 & M_3	`L1_M3M3_AUGER`
L_1	M_3 & M_4	`L1_M3M4_AUGER`
L_1	M_3 & M_5	`L1_M3M5_AUGER`
L_1	M_3 & N_1	`L1_M3N1_AUGER`
L_1	M_3 & N_2	`L1_M3N2_AUGER`
L_1	M_3 & N_3	`L1_M3N3_AUGER`
L_1	M_3 & N_4	`L1_M3N4_AUGER`
L_1	M_3 & N_5	`L1_M3N5_AUGER`
L_1	M_3 & N_6	`L1_M3N6_AUGER`
L_1	M_3 & N_7	`L1_M3N7_AUGER`
L_1	M_3 & O_1	`L1_M3O1_AUGER`
L_1	M_3 & O_2	`L1_M3O2_AUGER`
L_1	M_3 & O_3	`L1_M3O3_AUGER`
L_1	M_3 & O_4	`L1_M3O4_AUGER`
L_1	M_3 & O_5	`L1_M3O5_AUGER`
L_1	M_3 & O_6	`L1_M3O6_AUGER`
L_1	M_3 & O_7	`L1_M3O7_AUGER`
L_1	M_3 & P_1	`L1_M3P1_AUGER`
L_1	M_3 & P_2	`L1_M3P2_AUGER`
L_1	M_3 & P_3	`L1_M3P3_AUGER`
L_1	M_3 & P_4	`L1_M3P4_AUGER`
L_1	M_3 & P_5	`L1_M3P5_AUGER`
L_1	M_3 & Q_1	`L1_M3Q1_AUGER`
L_1	M_3 & Q_2	`L1_M3Q2_AUGER`
L_1	M_3 & Q_3	`L1_M3Q3_AUGER`
L_1	M_4 & L_2	`L1_M4L2_AUGER`
L_1	M_4 & L_3	`L1_M4L3_AUGER`
L_1	M_4 & M_1	`L1_M4M1_AUGER`
L_1	M_4 & M_2	`L1_M4M2_AUGER`
L_1	M_4 & M_3	`L1_M4M3_AUGER`
L_1	M_4 & M_4	`L1_M4M4_AUGER`
L_1	M_4 & M_5	`L1_M4M5_AUGER`
L_1	M_4 & N_1	`L1_M4N1_AUGER`
L_1	M_4 & N_2	`L1_M4N2_AUGER`
L_1	M_4 & N_3	`L1_M4N3_AUGER`
L_1	M_4 & N_4	`L1_M4N4_AUGER`
L_1	M_4 & N_5	`L1_M4N5_AUGER`
L_1	M_4 & N_6	`L1_M4N6_AUGER`

Excited shell	Ionized shells	Auger macro
L_1	M_4 & N_7	`L1_M4N7_AUGER`
L_1	M_4 & O_1	`L1_M4O1_AUGER`
L_1	M_4 & O_2	`L1_M4O2_AUGER`
L_1	M_4 & O_3	`L1_M4O3_AUGER`
L_1	M_4 & O_4	`L1_M4O4_AUGER`
L_1	M_4 & O_5	`L1_M4O5_AUGER`
L_1	M_4 & O_6	`L1_M4O6_AUGER`
L_1	M_4 & O_7	`L1_M4O7_AUGER`
L_1	M_4 & P_1	`L1_M4P1_AUGER`
L_1	M_4 & P_2	`L1_M4P2_AUGER`
L_1	M_4 & P_3	`L1_M4P3_AUGER`
L_1	M_4 & P_4	`L1_M4P4_AUGER`
L_1	M_4 & P_5	`L1_M4P5_AUGER`
L_1	M_4 & Q_1	`L1_M4Q1_AUGER`
L_1	M_4 & Q_2	`L1_M4Q2_AUGER`
L_1	M_4 & Q_3	`L1_M4Q3_AUGER`
L_1	M_5 & L_2	`L1_M5L2_AUGER`
L_1	M_5 & L_3	`L1_M5L3_AUGER`
L_1	M_5 & M_1	`L1_M5M1_AUGER`
L_1	M_5 & M_2	`L1_M5M2_AUGER`
L_1	M_5 & M_3	`L1_M5M3_AUGER`
L_1	M_5 & M_4	`L1_M5M4_AUGER`
L_1	M_5 & M_5	`L1_M5M5_AUGER`
L_1	M_5 & N_1	`L1_M5N1_AUGER`
L_1	M_5 & N_2	`L1_M5N2_AUGER`
L_1	M_5 & N_3	`L1_M5N3_AUGER`
L_1	M_5 & N_4	`L1_M5N4_AUGER`
L_1	M_5 & N_5	`L1_M5N5_AUGER`
L_1	M_5 & N_6	`L1_M5N6_AUGER`
L_1	M_5 & N_7	`L1_M5N7_AUGER`
L_1	M_5 & O_1	`L1_M5O1_AUGER`
L_1	M_5 & O_2	`L1_M5O2_AUGER`
L_1	M_5 & O_3	`L1_M5O3_AUGER`
L_1	M_5 & O_4	`L1_M5O4_AUGER`
L_1	M_5 & O_5	`L1_M5O5_AUGER`
L_1	M_5 & O_6	`L1_M5O6_AUGER`
L_1	M_5 & O_7	`L1_M5O7_AUGER`
L_1	M_5 & P_1	`L1_M5P1_AUGER`
L_1	M_5 & P_2	`L1_M5P2_AUGER`
L_1	M_5 & P_3	`L1_M5P3_AUGER`
L_1	M_5 & P_4	`L1_M5P4_AUGER`
L_1	M_5 & P_5	`L1_M5P5_AUGER`

Excited shell	Ionized shells	Auger macro
L_1	M_5 & Q_1	L1_M5Q1_AUGER
L_1	M_5 & Q_2	L1_M5Q2_AUGER
L_1	M_5 & Q_3	L1_M5Q3_AUGER
L_2	L_3 & L_3	L2_L3L3_AUGER
L_2	L_3 & M_1	L2_L3M1_AUGER
L_2	L_3 & M_2	L2_L3M2_AUGER
L_2	L_3 & M_3	L2_L3M3_AUGER
L_2	L_3 & M_4	L2_L3M4_AUGER
L_2	L_3 & M_5	L2_L3M5_AUGER
L_2	L_3 & N_1	L2_L3N1_AUGER
L_2	L_3 & N_2	L2_L3N2_AUGER
L_2	L_3 & N_3	L2_L3N3_AUGER
L_2	L_3 & N_4	L2_L3N4_AUGER
L_2	L_3 & N_5	L2_L3N5_AUGER
L_2	L_3 & N_6	L2_L3N6_AUGER
L_2	L_3 & N_7	L2_L3N7_AUGER
L_2	L_3 & O_1	L2_L3O1_AUGER
L_2	L_3 & O_2	L2_L3O2_AUGER
L_2	L_3 & O_3	L2_L3O3_AUGER
L_2	L_3 & O_4	L2_L3O4_AUGER
L_2	L_3 & O_5	L2_L3O5_AUGER
L_2	L_3 & O_6	L2_L3O6_AUGER
L_2	L_3 & O_7	L2_L3O7_AUGER
L_2	L_3 & P_1	L2_L3P1_AUGER
L_2	L_3 & P_2	L2_L3P2_AUGER
L_2	L_3 & P_3	L2_L3P3_AUGER
L_2	L_3 & P_4	L2_L3P4_AUGER
L_2	L_3 & P_5	L2_L3P5_AUGER
L_2	L_3 & Q_1	L2_L3Q1_AUGER
L_2	L_3 & Q_2	L2_L3Q2_AUGER
L_2	L_3 & Q_3	L2_L3Q3_AUGER
L_2	M_1 & M_1	L2_M1M1_AUGER
L_2	M_1 & M_2	L2_M1M2_AUGER
L_2	M_1 & M_3	L2_M1M3_AUGER
L_2	M_1 & M_4	L2_M1M4_AUGER
L_2	M_1 & M_5	L2_M1M5_AUGER
L_2	M_1 & N_1	L2_M1N1_AUGER
L_2	M_1 & N_2	L2_M1N2_AUGER
L_2	M_1 & N_3	L2_M1N3_AUGER
L_2	M_1 & N_4	L2_M1N4_AUGER
L_2	M_1 & N_5	L2_M1N5_AUGER
L_2	M_1 & N_6	L2_M1N6_AUGER

Excited shell	Ionized shells	Auger macro
L_2	M_1 & N_7	L2_M1N7_AUGER
L_2	M_1 & O_1	L2_M1O1_AUGER
L_2	M_1 & O_2	L2_M1O2_AUGER
L_2	M_1 & O_3	L2_M1O3_AUGER
L_2	M_1 & O_4	L2_M1O4_AUGER
L_2	M_1 & O_5	L2_M1O5_AUGER
L_2	M_1 & O_6	L2_M1O6_AUGER
L_2	M_1 & O_7	L2_M1O7_AUGER
L_2	M_1 & P_1	L2_M1P1_AUGER
L_2	M_1 & P_2	L2_M1P2_AUGER
L_2	M_1 & P_3	L2_M1P3_AUGER
L_2	M_1 & P_4	L2_M1P4_AUGER
L_2	M_1 & P_5	L2_M1P5_AUGER
L_2	M_1 & Q_1	L2_M1Q1_AUGER
L_2	M_1 & Q_2	L2_M1Q2_AUGER
L_2	M_1 & Q_3	L2_M1Q3_AUGER
L_2	M_2 & L_3	L2_M2L3_AUGER
L_2	M_2 & M_1	L2_M2M1_AUGER
L_2	M_2 & M_2	L2_M2M2_AUGER
L_2	M_2 & M_3	L2_M2M3_AUGER
L_2	M_2 & M_4	L2_M2M4_AUGER
L_2	M_2 & M_5	L2_M2M5_AUGER
L_2	M_2 & N_1	L2_M2N1_AUGER
L_2	M_2 & N_2	L2_M2N2_AUGER
L_2	M_2 & N_3	L2_M2N3_AUGER
L_2	M_2 & N_4	L2_M2N4_AUGER
L_2	M_2 & N_5	L2_M2N5_AUGER
L_2	M_2 & N_6	L2_M2N6_AUGER
L_2	M_2 & N_7	L2_M2N7_AUGER
L_2	M_2 & O_1	L2_M2O1_AUGER
L_2	M_2 & O_2	L2_M2O2_AUGER
L_2	M_2 & O_3	L2_M2O3_AUGER
L_2	M_2 & O_4	L2_M2O4_AUGER
L_2	M_2 & O_5	L2_M2O5_AUGER
L_2	M_2 & O_6	L2_M2O6_AUGER
L_2	M_2 & O_7	L2_M2O7_AUGER
L_2	M_2 & P_1	L2_M2P1_AUGER
L_2	M_2 & P_2	L2_M2P2_AUGER
L_2	M_2 & P_3	L2_M2P3_AUGER
L_2	M_2 & P_4	L2_M2P4_AUGER
L_2	M_2 & P_5	L2_M2P5_AUGER
L_2	M_2 & Q_1	L2_M2Q1_AUGER

Excited shell	Ionized shells	Auger macro
L_2	M_2 & Q_2	L2_M2Q2_AUGER
L_2	M_2 & Q_3	L2_M2Q3_AUGER
L_2	M_3 & L_3	L2_M3L3_AUGER
L_2	M_3 & M_1	L2_M3M1_AUGER
L_2	M_3 & M_2	L2_M3M2_AUGER
L_2	M_3 & M_3	L2_M3M3_AUGER
L_2	M_3 & M_4	L2_M3M4_AUGER
L_2	M_3 & M_5	L2_M3M5_AUGER
L_2	M_3 & N_1	L2_M3N1_AUGER
L_2	M_3 & N_2	L2_M3N2_AUGER
L_2	M_3 & N_3	L2_M3N3_AUGER
L_2	M_3 & N_4	L2_M3N4_AUGER
L_2	M_3 & N_5	L2_M3N5_AUGER
L_2	M_3 & N_6	L2_M3N6_AUGER
L_2	M_3 & N_7	L2_M3N7_AUGER
L_2	M_3 & O_1	L2_M3O1_AUGER
L_2	M_3 & O_2	L2_M3O2_AUGER
L_2	M_3 & O_3	L2_M3O3_AUGER
L_2	M_3 & O_4	L2_M3O4_AUGER
L_2	M_3 & O_5	L2_M3O5_AUGER
L_2	M_3 & O_6	L2_M3O6_AUGER
L_2	M_3 & O_7	L2_M3O7_AUGER
L_2	M_3 & P_1	L2_M3P1_AUGER
L_2	M_3 & P_2	L2_M3P2_AUGER
L_2	M_3 & P_3	L2_M3P3_AUGER
L_2	M_3 & P_4	L2_M3P4_AUGER
L_2	M_3 & P_5	L2_M3P5_AUGER
L_2	M_3 & Q_1	L2_M3Q1_AUGER
L_2	M_3 & Q_2	L2_M3Q2_AUGER
L_2	M_3 & Q_3	L2_M3Q3_AUGER
L_2	M_4 & L_3	L2_M4L3_AUGER
L_2	M_4 & M_1	L2_M4M1_AUGER
L_2	M_4 & M_2	L2_M4M2_AUGER
L_2	M_4 & M_3	L2_M4M3_AUGER
L_2	M_4 & M_4	L2_M4M4_AUGER
L_2	M_4 & M_5	L2_M4M5_AUGER
L_2	M_4 & N_1	L2_M4N1_AUGER
L_2	M_4 & N_2	L2_M4N2_AUGER
L_2	M_4 & N_3	L2_M4N3_AUGER
L_2	M_4 & N_4	L2_M4N4_AUGER
L_2	M_4 & N_5	L2_M4N5_AUGER
L_2	M_4 & N_6	L2_M4N6_AUGER

Excited shell	Ionized shells	Auger macro
L_2	M_4 & N_7	L2_M4N7_AUGER
L_2	M_4 & O_1	L2_M4O1_AUGER
L_2	M_4 & O_2	L2_M4O2_AUGER
L_2	M_4 & O_3	L2_M4O3_AUGER
L_2	M_4 & O_4	L2_M4O4_AUGER
L_2	M_4 & O_5	L2_M4O5_AUGER
L_2	M_4 & O_6	L2_M4O6_AUGER
L_2	M_4 & O_7	L2_M4O7_AUGER
L_2	M_4 & P_1	L2_M4P1_AUGER
L_2	M_4 & P_2	L2_M4P2_AUGER
L_2	M_4 & P_3	L2_M4P3_AUGER
L_2	M_4 & P_4	L2_M4P4_AUGER
L_2	M_4 & P_5	L2_M4P5_AUGER
L_2	M_4 & Q_1	L2_M4Q1_AUGER
L_2	M_4 & Q_2	L2_M4Q2_AUGER
L_2	M_4 & Q_3	L2_M4Q3_AUGER
L_2	M_5 & L_3	L2_M5L3_AUGER
L_2	M_5 & M_1	L2_M5M1_AUGER
L_2	M_5 & M_2	L2_M5M2_AUGER
L_2	M_5 & M_3	L2_M5M3_AUGER
L_2	M_5 & M_4	L2_M5M4_AUGER
L_2	M_5 & M_5	L2_M5M5_AUGER
L_2	M_5 & N_1	L2_M5N1_AUGER
L_2	M_5 & N_2	L2_M5N2_AUGER
L_2	M_5 & N_3	L2_M5N3_AUGER
L_2	M_5 & N_4	L2_M5N4_AUGER
L_2	M_5 & N_5	L2_M5N5_AUGER
L_2	M_5 & N_6	L2_M5N6_AUGER
L_2	M_5 & N_7	L2_M5N7_AUGER
L_2	M_5 & O_1	L2_M5O1_AUGER
L_2	M_5 & O_2	L2_M5O2_AUGER
L_2	M_5 & O_3	L2_M5O3_AUGER
L_2	M_5 & O_4	L2_M5O4_AUGER
L_2	M_5 & O_5	L2_M5O5_AUGER
L_2	M_5 & O_6	L2_M5O6_AUGER
L_2	M_5 & O_7	L2_M5O7_AUGER
L_2	M_5 & P_1	L2_M5P1_AUGER
L_2	M_5 & P_2	L2_M5P2_AUGER
L_2	M_5 & P_3	L2_M5P3_AUGER
L_2	M_5 & P_4	L2_M5P4_AUGER
L_2	M_5 & P_5	L2_M5P5_AUGER
L_2	M_5 & Q_1	L2_M5Q1_AUGER

Excited shell	Ionized shells	Auger macro
L_2	M_5 & Q_2	L2_M5Q2_AUGER
L_2	M_5 & Q_3	L2_M5Q3_AUGER
L_3	M_1 & M_1	L3_M1M1_AUGER
L_3	M_1 & M_2	L3_M1M2_AUGER
L_3	M_1 & M_3	L3_M1M3_AUGER
L_3	M_1 & M_4	L3_M1M4_AUGER
L_3	M_1 & M_5	L3_M1M5_AUGER
L_3	M_1 & N_1	L3_M1N1_AUGER
L_3	M_1 & N_2	L3_M1N2_AUGER
L_3	M_1 & N_3	L3_M1N3_AUGER
L_3	M_1 & N_4	L3_M1N4_AUGER
L_3	M_1 & N_5	L3_M1N5_AUGER
L_3	M_1 & N_6	L3_M1N6_AUGER
L_3	M_1 & N_7	L3_M1N7_AUGER
L_3	M_1 & O_1	L3_M1O1_AUGER
L_3	M_1 & O_2	L3_M1O2_AUGER
L_3	M_1 & O_3	L3_M1O3_AUGER
L_3	M_1 & O_4	L3_M1O4_AUGER
L_3	M_1 & O_5	L3_M1O5_AUGER
L_3	M_1 & O_6	L3_M1O6_AUGER
L_3	M_1 & O_7	L3_M1O7_AUGER
L_3	M_1 & P_1	L3_M1P1_AUGER
L_3	M_1 & P_2	L3_M1P2_AUGER
L_3	M_1 & P_3	L3_M1P3_AUGER
L_3	M_1 & P_4	L3_M1P4_AUGER
L_3	M_1 & P_5	L3_M1P5_AUGER
L_3	M_1 & Q_1	L3_M1Q1_AUGER
L_3	M_1 & Q_2	L3_M1Q2_AUGER
L_3	M_1 & Q_3	L3_M1Q3_AUGER
L_3	M_2 & M_1	L3_M2M1_AUGER
L_3	M_2 & M_2	L3_M2M2_AUGER
L_3	M_2 & M_3	L3_M2M3_AUGER
L_3	M_2 & M_4	L3_M2M4_AUGER
L_3	M_2 & M_5	L3_M2M5_AUGER
L_3	M_2 & N_1	L3_M2N1_AUGER
L_3	M_2 & N_2	L3_M2N2_AUGER
L_3	M_2 & N_3	L3_M2N3_AUGER
L_3	M_2 & N_4	L3_M2N4_AUGER
L_3	M_2 & N_5	L3_M2N5_AUGER
L_3	M_2 & N_6	L3_M2N6_AUGER
L_3	M_2 & N_7	L3_M2N7_AUGER
L_3	M_2 & O_1	L3_M2O1_AUGER

Excited shell	Ionized shells	Auger macro
L_3	M_2 & O_2	L3_M2O2_AUGER
L_3	M_2 & O_3	L3_M2O3_AUGER
L_3	M_2 & O_4	L3_M2O4_AUGER
L_3	M_2 & O_5	L3_M2O5_AUGER
L_3	M_2 & O_6	L3_M2O6_AUGER
L_3	M_2 & O_7	L3_M2O7_AUGER
L_3	M_2 & P_1	L3_M2P1_AUGER
L_3	M_2 & P_2	L3_M2P2_AUGER
L_3	M_2 & P_3	L3_M2P3_AUGER
L_3	M_2 & P_4	L3_M2P4_AUGER
L_3	M_2 & P_5	L3_M2P5_AUGER
L_3	M_2 & Q_1	L3_M2Q1_AUGER
L_3	M_2 & Q_2	L3_M2Q2_AUGER
L_3	M_2 & Q_3	L3_M2Q3_AUGER
L_3	M_3 & M_1	L3_M3M1_AUGER
L_3	M_3 & M_2	L3_M3M2_AUGER
L_3	M_3 & M_3	L3_M3M3_AUGER
L_3	M_3 & M_4	L3_M3M4_AUGER
L_3	M_3 & M_5	L3_M3M5_AUGER
L_3	M_3 & N_1	L3_M3N1_AUGER
L_3	M_3 & N_2	L3_M3N2_AUGER
L_3	M_3 & N_3	L3_M3N3_AUGER
L_3	M_3 & N_4	L3_M3N4_AUGER
L_3	M_3 & N_5	L3_M3N5_AUGER
L_3	M_3 & N_6	L3_M3N6_AUGER
L_3	M_3 & N_7	L3_M3N7_AUGER
L_3	M_3 & O_1	L3_M3O1_AUGER
L_3	M_3 & O_2	L3_M3O2_AUGER
L_3	M_3 & O_3	L3_M3O3_AUGER
L_3	M_3 & O_4	L3_M3O4_AUGER
L_3	M_3 & O_5	L3_M3O5_AUGER
L_3	M_3 & O_6	L3_M3O6_AUGER
L_3	M_3 & O_7	L3_M3O7_AUGER
L_3	M_3 & P_1	L3_M3P1_AUGER
L_3	M_3 & P_2	L3_M3P2_AUGER
L_3	M_3 & P_3	L3_M3P3_AUGER
L_3	M_3 & P_4	L3_M3P4_AUGER
L_3	M_3 & P_5	L3_M3P5_AUGER
L_3	M_3 & Q_1	L3_M3Q1_AUGER
L_3	M_3 & Q_2	L3_M3Q2_AUGER
L_3	M_3 & Q_3	L3_M3Q3_AUGER
L_3	M_4 & M_1	L3_M4M1_AUGER

Excited shell	Ionized shells	Auger macro
L_3	M_4 & M_2	L3_M4M2_AUGER
L_3	M_4 & M_3	L3_M4M3_AUGER
L_3	M_4 & M_4	L3_M4M4_AUGER
L_3	M_4 & M_5	L3_M4M5_AUGER
L_3	M_4 & N_1	L3_M4N1_AUGER
L_3	M_4 & N_2	L3_M4N2_AUGER
L_3	M_4 & N_3	L3_M4N3_AUGER
L_3	M_4 & N_4	L3_M4N4_AUGER
L_3	M_4 & N_5	L3_M4N5_AUGER
L_3	M_4 & N_6	L3_M4N6_AUGER
L_3	M_4 & N_7	L3_M4N7_AUGER
L_3	M_4 & O_1	L3_M4O1_AUGER
L_3	M_4 & O_2	L3_M4O2_AUGER
L_3	M_4 & O_3	L3_M4O3_AUGER
L_3	M_4 & O_4	L3_M4O4_AUGER
L_3	M_4 & O_5	L3_M4O5_AUGER
L_3	M_4 & O_6	L3_M4O6_AUGER
L_3	M_4 & O_7	L3_M4O7_AUGER
L_3	M_4 & P_1	L3_M4P1_AUGER
L_3	M_4 & P_2	L3_M4P2_AUGER
L_3	M_4 & P_3	L3_M4P3_AUGER
L_3	M_4 & P_4	L3_M4P4_AUGER
L_3	M_4 & P_5	L3_M4P5_AUGER
L_3	M_4 & Q_1	L3_M4Q1_AUGER
L_3	M_4 & Q_2	L3_M4Q2_AUGER
L_3	M_4 & Q_3	L3_M4Q3_AUGER
L_3	M_5 & M_1	L3_M5M1_AUGER
L_3	M_5 & M_2	L3_M5M2_AUGER
L_3	M_5 & M_3	L3_M5M3_AUGER
L_3	M_5 & M_4	L3_M5M4_AUGER
L_3	M_5 & M_5	L3_M5M5_AUGER
L_3	M_5 & N_1	L3_M5N1_AUGER
L_3	M_5 & N_2	L3_M5N2_AUGER
L_3	M_5 & N_3	L3_M5N3_AUGER
L_3	M_5 & N_4	L3_M5N4_AUGER
L_3	M_5 & N_5	L3_M5N5_AUGER
L_3	M_5 & N_6	L3_M5N6_AUGER
L_3	M_5 & N_7	L3_M5N7_AUGER
L_3	M_5 & O_1	L3_M5O1_AUGER
L_3	M_5 & O_2	L3_M5O2_AUGER
L_3	M_5 & O_3	L3_M5O3_AUGER
L_3	M_5 & O_4	L3_M5O4_AUGER

Excited shell	Ionized shells	Auger macro
L_3	M_5 & O_5	L3_M5O5_AUGER
L_3	M_5 & O_6	L3_M5O6_AUGER
L_3	M_5 & O_7	L3_M5O7_AUGER
L_3	M_5 & P_1	L3_M5P1_AUGER
L_3	M_5 & P_2	L3_M5P2_AUGER
L_3	M_5 & P_3	L3_M5P3_AUGER
L_3	M_5 & P_4	L3_M5P4_AUGER
L_3	M_5 & P_5	L3_M5P5_AUGER
L_3	M_5 & Q_1	L3_M5Q1_AUGER
L_3	M_5 & Q_2	L3_M5Q2_AUGER
L_3	M_5 & Q_3	L3_M5Q3_AUGER
M_1	M_2 & M_2	M1_M2M2_AUGER
M_1	M_2 & M_3	M1_M2M3_AUGER
M_1	M_2 & M_4	M1_M2M4_AUGER
M_1	M_2 & M_5	M1_M2M5_AUGER
M_1	M_2 & N_1	M1_M2N1_AUGER
M_1	M_2 & N_2	M1_M2N2_AUGER
M_1	M_2 & N_3	M1_M2N3_AUGER
M_1	M_2 & N_4	M1_M2N4_AUGER
M_1	M_2 & N_5	M1_M2N5_AUGER
M_1	M_2 & N_6	M1_M2N6_AUGER
M_1	M_2 & N_7	M1_M2N7_AUGER
M_1	M_2 & O_1	M1_M2O1_AUGER
M_1	M_2 & O_2	M1_M2O2_AUGER
M_1	M_2 & O_3	M1_M2O3_AUGER
M_1	M_2 & O_4	M1_M2O4_AUGER
M_1	M_2 & O_5	M1_M2O5_AUGER
M_1	M_2 & O_6	M1_M2O6_AUGER
M_1	M_2 & O_7	M1_M2O7_AUGER
M_1	M_2 & P_1	M1_M2P1_AUGER
M_1	M_2 & P_2	M1_M2P2_AUGER
M_1	M_2 & P_3	M1_M2P3_AUGER
M_1	M_2 & P_4	M1_M2P4_AUGER
M_1	M_2 & P_5	M1_M2P5_AUGER
M_1	M_2 & Q_1	M1_M2Q1_AUGER
M_1	M_2 & Q_2	M1_M2Q2_AUGER
M_1	M_2 & Q_3	M1_M2Q3_AUGER
M_1	M_3 & M_2	M1_M3M2_AUGER
M_1	M_3 & M_3	M1_M3M3_AUGER
M_1	M_3 & M_4	M1_M3M4_AUGER
M_1	M_3 & M_5	M1_M3M5_AUGER
M_1	M_3 & N_1	M1_M3N1_AUGER

Excited shell	Ionized shells	Auger macro
M_1	M_3 & N_2	M1_M3N2_AUGER
M_1	M_3 & N_3	M1_M3N3_AUGER
M_1	M_3 & N_4	M1_M3N4_AUGER
M_1	M_3 & N_5	M1_M3N5_AUGER
M_1	M_3 & N_6	M1_M3N6_AUGER
M_1	M_3 & N_7	M1_M3N7_AUGER
M_1	M_3 & O_1	M1_M3O1_AUGER
M_1	M_3 & O_2	M1_M3O2_AUGER
M_1	M_3 & O_3	M1_M3O3_AUGER
M_1	M_3 & O_4	M1_M3O4_AUGER
M_1	M_3 & O_5	M1_M3O5_AUGER
M_1	M_3 & O_6	M1_M3O6_AUGER
M_1	M_3 & O_7	M1_M3O7_AUGER
M_1	M_3 & P_1	M1_M3P1_AUGER
M_1	M_3 & P_2	M1_M3P2_AUGER
M_1	M_3 & P_3	M1_M3P3_AUGER
M_1	M_3 & P_4	M1_M3P4_AUGER
M_1	M_3 & P_5	M1_M3P5_AUGER
M_1	M_3 & Q_1	M1_M3Q1_AUGER
M_1	M_3 & Q_2	M1_M3Q2_AUGER
M_1	M_3 & Q_3	M1_M3Q3_AUGER
M_1	M_4 & M_2	M1_M4M2_AUGER
M_1	M_4 & M_3	M1_M4M3_AUGER
M_1	M_4 & M_4	M1_M4M4_AUGER
M_1	M_4 & M_5	M1_M4M5_AUGER
M_1	M_4 & N_1	M1_M4N1_AUGER
M_1	M_4 & N_2	M1_M4N2_AUGER
M_1	M_4 & N_3	M1_M4N3_AUGER
M_1	M_4 & N_4	M1_M4N4_AUGER
M_1	M_4 & N_5	M1_M4N5_AUGER
M_1	M_4 & N_6	M1_M4N6_AUGER
M_1	M_4 & N_7	M1_M4N7_AUGER
M_1	M_4 & O_1	M1_M4O1_AUGER
M_1	M_4 & O_2	M1_M4O2_AUGER
M_1	M_4 & O_3	M1_M4O3_AUGER
M_1	M_4 & O_4	M1_M4O4_AUGER
M_1	M_4 & O_5	M1_M4O5_AUGER
M_1	M_4 & O_6	M1_M4O6_AUGER
M_1	M_4 & O_7	M1_M4O7_AUGER
M_1	M_4 & P_1	M1_M4P1_AUGER
M_1	M_4 & P_2	M1_M4P2_AUGER
M_1	M_4 & P_3	M1_M4P3_AUGER

Excited shell	Ionized shells	Auger macro
M_1	M_4 & P_4	M1_M4P4_AUGER
M_1	M_4 & P_5	M1_M4P5_AUGER
M_1	M_4 & Q_1	M1_M4Q1_AUGER
M_1	M_4 & Q_2	M1_M4Q2_AUGER
M_1	M_4 & Q_3	M1_M4Q3_AUGER
M_1	M_5 & M_2	M1_M5M2_AUGER
M_1	M_5 & M_3	M1_M5M3_AUGER
M_1	M_5 & M_4	M1_M5M4_AUGER
M_1	M_5 & M_5	M1_M5M5_AUGER
M_1	M_5 & N_1	M1_M5N1_AUGER
M_1	M_5 & N_2	M1_M5N2_AUGER
M_1	M_5 & N_3	M1_M5N3_AUGER
M_1	M_5 & N_4	M1_M5N4_AUGER
M_1	M_5 & N_5	M1_M5N5_AUGER
M_1	M_5 & N_6	M1_M5N6_AUGER
M_1	M_5 & N_7	M1_M5N7_AUGER
M_1	M_5 & O_1	M1_M5O1_AUGER
M_1	M_5 & O_2	M1_M5O2_AUGER
M_1	M_5 & O_3	M1_M5O3_AUGER
M_1	M_5 & O_4	M1_M5O4_AUGER
M_1	M_5 & O_5	M1_M5O5_AUGER
M_1	M_5 & O_6	M1_M5O6_AUGER
M_1	M_5 & O_7	M1_M5O7_AUGER
M_1	M_5 & P_1	M1_M5P1_AUGER
M_1	M_5 & P_2	M1_M5P2_AUGER
M_1	M_5 & P_3	M1_M5P3_AUGER
M_1	M_5 & P_4	M1_M5P4_AUGER
M_1	M_5 & P_5	M1_M5P5_AUGER
M_1	M_5 & Q_1	M1_M5Q1_AUGER
M_1	M_5 & Q_2	M1_M5Q2_AUGER
M_1	M_5 & Q_3	M1_M5Q3_AUGER
M_2	M_3 & M_3	M2_M3M3_AUGER
M_2	M_3 & M_4	M2_M3M4_AUGER
M_2	M_3 & M_5	M2_M3M5_AUGER
M_2	M_3 & N_1	M2_M3N1_AUGER
M_2	M_3 & N_2	M2_M3N2_AUGER
M_2	M_3 & N_3	M2_M3N3_AUGER
M_2	M_3 & N_4	M2_M3N4_AUGER
M_2	M_3 & N_5	M2_M3N5_AUGER
M_2	M_3 & N_6	M2_M3N6_AUGER
M_2	M_3 & N_7	M2_M3N7_AUGER
M_2	M_3 & O_1	M2_M3O1_AUGER

Excited shell	Ionized shells	Auger macro
M_2	M_3 & O_2	M2_M3O2_AUGER
M_2	M_3 & O_3	M2_M3O3_AUGER
M_2	M_3 & O_4	M2_M3O4_AUGER
M_2	M_3 & O_5	M2_M3O5_AUGER
M_2	M_3 & O_6	M2_M3O6_AUGER
M_2	M_3 & O_7	M2_M3O7_AUGER
M_2	M_3 & P_1	M2_M3P1_AUGER
M_2	M_3 & P_2	M2_M3P2_AUGER
M_2	M_3 & P_3	M2_M3P3_AUGER
M_2	M_3 & P_4	M2_M3P4_AUGER
M_2	M_3 & P_5	M2_M3P5_AUGER
M_2	M_3 & Q_1	M2_M3Q1_AUGER
M_2	M_3 & Q_2	M2_M3Q2_AUGER
M_2	M_3 & Q_3	M2_M3Q3_AUGER
M_2	M_4 & M_3	M2_M4M3_AUGER
M_2	M_4 & M_4	M2_M4M4_AUGER
M_2	M_4 & M_5	M2_M4M5_AUGER
M_2	M_4 & N_1	M2_M4N1_AUGER
M_2	M_4 & N_2	M2_M4N2_AUGER
M_2	M_4 & N_3	M2_M4N3_AUGER
M_2	M_4 & N_4	M2_M4N4_AUGER
M_2	M_4 & N_5	M2_M4N5_AUGER
M_2	M_4 & N_6	M2_M4N6_AUGER
M_2	M_4 & N_7	M2_M4N7_AUGER
M_2	M_4 & O_1	M2_M4O1_AUGER
M_2	M_4 & O_2	M2_M4O2_AUGER
M_2	M_4 & O_3	M2_M4O3_AUGER
M_2	M_4 & O_4	M2_M4O4_AUGER
M_2	M_4 & O_5	M2_M4O5_AUGER
M_2	M_4 & O_6	M2_M4O6_AUGER
M_2	M_4 & O_7	M2_M4O7_AUGER
M_2	M_4 & P_1	M2_M4P1_AUGER
M_2	M_4 & P_2	M2_M4P2_AUGER
M_2	M_4 & P_3	M2_M4P3_AUGER
M_2	M_4 & P_4	M2_M4P4_AUGER
M_2	M_4 & P_5	M2_M4P5_AUGER
M_2	M_4 & Q_1	M2_M4Q1_AUGER
M_2	M_4 & Q_2	M2_M4Q2_AUGER
M_2	M_4 & Q_3	M2_M4Q3_AUGER
M_2	M_5 & M_3	M2_M5M3_AUGER
M_2	M_5 & M_4	M2_M5M4_AUGER
M_2	M_5 & M_5	M2_M5M5_AUGER

Excited shell	Ionized shells	Auger macro
M_2	M_5 & N_1	M2_M5N1_AUGER
M_2	M_5 & N_2	M2_M5N2_AUGER
M_2	M_5 & N_3	M2_M5N3_AUGER
M_2	M_5 & N_4	M2_M5N4_AUGER
M_2	M_5 & N_5	M2_M5N5_AUGER
M_2	M_5 & N_6	M2_M5N6_AUGER
M_2	M_5 & N_7	M2_M5N7_AUGER
M_2	M_5 & O_1	M2_M5O1_AUGER
M_2	M_5 & O_2	M2_M5O2_AUGER
M_2	M_5 & O_3	M2_M5O3_AUGER
M_2	M_5 & O_4	M2_M5O4_AUGER
M_2	M_5 & O_5	M2_M5O5_AUGER
M_2	M_5 & O_6	M2_M5O6_AUGER
M_2	M_5 & O_7	M2_M5O7_AUGER
M_2	M_5 & P_1	M2_M5P1_AUGER
M_2	M_5 & P_2	M2_M5P2_AUGER
M_2	M_5 & P_3	M2_M5P3_AUGER
M_2	M_5 & P_4	M2_M5P4_AUGER
M_2	M_5 & P_5	M2_M5P5_AUGER
M_2	M_5 & Q_1	M2_M5Q1_AUGER
M_2	M_5 & Q_2	M2_M5Q2_AUGER
M_2	M_5 & Q_3	M2_M5Q3_AUGER
M_3	M_4 & M_4	M3_M4M4_AUGER
M_3	M_4 & M_5	M3_M4M5_AUGER
M_3	M_4 & N_1	M3_M4N1_AUGER
M_3	M_4 & N_2	M3_M4N2_AUGER
M_3	M_4 & N_3	M3_M4N3_AUGER
M_3	M_4 & N_4	M3_M4N4_AUGER
M_3	M_4 & N_5	M3_M4N5_AUGER
M_3	M_4 & N_6	M3_M4N6_AUGER
M_3	M_4 & N_7	M3_M4N7_AUGER
M_3	M_4 & O_1	M3_M4O1_AUGER
M_3	M_4 & O_2	M3_M4O2_AUGER
M_3	M_4 & O_3	M3_M4O3_AUGER
M_3	M_4 & O_4	M3_M4O4_AUGER
M_3	M_4 & O_5	M3_M4O5_AUGER
M_3	M_4 & O_6	M3_M4O6_AUGER
M_3	M_4 & O_7	M3_M4O7_AUGER
M_3	M_4 & P_1	M3_M4P1_AUGER
M_3	M_4 & P_2	M3_M4P2_AUGER
M_3	M_4 & P_3	M3_M4P3_AUGER
M_3	M_4 & P_4	M3_M4P4_AUGER

Excited shell	Ionized shells	Auger macro
M_3	M_4 & P_5	M3_M4P5_AUGER
M_3	M_4 & Q_1	M3_M4Q1_AUGER
M_3	M_4 & Q_2	M3_M4Q2_AUGER
M_3	M_4 & Q_3	M3_M4Q3_AUGER
M_3	M_5 & M_4	M3_M5M4_AUGER
M_3	M_5 & M_5	M3_M5M5_AUGER
M_3	M_5 & N_1	M3_M5N1_AUGER
M_3	M_5 & N_2	M3_M5N2_AUGER
M_3	M_5 & N_3	M3_M5N3_AUGER
M_3	M_5 & N_4	M3_M5N4_AUGER
M_3	M_5 & N_5	M3_M5N5_AUGER
M_3	M_5 & N_6	M3_M5N6_AUGER
M_3	M_5 & N_7	M3_M5N7_AUGER
M_3	M_5 & O_1	M3_M5O1_AUGER
M_3	M_5 & O_2	M3_M5O2_AUGER
M_3	M_5 & O_3	M3_M5O3_AUGER
M_3	M_5 & O_4	M3_M5O4_AUGER
M_3	M_5 & O_5	M3_M5O5_AUGER
M_3	M_5 & O_6	M3_M5O6_AUGER
M_3	M_5 & O_7	M3_M5O7_AUGER
M_3	M_5 & P_1	M3_M5P1_AUGER
M_3	M_5 & P_2	M3_M5P2_AUGER
M_3	M_5 & P_3	M3_M5P3_AUGER
M_3	M_5 & P_4	M3_M5P4_AUGER
M_3	M_5 & P_5	M3_M5P5_AUGER
M_3	M_5 & Q_1	M3_M5Q1_AUGER
M_3	M_5 & Q_2	M3_M5Q2_AUGER
M_3	M_5 & Q_3	M3_M5Q3_AUGER
M_4	M_5 & M_5	M4_M5M5_AUGER
M_4	M_5 & N_1	M4_M5N1_AUGER
M_4	M_5 & N_2	M4_M5N2_AUGER
M_4	M_5 & N_3	M4_M5N3_AUGER
M_4	M_5 & N_4	M4_M5N4_AUGER
M_4	M_5 & N_5	M4_M5N5_AUGER
M_4	M_5 & N_6	M4_M5N6_AUGER
M_4	M_5 & N_7	M4_M5N7_AUGER
M_4	M_5 & O_1	M4_M5O1_AUGER
M_4	M_5 & O_2	M4_M5O2_AUGER
M_4	M_5 & O_3	M4_M5O3_AUGER
M_4	M_5 & O_4	M4_M5O4_AUGER
M_4	M_5 & O_5	M4_M5O5_AUGER
M_4	M_5 & O_6	M4_M5O6_AUGER

Excited shell	Ionized shells	Auger macro
M_4	M_5 & O_7	M4_M5O7_AUGER
M_4	M_5 & P_1	M4_M5P1_AUGER
M_4	M_5 & P_2	M4_M5P2_AUGER
M_4	M_5 & P_3	M4_M5P3_AUGER
M_4	M_5 & P_4	M4_M5P4_AUGER
M_4	M_5 & P_5	M4_M5P5_AUGER
M_4	M_5 & Q_1	M4_M5Q1_AUGER
M_4	M_5 & Q_2	M4_M5Q2_AUGER
M_4	M_5 & Q_3	M4_M5Q3_AUGER

A.6 NIST compound catalog macros

The functions `GetCompoundDataNISTByIndex` and `GetCompoundDataNISTByName` provide access to the compositions of a large number of commonly used compounds, obtained from the NIST website.
The following table has two columns: the first for the macros that are intended to be used as argument to `GetCompoundDataNISTByIndex`, while the second contains the names that should be provided to the `GetCompoundDataNISTByName` function, as well as the cross section functions with the `_CP` suffix. The list of names in the second column is identical to the array of strings that can be queried using `GetCompoundDataNISTList`.

NIST compound catalog macro	NIST compound catalog name
NIST_COMPOUND_A_150_TISSUE_EQUIVALENT_PLASTIC	A-150 Tissue-Equivalent Plastic
NIST_COMPOUND_ACETONE	Acetone
NIST_COMPOUND_ACETYLENE	Acetylene
NIST_COMPOUND_ADENINE	Adenine
NIST_COMPOUND_ADIPOSE_TISSUE_ICRP	Adipose Tissue (ICRP)
NIST_COMPOUND_AIR_DRY_NEAR_SEA_LEVEL	Air, Dry (near sea level)
NIST_COMPOUND_ALANINE	Alanine
NIST_COMPOUND_ALUMINUM_OXIDE	Aluminum Oxide
NIST_COMPOUND_AMBER	Amber
NIST_COMPOUND_AMMONIA	Ammonia
NIST_COMPOUND_ANILINE	Aniline
NIST_COMPOUND_ANTHRACENE	Anthracene
NIST_COMPOUND_B_100_BONE_EQUIVALENT_PLASTIC	B-100 Bone-Equivalent Plastic
NIST_COMPOUND_BAKELITE	Bakelite
NIST_COMPOUND_BARIUM_FLUORIDE	Barium Fluoride
NIST_COMPOUND_BARIUM_SULFATE	Barium Sulfate
NIST_COMPOUND_BENZENE	Benzene
NIST_COMPOUND_BERYLLIUM_OXIDE	Beryllium oxide

NIST compound catalog macro	NIST compound catalog name
NIST_COMPOUND_BISMUTH_GERMANIUM_OXIDE	Bismuth Germanium oxide
NIST_COMPOUND_BLOOD_ICRP	Blood (ICRP)
NIST_COMPOUND_BONE_COMPACT_ICRU	Bone, Compact (ICRU)
NIST_COMPOUND_BONE_CORTICAL_ICRP	Bone, Cortical (ICRP)
NIST_COMPOUND_BORON_CARBIDE	Boron Carbide
NIST_COMPOUND_BORON_OXIDE	Boron Oxide
NIST_COMPOUND_BRAIN_ICRP	Brain (ICRP)
NIST_COMPOUND_BUTANE	Butane
NIST_COMPOUND_N_BUTYL_ALCOHOL	N-Butyl Alcohol
NIST_COMPOUND_C_552_AIR_EQUIVALENT_PLASTIC	C-552 Air-Equivalent Plastic
NIST_COMPOUND_CADMIUM_TELLURIDE	Cadmium Telluride
NIST_COMPOUND_CADMIUM_TUNGSTATE	Cadmium Tungstate
NIST_COMPOUND_CALCIUM_CARBONATE	Calcium Carbonate
NIST_COMPOUND_CALCIUM_FLUORIDE	Calcium Fluoride
NIST_COMPOUND_CALCIUM_OXIDE	Calcium Oxide
NIST_COMPOUND_CALCIUM_SULFATE	Calcium Sulfate
NIST_COMPOUND_CALCIUM_TUNGSTATE	Calcium Tungstate
NIST_COMPOUND_CARBON_DIOXIDE	Carbon Dioxide
NIST_COMPOUND_CARBON_TETRACHLORIDE	Carbon Tetrachloride
NIST_COMPOUND_CELLULOSE_ACETATE_CELLOPHANE	Cellulose Acetate, Cellophane
NIST_COMPOUND_CELLULOSE_ACETATE_BUTYRATE	Cellulose Acetate Butyrate
NIST_COMPOUND_CELLULOSE_NITRATE	Cellulose Nitrate
NIST_COMPOUND_CERIC_SULFATE_DOSIMETER_SOLUTION	Ceric Sulfate Dosimeter Solution
NIST_COMPOUND_CESIUM_FLUORIDE	Cesium Fluoride
NIST_COMPOUND_CESIUM_IODIDE	Cesium Iodide
NIST_COMPOUND_CHLOROBENZENE	Chlorobenzene
NIST_COMPOUND_CHLOROFORM	Chloroform
NIST_COMPOUND_CONCRETE_PORTLAND	Concrete, Portland
NIST_COMPOUND_CYCLOHEXANE	Cyclohexane
NIST_COMPOUND_12_DDIHLOROBENZENE	1,2-Ddihlorobenzene
NIST_COMPOUND_DICHLORODIETHYL_ETHER	Dichlorodiethyl Ether
NIST_COMPOUND_12_DICHLOROETHANE	1,2-Dichloroethane
NIST_COMPOUND_DIETHYL_ETHER	Diethyl Ether
NIST_COMPOUND_NN_DIMETHYL_FORMAMIDE	N,N-Dimethyl Formamide
NIST_COMPOUND_DIMETHYL_SULFOXIDE	Dimethyl Sulfoxide
NIST_COMPOUND_ETHANE	Ethane
NIST_COMPOUND_ETHYL_ALCOHOL	Ethyl Alcohol
NIST_COMPOUND_ETHYL_CELLULOSE	Ethyl Cellulose
NIST_COMPOUND_ETHYLENE	Ethylene
NIST_COMPOUND_EYE_LENS_ICRP	Eye Lens (ICRP)
NIST_COMPOUND_FERRIC_OXIDE	Ferric Oxide

NIST compound catalog macro	NIST compound catalog name
NIST_COMPOUND_FERROBORIDE	Ferroboride
NIST_COMPOUND_FERROUS_OXIDE	Ferrous Oxide
NIST_COMPOUND_FERROUS_SULFATE_DOSIMETER_SOLUTION	Ferrous Sulfate Dosimeter Solution
NIST_COMPOUND_FREON_12	Freon-12
NIST_COMPOUND_FREON_12B2	Freon-12B2
NIST_COMPOUND_FREON_13	Freon-13
NIST_COMPOUND_FREON_13B1	Freon-13B1
NIST_COMPOUND_FREON_13I1	Freon-13I1
NIST_COMPOUND_GADOLINIUM_OXYSULFIDE	Gadolinium Oxysulfide
NIST_COMPOUND_GALLIUM_ARSENIDE	Gallium Arsenide
NIST_COMPOUND_GEL_IN_PHOTOGRAPHIC_EMULSION	Gel in Photographic Emulsion
NIST_COMPOUND_GLASS_PYREX	Glass, Pyrex
NIST_COMPOUND_GLASS_LEAD	Glass, Lead
NIST_COMPOUND_GLASS_PLATE	Glass, Plate
NIST_COMPOUND_GLUCOSE	Glucose
NIST_COMPOUND_GLUTAMINE	Glutamine
NIST_COMPOUND_GLYCEROL	Glycerol
NIST_COMPOUND_GUANINE	Guanine
NIST_COMPOUND_GYPSUM_PLASTER_OF_PARIS	Gypsum, Plaster of Paris
NIST_COMPOUND_N_HEPTANE	N-Heptane
NIST_COMPOUND_N_HEXANE	N-Hexane
NIST_COMPOUND_KAPTON_POLYIMIDE_FILM	Kapton Polyimide Film
NIST_COMPOUND_LANTHANUM_OXYBROMIDE	Lanthanum Oxybromide
NIST_COMPOUND_LANTHANUM_OXYSULFIDE	Lanthanum Oxysulfide
NIST_COMPOUND_LEAD_OXIDE	Lead Oxide
NIST_COMPOUND_LITHIUM_AMIDE	Lithium Amide
NIST_COMPOUND_LITHIUM_CARBONATE	Lithium Carbonate
NIST_COMPOUND_LITHIUM_FLUORIDE	Lithium Fluoride
NIST_COMPOUND_LITHIUM_HYDRIDE	Lithium Hydride
NIST_COMPOUND_LITHIUM_IODIDE	Lithium Iodide
NIST_COMPOUND_LITHIUM_OXIDE	Lithium Oxide
NIST_COMPOUND_LITHIUM_TETRABORATE	Lithium Tetraborate
NIST_COMPOUND_LUNG_ICRP	Lung (ICRP)
NIST_COMPOUND_M3_WAX	M3 Wax
NIST_COMPOUND_MAGNESIUM_CARBONATE	Magnesium Carbonate
NIST_COMPOUND_MAGNESIUM_FLUORIDE	Magnesium Fluoride
NIST_COMPOUND_MAGNESIUM_OXIDE	Magnesium Oxide
NIST_COMPOUND_MAGNESIUM_TETRABORATE	Magnesium Tetraborate
NIST_COMPOUND_MERCURIC_IODIDE	Mercuric Iodide
NIST_COMPOUND_METHANE	Methane
NIST_COMPOUND_METHANOL	Methanol

NIST compound catalog macro	NIST compound catalog name
NIST_COMPOUND_MIX_D_WAX	Mix D Wax
NIST_COMPOUND_MS20_TISSUE_SUBSTITUTE	MS20 Tissue Substitute
NIST_COMPOUND_MUSCLE_SKELETAL	Muscle, Skeletal
NIST_COMPOUND_MUSCLE_STRIATED	Muscle, Striated
NIST_COMPOUND_MUSCLE_EQUIVALENT_LIQUID_WITH_SUCROSE	Muscle-Equivalent Liquid, with Sucrose
NIST_COMPOUND_MUSCLE_EQUIVALENT_LIQUID_WITHOUT_SUCROSE	Muscle-Equivalent Liquid, without Sucrose
NIST_COMPOUND_NAPHTHALENE	Naphthalene
NIST_COMPOUND_NITROBENZENE	Nitrobenzene
NIST_COMPOUND_NITROUS_OXIDE	Nitrous Oxide
NIST_COMPOUND_NYLON_DU_PONT_ELVAMIDE_8062	Nylon, Du Pont ELVAmide 8062
NIST_COMPOUND_NYLON_TYPE_6_AND_TYPE_66	Nylon, type 6 and type 6/6
NIST_COMPOUND_NYLON_TYPE_610	Nylon, type 6/10
NIST_COMPOUND_NYLON_TYPE_11_RILSAN	Nylon, type 11 (Rilsan)
NIST_COMPOUND_OCTANE_LIQUID	Octane, Liquid
NIST_COMPOUND_PARAFFIN_WAX	Paraffin Wax
NIST_COMPOUND_N_PENTANE	N-Pentane
NIST_COMPOUND_PHOTOGRAPHIC_EMULSION	Photographic Emulsion
NIST_COMPOUND_PLASTIC_SCINTILLATOR_VINYLTOLUENE_BASED	Plastic Scintillator (Vinyltoluene based)
NIST_COMPOUND_PLUTONIUM_DIOXIDE	Plutonium Dioxide
NIST_COMPOUND_POLYACRYLONITRILE	Polyacrylonitrile
NIST_COMPOUND_POLYCARBONATE_MAKROLON_LEXAN	Polycarbonate (Makrolon, Lexan)
NIST_COMPOUND_POLYCHLOROSTYRENE	Polychlorostyrene
NIST_COMPOUND_POLYETHYLENE	Polyethylene
NIST_COMPOUND_POLYETHYLENE_TEREPHTHALATE_MYLAR	Polyethylene Terephthalate (Mylar)
NIST_COMPOUND_POLYMETHYL_METHACRALATE_LUCITE_PERSPEX	Polymethyl Methacralate (Lucite, Perspex)
NIST_COMPOUND_POLYOXYMETHYLENE	Polyoxymethylene
NIST_COMPOUND_POLYPROPYLENE	Polypropylene
NIST_COMPOUND_POLYSTYRENE	Polystyrene
NIST_COMPOUND_POLYTETRAFLUOROETHYLENE_TEFLON	Polytetrafluoroethylene (Teflon)
NIST_COMPOUND_POLYTRIFLUOROCHLOROETHYLENE	Polytrifluorochloroethylene
NIST_COMPOUND_POLYVINYL_ACETATE	Polyvinyl Acetate
NIST_COMPOUND_POLYVINYL_ALCOHOL	Polyvinyl Alcohol
NIST_COMPOUND_POLYVINYL_BUTYRAL	Polyvinyl Butyral
NIST_COMPOUND_POLYVINYL_CHLORIDE	Polyvinyl Chloride

NIST compound catalog macro	NIST compound catalog name
NIST_COMPOUND_POLYVINYLIDENE_CHLORIDE_SARAN	Polyvinylidene Chloride, Saran
NIST_COMPOUND_POLYVINYLIDENE_FLUORIDE	Polyvinylidene Fluoride
NIST_COMPOUND_POLYVINYL_PYRROLIDONE	Polyvinyl Pyrrolidone
NIST_COMPOUND_POTASSIUM_IODIDE	Potassium Iodide
NIST_COMPOUND_POTASSIUM_OXIDE	Potassium Oxide
NIST_COMPOUND_PROPANE	Propane
NIST_COMPOUND_PROPANE_LIQUID	Propane, Liquid
NIST_COMPOUND_N_PROPYL_ALCOHOL	N-Propyl Alcohol
NIST_COMPOUND_PYRIDINE	Pyridine
NIST_COMPOUND_RUBBER_BUTYL	Rubber, Butyl
NIST_COMPOUND_RUBBER_NATURAL	Rubber, Natural
NIST_COMPOUND_RUBBER_NEOPRENE	Rubber, Neoprene
NIST_COMPOUND_SILICON_DIOXIDE	Silicon Dioxide
NIST_COMPOUND_SILVER_BROMIDE	Silver Bromide
NIST_COMPOUND_SILVER_CHLORIDE	Silver Chloride
NIST_COMPOUND_SILVER_HALIDES_IN_PHOTOGRAPHIC_EMULSION	Silver Halides in Photographic Emulsion
NIST_COMPOUND_SILVER_IODIDE	Silver Iodide
NIST_COMPOUND_SKIN_ICRP	Skin (ICRP)
NIST_COMPOUND_SODIUM_CARBONATE	Sodium Carbonate
NIST_COMPOUND_SODIUM_IODIDE	Sodium Iodide
NIST_COMPOUND_SODIUM_MONOXIDE	Sodium Monoxide
NIST_COMPOUND_SODIUM_NITRATE	Sodium Nitrate
NIST_COMPOUND_STILBENE	Stilbene
NIST_COMPOUND_SUCROSE	Sucrose
NIST_COMPOUND_TERPHENYL	Terphenyl
NIST_COMPOUND_TESTES_ICRP	Testes (ICRP)
NIST_COMPOUND_TETRACHLOROETHYLENE	Tetrachloroethylene
NIST_COMPOUND_THALLIUM_CHLORIDE	Thallium Chloride
NIST_COMPOUND_TISSUE_SOFT_ICRP	Tissue, Soft (ICRP)
NIST_COMPOUND_TISSUE_SOFT_ICRU_FOUR_COMPONENT	Tissue, Soft (ICRU four-component)
NIST_COMPOUND_TISSUE_EQUIVALENT_GAS_METHANE_BASED	Tissue-Equivalent GAS (Methane based)
NIST_COMPOUND_TISSUE_EQUIVALENT_GAS_PROPANE_BASED	Tissue-Equivalent GAS (Propane based)
NIST_COMPOUND_TITANIUM_DIOXIDE	Titanium Dioxide
NIST_COMPOUND_TOLUENE	Toluene
NIST_COMPOUND_TRICHLOROETHYLENE	Trichloroethylene
NIST_COMPOUND_TRIETHYL_PHOSPHATE	Triethyl Phosphate
NIST_COMPOUND_TUNGSTEN_HEXAFLUORIDE	Tungsten Hexafluoride
NIST_COMPOUND_URANIUM_DICARBIDE	Uranium Dicarbide

NIST compound catalog macro	NIST compound catalog name
NIST_COMPOUND_URANIUM_MONOCARBIDE	Uranium Monocarbide
NIST_COMPOUND_URANIUM_OXIDE	Uranium Oxide
NIST_COMPOUND_UREA	Urea
NIST_COMPOUND_VALINE	Valine
NIST_COMPOUND_VITON_FLUOROELASTOMER	Viton Fluoroelastomer
NIST_COMPOUND_WATER_LIQUID	Water, Liquid
NIST_COMPOUND_WATER_VAPOR	Water Vapor
NIST_COMPOUND_XYLENE	Xylene

A.7 Radionuclide macros

The functions `GetRadioNuclideDataByIndex` and `GetRadioNuclideDataByName` provide access to the X-ray and gamma intensity profiles of several commonly used radionuclides, obtained from the website of the Lawrence Berkeley National Laboratory.
The following table has two columns: the first for the macros that are intended to be used as argument to `GetRadioNuclideDataByIndex`, while the second contains the names that should be provided to the `GetRadioNuclideDataByName` function. The list of names in the second column is identical to the array of strings that can be queried using `GetRadioNuclideDataList`.

Radionuclide macros	Radionuclide name
RADIO_NUCLIDE_55FE	55Fe
RADIO_NUCLIDE_57CO	57Co
RADIO_NUCLIDE_109CD	109Cd
RADIO_NUCLIDE_125I	125I
RADIO_NUCLIDE_137CS	137Cs
RADIO_NUCLIDE_133BA	133Ba
RADIO_NUCLIDE_153GD	153Gd
RADIO_NUCLIDE_238PU	238Pu
RADIO_NUCLIDE_241AM	241Am
RADIO_NUCLIDE_244CM	244Cm