Joint Center for Satellite Data Assimilation

CRTM Utility: LBLRTM I/O library v0.1.0 User Guide

January 8, 2014; rev35422

Change History

Date	Author	Change
2014-01-08	P.van Delst	Initial release.

Contents

List of Figures

List of Tables

What's Next

Single- and double-precision aware procedures Currently the setting of the default real and integer kind definitions to work with either single- or double-precision LBLRTM files is done via the compilation step via preprocessing. That is, the default real and integer kind definitions are explicitly set. So, once the library is built a partiular way, e.g. to read double-precision files, it cannot be used to read the other type. This was done because (a) it used already existing functionality in the CRTM codebase, and (b) for the CRTM at least we always use double-precision files.

However, to make the library more flexible, the relevant definitions can be made and procedures overloaded for programmatic switching between reading single- or double-precision LBLRTM datafiles.

NetCDF4 conversion and I/O A conversion procedure is planned to read the LBLRTM format datafile and output a netCDF4 datafile.

More comprehensive use of OOP This is being planned mostly to insulate users from changes to the underlying data structures and objects. Procedures to get and set attributes of the various objects are planned so that direct access of those attributes is not necessary. Use of type-bound procedures is also planned to simplify the calling syntax. Fully-compliant Fortran2003 compilers are still relatively rare so this change may take some time.

1 Introduction

1.1 Components

The LBLRTM I/O library is constructed around five data constructs, described in table 1.1:

Table 1.1: The data constructs of the LBLRTM I/O library

Component Name	**Description**
`Fhdr`	The file header construct that is present at the start of each layer of data.
`Phdr`	This is the panel header construct that is present at the start of each "chunk" of data (usually referred to as a "panel". See following.)
`Panel`	This construct corresponds to a "chunk" of spectral data. An LBLRTM datafile is referred to as being a single- or double-panel file. The former means a single spectral quantity is present (e.g. optical depth), and the latter means that two spectral quantities are present (e.g. radiance and transmittances).
`Layer`	This construct contains spectral data for the entire frequency range of an LBLRTM calculation for a single layer. The concept "layer" can correspond to the spectral data for individual atmospheric layers of the input profile, or to the final result for an entire atmosphere.
`File`	This contruct is true to its name. It corresponds to an entire datafile of data which may consist of a single layer or multiple layers, and for single- or double-panel spectral data.

Each component has a definition module to define the object and some basic methods to manipulate it, and an I/O module to read and write instances of the objects from/to file.

Two components – the file header and panel header – are standalone, but the others contain other components, i.e. the panel object contains panel headers; the layer object contains file headers; and the file object contains layers. A schematic illustration of how the actual LBLRTM datafile format relates to the component definitions is shown in figure 1.1.

Note, however, that when an LBLRTM file is read, the individual panel "chunks" of spectral data are concatenated into a single spectrum. Thus the `Panel` object itself is only used when reading from an LBLRTM file and is not used in the `File` or `Layer` objects.

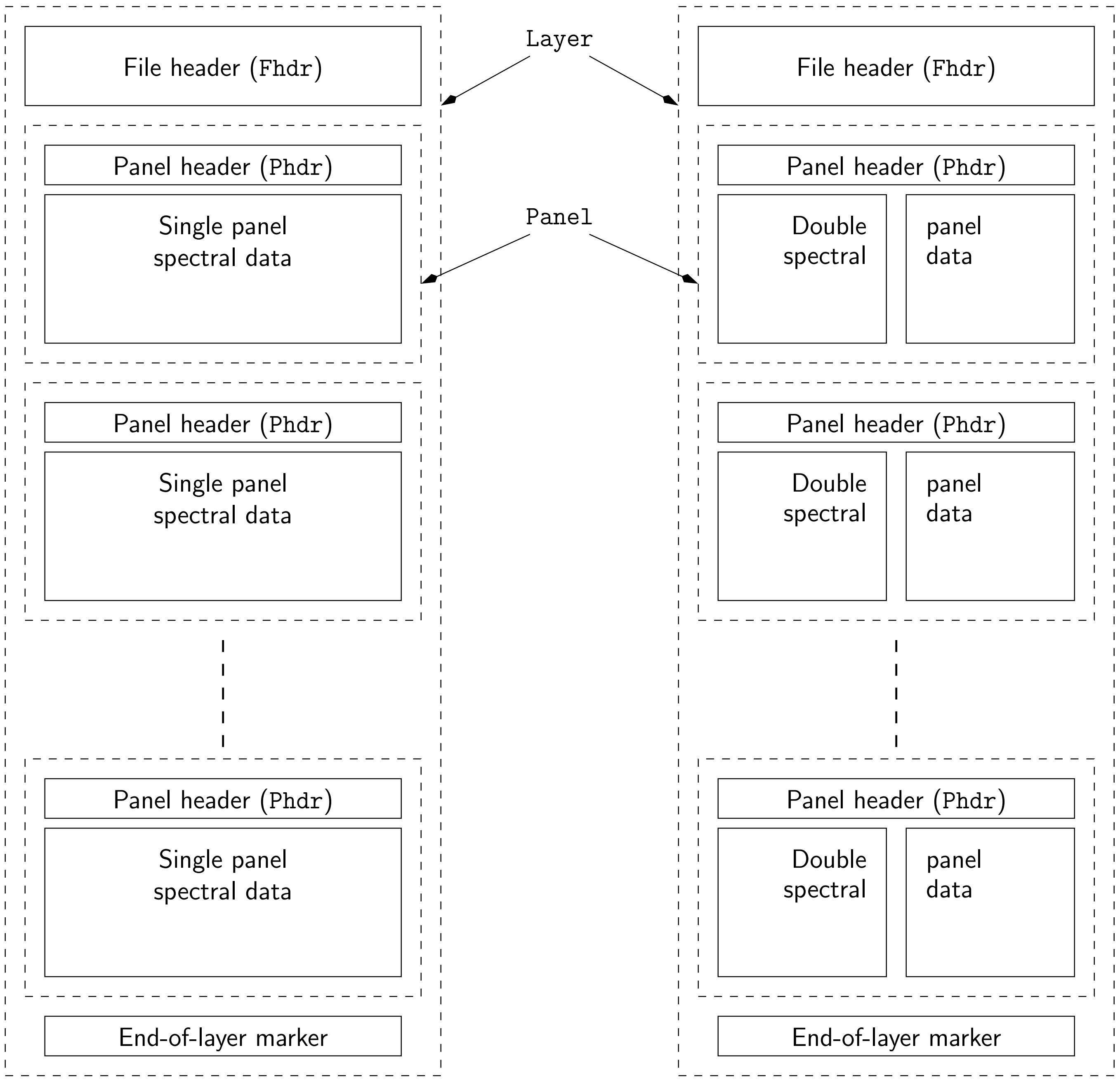

Figure 1.1: Schematic illustration of the LBLRTM single- and double-panel datafile format. A datafile can contain one, or multiple, layers of data.

1.2 Conventions

The following are conventions that have been adhered to in the current release of the LBLRTM I/O library. They are guidelines intended to make understanding the code at a glance easier, to provide a recognisable "look and feel", and to minimise name space clashes.

1.2.1 Naming of Objects and Instances of Objects

The object[1] naming convention adopted for use in the LBLRTM I/O library is,

`LBLRTM_`*name*`_type`

where *name* is an identifier for the particular component (e.g. panel header, layer, file, etc as listed in table 1.1). All object type names are suffixed with "`_type`". The "`LBLRTM_`" prefix is to define a namespace to minimise name clashes. An instance of a object is then referred to via its *name*, or some sort of derivate of its *name*. Some object declarations examples are,

```
TYPE(LBLRTM_File_type) :: sp_file, dp_file
TYPE(LBLRTM_Layer_type) :: layer
```

1.2.2 Naming of Definition Modules

Modules containing object definitions are termed *definition modules*. These modules contain the actual object definitions as well as various utility procedures that operate on the object. The naming convention adopted for definition modules in the LBLRTM I/O library is,

`LBLRTM_`*name*`_Define`

where all definition module names are suffixed with "`_Define`". The actual source code files for these modules have the same name with a "`.f90`" suffix.

1.2.3 Naming of I/O Modules

Modules containing all the object I/O procedures are termed, surprise, surprise, *I/O modules*. These modules contain function to read and write LBLRTM format datafiles. The naming convention adopted for I/O modules in the LBLRTM I/O library is,

`LBLRTM_`*name*`_IO`

where all I/O module names are suffixed with "`_IO`". As with the definition modules, the actual source code files for these modules have the same name with a "`.f90`" suffix.

1.2.4 Standard Definition Module Procedures

The definition modules for the user-accessible objects (for practical purposes just `File`, although `Layer`, `Fhdr`, `Panel`, and `Phdr`are accessible for now) contain a standard set of procedures for use with the object being defined. The naming convention for these procedures is,

`LBLRTM_`*name*_*action*

where the available default actions for each procedure are listed in table 1.2. This is not an exhaustive list but procedures for the actions listed in table 1.2 are generally going to be present.

The exception is that the objects with no allocatable components do not have a creation procedure.

[1]The terms "derived type" and "structure" can also be used as the code is not yet fully OO - that's for future updates.

Table 1.2: Default action procedures available in object definition modules. †Procedures not available for the `Fhdr` and `Phdr` objects. ‡Procedure available only for the `Layer` object.

Action	**Type**	**Description**
`OPERATOR(==)`	Elemental function	Tests the equality of two structures.
`OPERATOR(/=)`	Elemental function	Tests the inequality of two structures.
`Associated`†	Elemental function	Tests if the object components have been allocated.
`Create`†	Elemental subroutine	Allocates any allocatable object components.
`Destroy`	Elemental subroutine	Reinitialises an object.
`DefineVersion`	Subroutine	Returns the module version information.
`Frequency`‡	Subroutine	Compute and return the spectral frequency grid.
`Inspect`	Subroutine	Displays object contents to `stdout`.
`IsValid`	Elemental function	Tests if the object contains valid data.
`SetValid`	Elemental subroutine	Flags the object as containing valid data.

Table 1.3: Default action procedures available in object I/O modules.

Action	**Type**	**Description**
`IOVersion`	Subroutine	Returns the module version information.
`Read`	Function	Loads an instance of an object with data read from file.
`Write`	Function	Write an instance of an object to file.

Some examples of these procedure names are,

```
LBLRTM_File_Associated
LBLRTM_File_IsValid
LBLRTM_Layer_Destroy
LBLRTM_Layer_Frequency
LBLRTM_File_Inspect
```

The relational operators, `==` and `/=`, are implemented via overloaded `Equal` and `NotEqual` action procedures respectively, as is shown below for the `File` structure,

```
INTERFACE OPERATOR(==)
  MODULE PROCEDURE LBLRTM_File_Equal
END INTERFACE OPERATOR(==)

INTERFACE OPERATOR(/=)
  MODULE PROCEDURE LBLRTM_File_NotEqual
END INTERFACE OPERATOR(/=)
```

For a complete list of the definition and I/O module procedures for use with the available objects, see appendix A.

2 How to obtain the LBLRTM I/O library

The LBLRTM I/O library source code is released in a compressed tarball via the CRTM ftp site utility directory:

`ftp://ftp.emc.ncep.noaa.gov/jcsda/CRTM/utility/lblrtmio`

The v0.1.0 release is available directly from

`ftp://ftp.emc.ncep.noaa.gov/jcsda/CRTM/utility/lblrtmio/v0.1.0`

Also note that additional releases, e.g. beta or experimental branches, may also made available on this ftp site.

3 How to build the LBLRTM I/O library

3.1 Configuration

The LBLRTM I/O tarball directory structure looks like:

```
 ./
  |-README  .................. library build instructions
  |-configure  ............... configuration script
  |-Makefile.in  ............. makefile template
  `-libsrc/  ................. library source files
```

The build system for the LBLRTM I/O library uses a GNU autoconf-generated configure script.

You run the configuration script like so:

```
$ ./configure --prefix=install-directory
```

The `--prefix` switch sets the installation directory and defaults to `/usr/local` so make sure you set it to a directory in which you have write access.

By default, the LBLRTM I/O library is set to read files generated from a double-precision LBLRTM executable (what is used in the CRTM project by default). If you want to read datafiles from a "standard" single precision LBLRTM executable, it needs to be enabled in the configuration:

```
$ ./configure --enable-single --prefix=install-directory
```

The switch between single-and double-precision capability is achieved via preprocessing that redefines the default real and integer kind types. By default the configuration step sets preprocessing macros `INT_SIZE` and `REAL_SIZE` to a value of 8 (indicating byte size) to allow for reading of double-precision files. The `--enable-single` switch sets the value of those macros to 4 for use with single-precision files.

Note, however, that the test datafiles provided with the LBLRTM I/O library are double-precision files generated on a little-endian machine. So if you build the single-precision version of the LBLRTM I/O library the associated test will fail.

3.1.1 Supported compilers

The LBLRTM I/O library configuration is set up for the following four compilers:

- ifort
- gfortran
- xlf2003
- pgf95

For these compilers, the configuration file will use the correct compiler switches to promote the single-precision real and integer variables to double-precision by default. For any other compilers, please contact CRTM support[a] for information on how to get the library built.

3.2 Building the library

Once the `configure` script has been run successfully, to start building the library simply type

```
$ make
```

3.3 Checking the library build

To run the accompanying tests using the just-built library, simply type

```
$ make check
```

This will build and run any tests. The current output from the (successful) test runs looks like:

```
         ***********************************************************
                               check_lblrtmio

          Check/example program for the LBLRTM File I/O functions
          using

          LBLRTM I/O library version: v0.1.0
         ***********************************************************


         Test reading a single layer, single panel LBLRTM file...

     LBLRTM_Utility::File_Open(INFORMATION) : Set for DOUBLE-precision LBLRTM files
     LBLRTM_File_IO::Read(INFORMATION) : Reading layer #1...
     LBLRTM_Layer_IO::Read(INFORMATION) :   Reading spectral chunk #1...
     LBLRTM_Panel_IO::Read(INFORMATION) :     Reading spectrum #1...
     LBLRTM_Layer_IO::Read(INFORMATION) :   Reading spectral chunk #2...
     LBLRTM_Panel_IO::Read(INFORMATION) :     Reading spectrum #1...

     ...etc...

          Test reading a single layer, double panel LBLRTM file...
```

[a]ncep.list.emc.jcsda_crtm.support@noaa.gov

```
LBLRTM_Utility::File_Open(INFORMATION) : Set for DOUBLE-precision LBLRTM files
LBLRTM_File_IO::Read(INFORMATION) : Reading layer #1...
LBLRTM_Layer_IO::Read(INFORMATION) :   Reading spectral chunk #1...
LBLRTM_Panel_IO::Read(INFORMATION) :     Reading spectrum #1...
LBLRTM_Panel_IO::Read(INFORMATION) :     Reading spectrum #2...
LBLRTM_Layer_IO::Read(INFORMATION) :   Reading spectral chunk #2...
LBLRTM_Panel_IO::Read(INFORMATION) :     Reading spectrum #1...
LBLRTM_Panel_IO::Read(INFORMATION) :     Reading spectrum #2...

...etc...

    Test reading some layers from a multiple layer, double panel LBLRTM file...

LBLRTM_Utility::File_Open(INFORMATION) : Set for DOUBLE-precision LBLRTM files
LBLRTM_File_IO::Read(INFORMATION) : Reading layer #1...
LBLRTM_Layer_IO::Read(INFORMATION) :   Reading spectral chunk #1...
LBLRTM_Panel_IO::Read(INFORMATION) :     Reading spectrum #1...
LBLRTM_Panel_IO::Read(INFORMATION) :     Reading spectrum #2...
LBLRTM_Layer_IO::Read(INFORMATION) :   Reading spectral chunk #2...
LBLRTM_File_IO::Read(INFORMATION) : Reading layer #2...
LBLRTM_Layer_IO::Read(INFORMATION) :   Reading spectral chunk #1...
LBLRTM_Panel_IO::Read(INFORMATION) :     Reading spectrum #1...
LBLRTM_Panel_IO::Read(INFORMATION) :     Reading spectrum #2...

...etc...

 TEST SUCCESSFUL!
```

As mentioned previously, the test datafiles used in the above check target are little-endian double-precision datafiles.

3.4 Installing the library

To install the library, type:

```
$ make install
```

Installation of the library *always* occurs into its own directory within the directory specified by the `--prefix` switch. The name of the installation directory follows the convention:

library-name_version

So, if a library version (say, v1.0.0) build was configured with `--prefix=$PWD`, then the installation directory will be

```
${PWD}/lblrtmio_v1.0.0
```

3.5 Linking to the library

Let's assume the above install was moved into `$HOME/local`. To use the library in your own application, the usual environment variables would be modified something like:

```
  libroot="${HOME}/local/lblrtmio_v1.0.0"
  FCFLAGS="-I${libroot}/include ${FCFLAGS}"
  LDFLAGS="-L${libroot}/lib ${LDFLAGS}"
  LIBS="-llblrtmio"
```

(with appropriate syntax changes for `csh`)

3.6 Uninstalling the library

To uninstall the library (assuming you haven't moved the installation directory contents somewhere else) you can type:

```
$ make uninstall
```

This will *delete* the created installation directory. So, for a library version, say, v1.0.0, if your configure script invocation was something like

`$ ./configure --prefix=$PWD` ...other command line arguments...

then the `uninstall` target will delete the `$PWD/lblrtmio_v1.0.0` directory.

3.7 Cleaning up

Two targets are provided for cleaning up after the build. To remove all of the build products type

```
$ make clean
```

To also remove all of the configuration products (i.e. the generated `makefiles`) type

```
$ make distclean
```

3.8 Feedback and contact information

That's pretty much it. Any questions or bug reports can be sent to CRTM Support.

ncep.list.emc.jcsda_crtm.support@noaa.gov

If you have problems building the library please include the generated `config.log` file in your email correspondence.

4 How to use the LBLRTM I/O library

This section will hopefully get you started using the LBLRTM I/O library as quickly as possible. We will only be referring to the reading of datafiles in this section.

4.1 Environment setup in your Fortran program

All of the LBLRTM I/O user procedures, parameters, and derived data type definitions are accessible via the container module `LBLRTMIO_Module`. Thus, one needs to put the following statement in any calling program, module or procedure,

```
USE LBLRTMIO_Module
```

Once you become more familiar with the components of the LBLRTM I/O library you require, you can also specify an `ONLY` clause with the `USE` statement,

`USE LBLRTMIO_Module`[`, ONLY:`*only-list*]

where *only-list* is a list of the symbols you want to "import" from `LBLRTMIO_Module`. This latter form is the preferred style for self-documenting your code; e.g. when you give the code to someone else, they will be able to identify from which module various symbols in your code originate.

You can find out what version of the LBLRTM I/O library you are using by calling the `LBLRTMIO_Version` subroutine.

4.2 Define the LBLRTM `File` object

An LBLRTM `File` object is declared like so,

```
TYPE(LBLRTM_File_type) :: ofile
```

One additional variable required is an integer to hold the error status of the I/O functions, e.g.

```
INTEGER :: err_stat
```

That's pretty much it.

4.3 Call the LBLRTM File read function

Below are three examples of calling the LBLRTM `File` read function. See A.6.9 for the full description of the function interface.

4.3.1 Single layer, single panel LBLRTM file

Reading a single layer, single panel file (e.g. an optical depth file) is the simplest. Let's call the input file "`ODdeflt_100`". To read it, the syntax is:

```
err_stat = LBLRTM_File_Read(ofile, 'ODdeflt_100')
IF ( err_stat /= SUCCESS ) THEN
  handle error...
END IF
```

The error status return code "`SUCCESS`" is defined in the library.

4.3.2 Single layer, double panel LBLRTM file

Because there is no unambiguous way to determine if an LBLRTM file is single or double panel, if a file is a double panel file (e.g. containing radiance and transmittance data) that information must be supplied to the read function. By default the function reads single panel files. To read a double panel file (let's call this one "`TAPE12`"), the syntax is:

```
err_stat = LBLRTM_File_Read(ofile, 'TAPE12', Double_Panel=.TRUE.)
IF ( err_stat /= SUCCESS ) THEN
  handle error...
END IF
```

The optional logical argument "`Double_Panel`" causes the `File` object to be correctly allocated to hold both sets of spectral data.

4.3.3 Multiple layer, double panel LBLRTM file

As with the single/double panel format issue, there is also no unambiguous way to determine up front if an LBLRTM file contains multiple layers. So, again, that information must be supplied to the read function. By default the function reads only a single layer. To read multiple layers from a file (let's call this one "`TAPE13`"), the syntax is:

```
err_stat = LBLRTM_File_Read(ofile, 'TAPE13', n_Layers=10, Double_Panel=.TRUE.)
IF ( err_stat /= SUCCESS ) THEN
  handle error...
END IF
```

The optional integer argument "`n_Layers`" specifies the amount of data allocated for the `File` object and the number of layers to read in. If there are more layers in the file than that specified, they are ignored.

4.4 Inspecting the File object contents

You can dump the contents of the LBLRTM `File` object to stdout via the `LBLRTM_File_Inspect` subroutine like so:

```
CALL LBLRTM_File_Inspect(ofile)
```

Note that this will generate a *lot* of output so it's mostly useful for debugging purposes.

4.5 Accessing the File object contents

The LBLRTM I/O library is (not yet) fully object oriented. As such, there are no `Get` methods for the `File` object and access to the data is done via direct reference to the object components. See figures A.5 and A.4 for the complete `File` and `Layer` object definitions respetively.

The following code snippet shows how the individual layer spectra can be accessed, including use of a `Layer` object "method" to compute the frequency grid for the spectrum:

```
! Some type declarations
TYPE(LBLRTM_File_type) :: oFile
INTEGER :: i, k, n
REAL(DP), ALLOCATABLE :: frequency

...Read file, etc...

! Loop over the layer data
DO k = 1, oFile%n_Layers

  ! Compute the frequency grid for the current layer
  CALL LBLRTM_Layer_Frequency(oFile%Layer(k), frequency)
  IF ( .NOT. ALLOCATED(frequency) ) THEN
    handle error...
  END IF

  ! Loop over the spectra (i.e. single- or double-panel)
  DO n = 1, oFile%Layer(k)%n_Spectra

    ! Loop over the spectral points
    DO i = 1, oFile%Layer(k)%n_Points

      ! Display current spectrum value for each frequency
      PRINT *, f(i), oFile%Layer(k)%Spectrum(i,n)

    END DO
  END DO

  ! Not strictly necessary, but a good habit
  DEALLOCATE(frequency)

END DO
```

4.6 Cleaning up

You don't have to explicitly destroy LBLRTM `File` objects, but it's a good habit to get into. Deallocation of the `File` object is done via the `LBLRTM_File_Destroy` subroutine like so:

```
CALL LBLRTM_File_Destroy(ofile)
```

A
Object and procedure interface definitions

A.1 Main LBLRTM I/O Module

A.1.1 LBLRTMIO_Version interface

```
NAME:
      LBLRTMIO_Version

PURPOSE:
      Subroutine to the LBLRTM I/O version information.

CALLING SEQUENCE:
      CALL LBLRTMIO_Version( version )

OUTPUTS:
      version:       Character string identifying the LBLRTM I/O library
                     release version.
                     UNITS:      N/A
                     TYPE:       CHARACTER(*)
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(OUT)
```

A.2 Fhdr Structure

```
TYPE :: LBLRTM_Fhdr_type
  LOGICAL :: Is_Valid = .FALSE.
  CHARACTER(80)       :: User_ID                         = ''
  REAL(DP)            :: Column_Scale_Factor             = 0.0_DP
  REAL(FP)            :: Average_Layer_Pressure          = 0.0_FP
  REAL(FP)            :: Average_Layer_Temperature       = 0.0_FP
  CHARACTER(8)        :: Molecule_Id(N_MOL)              = ''
  REAL(FP)            :: Molecule_Column_Density(N_MOL) = 0.0_FP
  REAL(FP)            :: Broadening_Gas_Column_Density  = 0.0_FP
  REAL(FP)            :: Frequency_Interval              = 0.0_FP
  REAL(DP)            :: Begin_Frequency                 = 0.0_DP
  REAL(DP)            :: End_Frequency                   = 0.0_DP
  REAL(FP)            :: Boundary_Temperature            = 0.0_FP
  REAL(FP)            :: Boundary_Emissivity             = 0.0_FP
  INTEGER(IP)         :: n_Molecules                     = 0_IP
  INTEGER(IP)         :: n_Layer                         = 0_IP
  INTEGER(IP)         :: OD_Layering_Control_Flag        = 0_IP
  CHARACTER(8)        :: Calculation_Date                = ''
  CHARACTER(8)        :: Calculation_Time                = ''
  CHARACTER(8)        :: ancillary(8)                    = ''
  TYPE(RunFlags_type) :: RunFlags
END TYPE LBLRTM_Fhdr_type
```

Figure A.1: LBLRTM_Fhdr_type structure definition.

A.2.1 LBLRTM_Fhdr_DefineVersion interface

```
NAME:
      LBLRTM_Fhdr_DefineVersion

PURPOSE:
      Subroutine to return the version information for the
      definition module(s).

CALLING SEQUENCE:
      CALL LBLRTM_Fhdr_DefineVersion( Id )

OUTPUTS:
      Id:     Character string containing the version Id information for the
              definition module(s).
              UNITS:      N/A
              TYPE:       CHARACTER(*)
              DIMENSION:  Scalar
              ATTRIBUTES: INTENT(OUT)
```

A.2.2 LBLRTM_Fhdr_Destroy interface

```
NAME:
      LBLRTM_Fhdr_Destroy

PURPOSE:
      Elemental subroutine to re-initialize LBLRTM_Fhdr objects.

CALLING SEQUENCE:
      CALL LBLRTM_Fhdr_Destroy( LBLRTM_Fhdr )

OBJECTS:
      LBLRTM_Fhdr:  Re-initialized LBLRTM_Fhdr instance.
                    UNITS:      N/A
                    TYPE:       LBLRTM_Fhdr_type
                    DIMENSION:  Scalar or any rank
                    ATTRIBUTES: INTENT(OUT)
```

A.2.3 LBLRTM_Fhdr_Inspect interface

```
NAME:
      LBLRTM_Fhdr_Inspect

PURPOSE:
      Subroutine to print the contents of an instance of an LBLRTM_Fhdr
      object to stdout.

CALLING SEQUENCE:
      CALL LBLRTM_Fhdr_Inspect( LBLRTM_Fhdr )

OBJECTS:
      LBLRTM_Fhdr:   LBLRTM_Fhdr object to display.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Fhdr_type
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN)
```

A.2.4 LBLRTM_Fhdr_IsValid interface

```
NAME:
      LBLRTM_Fhdr_IsValid

PURPOSE:
      Elemental function to test if the LBLRTM_Fhdr object contains
      valid data.

CALLING SEQUENCE:
      Status = LBLRTM_Fhdr_IsValid( LBLRTM_Fhdr )

OBJECTS:
      LBLRTM_Fhdr:   Instance which is to have its status tested.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Fhdr_type
                     DIMENSION:  Scalar or any rank
                     ATTRIBUTES: INTENT(IN)

FUNCTION RESULT:
      Status:        The return value is a logical value indicating
                     if the object contains valid data.
                     UNITS:      N/A
                     TYPE:       LOGICAL
                     DIMENSION:  Same as input
```

A.2.5 LBLRTM_Fhdr_SetValid interface

```
NAME:
      LBLRTM_Fhdr_SetValid

PURPOSE:
      Elemental subroutine to mark an instance of an LBLRTM_Fhdr object
      as containing valid data.
```

```
CALLING SEQUENCE:
      CALL LBLRTM_Fhdr_SetValid( LBLRTM_Fhdr )

OBJECTS:
      LBLRTM_Fhdr:   Instance which is to have its validity set.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Fhdr_type
                     DIMENSION:  Scalar or any rank
                     ATTRIBUTES: INTENT(IN)
```

A.2.6 *LBLRTM_Fhdr_IOVersion interface*

```
NAME:
      LBLRTM_Fhdr_IOVersion

PURPOSE:
      Subroutine to return the version information for the module.

CALLING SEQUENCE:
      CALL LBLRTM_Fhdr_IOVersion( Id )

OUTPUTS:
      Id:     Character string containing the version Id information for the
              module.
              UNITS:      N/A
              TYPE:       CHARACTER(*)
              DIMENSION:  Scalar
              ATTRIBUTES: INTENT(OUT)
```

A.2.7 *LBLRTM_Fhdr_Read interface*

```
NAME:
      LBLRTM_Fhdr_Read

PURPOSE:
      Function to read an LBLRTM file header from an LBLRTM format
      file.

CALLING SEQUENCE:
      Error_Status = LBLRTM_Fhdr_Read( &
                       LBLRTM_Fhdr, &
                       FileId     , &
                       EOF          )

OBJECTS:
      LBLRTM_Fhdr:  LBLRTM file header object to hold the data.
                    UNITS:      N/A
                    TYPE:       LBLRTM_Fhdr_type
                    DIMENSION:  Scalar
                    ATTRIBUTES: INTENT(OUT)

INPUTS:
      FileId:       The unit number for the already open LBLRTM file.
                    UNITS:      N/A
                    TYPE:       INTEGER
                    DIMENSION:  Scalar
                    ATTRIBUTES: INTENT(IN)

OUTPUTS:
      EOF:          Integer flag indicating end-of-file status for the
                    LBLRTM format file after the read. Valid return values
                    are defined in the LBLRTM_Parameters module.
                    If == LBLRTM_FILE_EOF:   End-Of-File has been reached.
                                             The file is then closed.
                       == LBLRTM_FILE_OK:    No EOF or EOL condition. File
                                             is positioned for further
                                             reading.
                       == LBLRTM_FILE_UNDEF: An error occurred. The file is
                                             closed.
                    UNITS:      N/A
                    TYPE:       INTEGER
                    DIMENSION:  Scalar
                    ATTRIBUTES: INTENT(OUT)

FUNCTION RESULT:
      Error_Status: The return value is an integer defining the error status.
                    The error codes are defined in the Message_Handler module.
                    If == SUCCESS the LBLRTM file header read was successful
                       == FAILURE an error occurred
                    UNITS:      N/A
                    TYPE:       INTEGER
                    DIMENSION:  Scalar

SIDE EFFECTS:
      If an error occurs or the end-of-file is encountered, the input file is
      closed.
```

A.2.8 LBLRTM_Fhdr_Write interface

```
NAME:
      LBLRTM_Fhdr_Write

PURPOSE:
      Function to write an LBLRTM file header to an LBLRTM format
      file.
```

```
CALLING SEQUENCE:
      Error_Status = LBLRTM_Fhdr_Write( &
                       LBLRTM_Fhdr, &
                       FileId      )

OBJECTS:
      LBLRTM_Fhdr:  LBLRTM file header object to write to file.
                    UNITS:      N/A
                    TYPE:       LBLRTM_Fhdr_type
                    DIMENSION:  Scalar
                    ATTRIBUTES: INTENT(IN)

INPUTS:
      FileId:       The unit number for the already open LBLRTM file.
                    UNITS:      N/A
                    TYPE:       INTEGER
                    DIMENSION:  Scalar
                    ATTRIBUTES: INTENT(IN)

FUNCTION RESULT:
      Error_Status: The return value is an integer defining the error status.
                    The error codes are defined in the Message_Handler module.
                    If == SUCCESS the LBLRTM file header write was successful
                       == FAILURE an error occurred
                    UNITS:      N/A
                    TYPE:       INTEGER
                    DIMENSION:  Scalar

SIDE EFFECTS:
      If an error occurs, the output file is closed.
```

A.3 `Phdr` Structure

```
  TYPE :: LBLRTM_Phdr_type
    LOGICAL     :: Is_Valid            = .FALSE.
    REAL(DP)    :: Begin_Frequency     = 0.0_DP
    REAL(DP)    :: End_Frequency       = 0.0_DP
    REAL(FP)    :: Frequency_Interval  = 0.0_FP
    INTEGER(IP) :: n_Points            = 0_IP
  END TYPE LBLRTM_Phdr_type
```

Figure A.2: LBLRTM_Phdr_type structure definition.

A.3.1 LBLRTM_Phdr_DefineVersion interface

```
NAME:
      LBLRTM_Phdr_DefineVersion

PURPOSE:
      Subroutine to return the version information for the
      definition module(s).

CALLING SEQUENCE:
      CALL LBLRTM_Phdr_DefineVersion( Id )

OUTPUTS:
      Id:     Character string containing the version Id information for the
              definition module(s).
              UNITS:      N/A
              TYPE:       CHARACTER(*)
              DIMENSION:  Scalar
              ATTRIBUTES: INTENT(OUT)
```

A.3.2 LBLRTM_Phdr_Destroy interface

```
NAME:
      LBLRTM_Phdr_Destroy

PURPOSE:
      Elemental subroutine to re-initialize LBLRTM_Phdr objects.

CALLING SEQUENCE:
      CALL LBLRTM_Phdr_Destroy( LBLRTM_Phdr )

OBJECTS:
      LBLRTM_Phdr:  Re-initialized LBLRTM_Phdr instance.
                    UNITS:      N/A
                    TYPE:       LBLRTM_Phdr_type
                    DIMENSION:  Scalar or any rank
                    ATTRIBUTES: INTENT(OUT)
```

A.3.3 LBLRTM_Phdr_Inspect interface

```
NAME:
      LBLRTM_Phdr_Inspect

PURPOSE:
      Subroutine to print the contents of an instance of an LBLRTM_Phdr
      object to stdout.

CALLING SEQUENCE:
      CALL LBLRTM_Phdr_Inspect( LBLRTM_Phdr )

OBJECTS:
      LBLRTM_Phdr:   LBLRTM_Phdr object to display.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Phdr_type
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN)
```

A.3.4 LBLRTM_Phdr_IsValid interface

```
NAME:
      LBLRTM_Phdr_IsValid

PURPOSE:
      Elemental function to test if the LBLRTM_Phdr object contains
      valid data.

CALLING SEQUENCE:
      Status = LBLRTM_Phdr_IsValid( LBLRTM_Phdr )

OBJECTS:
      LBLRTM_Phdr:   Instance which is to have its status tested.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Phdr_type
                     DIMENSION:  Scalar or any rank
                     ATTRIBUTES: INTENT(IN)

FUNCTION RESULT:
      Status:        The return value is a logical value indicating
                     if the object contains valid data.
                     UNITS:      N/A
                     TYPE:       LOGICAL
                     DIMENSION:  Same as input
```

A.3.5 LBLRTM_Phdr_SetValid interface

```
NAME:
      LBLRTM_Phdr_SetValid

PURPOSE:
      Elemental subroutine to mark an instance of an LBLRTM_Phdr object
      as containing valid data.
```

```
CALLING SEQUENCE:
      CALL LBLRTM_Phdr_SetValid( LBLRTM_Phdr )

OBJECTS:
      LBLRTM_Phdr:   Instance which is to have its validity set.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Phdr_type
                     DIMENSION:  Scalar or any rank
                     ATTRIBUTES: INTENT(IN)
```

A.3.6 LBLRTM_Phdr_IOVersion interface

```
NAME:
      LBLRTM_Phdr_IOVersion

PURPOSE:
      Subroutine to return the version information for the module.

CALLING SEQUENCE:
      CALL LBLRTM_Phdr_IOVersion( Id )

OUTPUTS:
      Id:     Character string containing the version Id information for the
              module.
              UNITS:      N/A
              TYPE:       CHARACTER(*)
              DIMENSION:  Scalar
              ATTRIBUTES: INTENT(OUT)
```

A.3.7 LBLRTM_Phdr_Read interface

```
NAME:
      LBLRTM_Phdr_Read

PURPOSE:
      Function to read an LBLRTM panel header from an LBLRTM format
      file.

CALLING SEQUENCE:
      Error_Status = LBLRTM_Phdr_Read( &
                       LBLRTM_Phdr, &
                       FileId     , &
                       EOF          )

OBJECTS:
      LBLRTM_Phdr:  LBLRTM panel header object to hold the data.
                    UNITS:      N/A
                    TYPE:       LBLRTM_Phdr_type
                    DIMENSION:  Scalar
                    ATTRIBUTES: INTENT(OUT)

INPUTS:
      FileId:       The unit number for the already open LBLRTM file.
                    UNITS:      N/A
                    TYPE:       INTEGER
                    DIMENSION:  Scalar
                    ATTRIBUTES: INTENT(IN)

OUTPUTS:
      EOF:          Integer flag indicating end-of-file status for the
                    LBLRTM format file after the read. Valid return values
                    are defined in the LBLRTM_Parameters module.
                    If == LBLRTM_FILE_EOF:   End-Of-File has been reached.
                                             The file is then closed.
                       == LBLRTM_FILE_EOL:   End-Of-Layer has been reached.
                                             In this case, the next read
                                             should be of the file header
                                             for the (possible) next layer.
                       == LBLRTM_FILE_OK:    No EOF or EOL condition. The
                                             next read should be the panel
                                             data.
                       == LBLRTM_FILE_UNDEF: An error occurred. The file is
                                             closed.
                    UNITS:      N/A
                    TYPE:       INTEGER
                    DIMENSION:  Scalar
                    ATTRIBUTES: INTENT(OUT)

FUNCTION RESULT:
      Error_Status: The return value is an integer defining the error status.
                    The error codes are defined in the Message_Handler module.
                    If == SUCCESS the LBLRTM panel header read was successful
                       == FAILURE an error occurred
                    UNITS:      N/A
                    TYPE:       INTEGER
                    DIMENSION:  Scalar

SIDE EFFECTS:
      If an error occurs or the end-of-file is encountered, the input file is
      closed.
```

A.3.8 LBLRTM_Phdr_Write interface

```
NAME:
      LBLRTM_Phdr_Write
```

```
PURPOSE:
      Function to write an LBLRTM panel header to an LBLRTM format
      file.

CALLING SEQUENCE:
      Error_Status = LBLRTM_Phdr_Write( &
                       LBLRTM_Phdr, &
                       FileId       )

OBJECTS:
      LBLRTM_Phdr:  LBLRTM panel header object to write to file.
                    UNITS:      N/A
                    TYPE:       LBLRTM_Phdr_type
                    DIMENSION:  Scalar
                    ATTRIBUTES: INTENT(IN)

INPUTS:
      FileId:       The unit number for the already open LBLRTM file.
                    UNITS:      N/A
                    TYPE:       INTEGER
                    DIMENSION:  Scalar
                    ATTRIBUTES: INTENT(IN)

FUNCTION RESULT:
      Error_Status: The return value is an integer defining the error status.
                    The error codes are defined in the Message_Handler module.
                    If == SUCCESS the LBLRTM panel header write was successful
                       == FAILURE an error occurred
                    UNITS:      N/A
                    TYPE:       INTEGER
                    DIMENSION:  Scalar

SIDE EFFECTS:
      If an error occurs, the output file is closed.
```

A.4 `Panel` Structure

```
TYPE :: LBLRTM_Panel_type
  ! Allocation and valid data indicator
  LOGICAL :: Is_Allocated = .FALSE.
  LOGICAL :: Is_Valid     = .FALSE.
  ! The panel header
  TYPE(LBLRTM_Phdr_type) :: Header
  ! Dimensions
  INTEGER :: n_Points  = 0 ! L
  INTEGER :: n_Spectra = 0 ! N
  ! Data
  REAL(FP), ALLOCATABLE  :: Spectrum(:,:)  ! L x N
END TYPE LBLRTM_Panel_type
```

Figure A.3: LBLRTM_Panel_type structure definition.

A.4.1 LBLRTM_Panel_Associated interface

```
NAME:
      LBLRTM_Panel_Associated

PURPOSE:
      Elemental function to test the status of the allocatable components
      of the LBLRTM_Panel object.

CALLING SEQUENCE:
      Status = LBLRTM_Panel_Associated( LBLRTM_Panel )

OBJECTS:
      LBLRTM_Panel:  Structure which is to have its member's
                     status tested.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Panel_type
                     DIMENSION:  Scalar or any rank
                     ATTRIBUTES: INTENT(IN)

FUNCTION RESULT:
      Status:        The return value is a logical value indicating if the
                     object has been allocated.
                     UNITS:      N/A
                     TYPE:       LOGICAL
                     DIMENSION:  Same as input
```

A.4.2 LBLRTM_Panel_Create interface

```
NAME:
      LBLRTM_Panel_Create

PURPOSE:
      Elemental subroutine to create an instance of an LBLRTM_Panel object.

CALLING SEQUENCE:
      CALL LBLRTM_Panel_Create( LBLRTM_Panel, &
                                LBLRTM_PHdr , &
                                n_Spectra   , &
                                Err_Msg = Err_Msg )

OBJECTS:
      LBLRTM_Panel:       LBLRTM_Panel object structure.
                          UNITS:      N/A
                          TYPE:       LBLRTM_Panel_type
                          DIMENSION:  Scalar or any rank
                          ATTRIBUTES: INTENT(OUT)

INPUTS:
      LBLRTM_Phdr:        LBLRTM_Phdr object for the current panel.
                          UNITS:      N/A
                          TYPE:       LBLRTM_Phdr_type
                          DIMENSION:  Conformable with LBLRTM_Panel
                          ATTRIBUTES: INTENT(IN)

      n_Spectra:          Number of spectra (or "panels").
                          Must be > 0.
                          UNITS:      N/A
                          TYPE:       INTEGER
                          DIMENSION:  Conformable with LBLRTM_Panel
                          ATTRIBUTES: INTENT(IN)

OPTIONAL OUTPUTS:
      Err_Msg:            String containing error message text if allocation
                          failde.
                          UNITS:      N/A
                          TYPE:       CHARACTER(*)
                          DIMENSION:  Conformable with LBLRTM_Panel
                          ATTRIBUTES: INTENT(OUT), OPTIONAL
```

A.4.3 LBLRTM_Panel_DefineVersion interface

```
NAME:
      LBLRTM_Panel_DefineVersion

PURPOSE:
      Subroutine to return the version information for the
      definition module(s).

CALLING SEQUENCE:
      CALL LBLRTM_Panel_DefineVersion( Id )

OUTPUTS:
      Id:     Character string containing the version Id information for the
              definition module(s).
              UNITS:      N/A
              TYPE:       CHARACTER(*)
              DIMENSION:  Scalar
              ATTRIBUTES: INTENT(OUT)
```

A.4.4 LBLRTM_Panel_Destroy interface

```
NAME:
      LBLRTM_Panel_Destroy
```

```
PURPOSE:
      Elemental subroutine to re-initialize LBLRTM_Panel objects.

CALLING SEQUENCE:
      CALL LBLRTM_Panel_Destroy( LBLRTM_Panel )

OBJECTS:
      LBLRTM_Panel: Re-initialized LBLRTM_Panel instance.
                    UNITS:      N/A
                    TYPE:       LBLRTM_Panel_type
                    DIMENSION:  Scalar or any rank
                    ATTRIBUTES: INTENT(OUT)
```

A.4.5 LBLRTM_Panel_Inspect interface

```
NAME:
      LBLRTM_Panel_Inspect

PURPOSE:
      Subroutine to print the contents of an instance of an LBLRTM_Panel
      object to stdout.

CALLING SEQUENCE:
      CALL LBLRTM_Panel_Inspect( LBLRTM_Panel )

OBJECTS:
      LBLRTM_Panel:  LBLRTM_Panel object to display.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Panel_type
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN)
```

A.4.6 LBLRTM_Panel_IsValid interface

```
NAME:
      LBLRTM_Panel_IsValid

PURPOSE:
      Elemental function to test if the LBLRTM_Panel object contains
      valid data.

CALLING SEQUENCE:
      Status = LBLRTM_Panel_IsValid( LBLRTM_Panel )

OBJECTS:
      LBLRTM_Panel:  Instance which is to have its status tested.
```

```
                    UNITS:      N/A
                    TYPE:       LBLRTM_Panel_type
                    DIMENSION:  Scalar or any rank
                    ATTRIBUTES: INTENT(IN)

  FUNCTION RESULT:
        Status:        The return value is a logical value indicating
                       if the object contains valid data.
                       UNITS:      N/A
                       TYPE:       LOGICAL
                       DIMENSION:  Same as input
```

A.4.7 LBLRTM_Panel_SetValid interface

```
  NAME:
        LBLRTM_Panel_SetValid

  PURPOSE:
        Elemental subroutine to mark an instance of an LBLRTM_Panel object
        as containing valid data.

        Valid flag is set only if the LBLRTM_Panel object is allocated AND
        if the embedded LBLRTM_Phdr object is also valid.

  CALLING SEQUENCE:
        CALL LBLRTM_Panel_SetValid( LBLRTM_Panel )

  OBJECTS:
        LBLRTM_Panel:  Instance which is to have its validity set.
                       UNITS:      N/A
                       TYPE:       LBLRTM_Panel_type
                       DIMENSION:  Scalar or any rank
                       ATTRIBUTES: INTENT(IN)
```

A.4.8 LBLRTM_Panel_IOVersion interface

```
  NAME:
        LBLRTM_Panel_IOVersion

  PURPOSE:
        Subroutine to return the version information for the module.

  CALLING SEQUENCE:
        CALL LBLRTM_Panel_IOVersion( Id )

  OUTPUTS:
        Id:     Character string containing the version Id information for the
                module.
                UNITS:      N/A
                TYPE:       CHARACTER(*)
                DIMENSION:  Scalar
                ATTRIBUTES: INTENT(OUT)
```

A.4.9 LBLRTM_Panel_Read interface

```
NAME:
      LBLRTM_Panel_Read

PURPOSE:
      Function to read an LBLRTM panel from an LBLRTM format
      file.

CALLING SEQUENCE:
      Error_Status = LBLRTM_Panel_Read( &
                       LBLRTM_Panel, &
                       FileId      , &
                       EOF         , &
                       Double_Panel = Double_Panel, &
                       Quiet        = Quiet         )

OBJECTS:
      LBLRTM_Panel:  LBLRTM panel object to hold the data.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Panel_type
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(OUT)

INPUTS:
      FileId:        The unit number for the already open LBLRTM file.
                     UNITS:      N/A
                     TYPE:       INTEGER
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN)

OUTPUTS:
      EOF:           Integer flag indicating end-of-file status for the
                     LBLRTM format file after the read. Valid return values
                     are defined in the LBLRTM_Parameters module.
                     If == LBLRTM_FILE_EOF:   End-Of-File has been reached.
                                              The file is then closed.
                        == LBLRTM_FILE_OK:    No EOF or EOL condition. The
                                              next read should be the panel
                                              data.
                        == LBLRTM_FILE_UNDEF: An error occurred. The file is
                                              closed.
                     UNITS:      N/A
```

```
                    TYPE:       INTEGER
                    DIMENSION:  Scalar
                    ATTRIBUTES: INTENT(OUT)

OPTIONAL INPUTS:
      Double_Panel:  Set this logical flag to indicate a double-panel file.
                     If == .FALSE., the file is assumed to be single panel. [DEFAULT]
                        == .TRUE.,  the file is assumed to be double panel.
                     If not specified, default is .FALSE.
                     UNITS:      N/A
                     TYPE:       LOGICAL
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN), OPTIONAL

      Quiet:         Set this logical argument to suppress INFORMATION
                     messages being printed to stdout
                     If == .FALSE., INFORMATION messages are OUTPUT [DEFAULT].
                        == .TRUE.,  INFORMATION messages are SUPPRESSED.
                     If not specified, default is .FALSE.
                     UNITS:      N/A
                     TYPE:       LOGICAL
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:
      Error_Status:  The return value is an integer defining the error status.
                     The error codes are defined in the Message_Handler module.
                     If == SUCCESS the LBLRTM panel read was successful
                        == FAILURE an error occurred
                     UNITS:      N/A
                     TYPE:       INTEGER
                     DIMENSION:  Scalar

SIDE EFFECTS:
      If an error occurs or the end-of-file is encountered, the input file is
      closed.
```

A.4.10 LBLRTM_Panel_Write interface

```
NAME:
      LBLRTM_Panel_Write

PURPOSE:
      Function to write an LBLRTM panel to an LBLRTM format
      file.

CALLING SEQUENCE:
      Error_Status = LBLRTM_Panel_Write( &
                       LBLRTM_Panel, &
                       FileId      , &
                       Quiet = Quiet )

OBJECTS:
      LBLRTM_Panel:  LBLRTM panel object to write to file.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Panel_type
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN)

INPUTS:
      FileId:        The unit number for the already open LBLRTM file.
                     UNITS:      N/A
                     TYPE:       INTEGER
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN)

OPTIONAL INPUTS:
      Quiet:         Set this logical argument to suppress INFORMATION
                     messages being printed to stdout
                     If == .FALSE., INFORMATION messages are OUTPUT [DEFAULT].
                        == .TRUE.,  INFORMATION messages are SUPPRESSED.
                     If not specified, default is .FALSE.
                     UNITS:      N/A
                     TYPE:       LOGICAL
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:
      Error_Status:  The return value is an integer defining the error status.
                     The error codes are defined in the Message_Handler module.
                     If == SUCCESS the LBLRTM panel write was successful
                        == FAILURE an error occurred
                     UNITS:      N/A
                     TYPE:       INTEGER
                     DIMENSION:  Scalar

SIDE EFFECTS:
      If an error occurs, the output file is closed.
```

A.5 Layer Structure

```
TYPE :: LBLRTM_Layer_type
  ! Allocation and valid data indicator
  LOGICAL :: Is_Allocated = .FALSE.
  LOGICAL :: Is_Valid     = .FALSE.
  ! The panel header
  TYPE(LBLRTM_Fhdr_type) :: Header
  ! Dimensions
  INTEGER  :: n_Points  = 0 ! L
  INTEGER  :: n_Spectra = 0 ! N
  ! Frequency data
  REAL(DP) :: Begin_Frequency    = 0.0_DP
  REAL(DP) :: End_Frequency      = 0.0_DP
  REAL(FP) :: Frequency_Interval = 0.0_FP
  ! Spectral data
  REAL(FP), ALLOCATABLE  :: Spectrum(:,:)  ! L x N
END TYPE LBLRTM_Layer_type
```

Figure A.4: LBLRTM_Layer_type structure definition.

A.5.1 LBLRTM_Layer_Associated interface

```
NAME:
      LBLRTM_Layer_Associated

PURPOSE:
      Elemental function to test the status of the allocatable components
      of the LBLRTM_Layer object.

CALLING SEQUENCE:
      Status = LBLRTM_Layer_Associated( LBLRTM_Layer )

OBJECTS:
      LBLRTM_Layer:  Structure which is to have its member's
                     status tested.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Layer_type
                     DIMENSION:  Scalar or any rank
                     ATTRIBUTES: INTENT(IN)

FUNCTION RESULT:
      Status:        The return value is a logical value indicating if the
                     object has been allocated.
                     UNITS:      N/A
                     TYPE:       LOGICAL
                     DIMENSION:  Same as input
```

A.5.2 LBLRTM_Layer_Create interface

```
NAME:
      LBLRTM_Layer_Create

PURPOSE:
      Elemental subroutine to create an instance of an LBLRTM_Layer object.

CALLING SEQUENCE:
      CALL LBLRTM_Layer_Create( LBLRTM_Layer, &
                                LBLRTM_Fhdr , &
                                n_Spectra     )

OBJECTS:
      LBLRTM_Layer:       LBLRTM_Layer object structure.
                          UNITS:      N/A
                          TYPE:       LBLRTM_Layer_type
                          DIMENSION:  Scalar or any rank
                          ATTRIBUTES: INTENT(OUT)

INPUTS:
      LBLRTM_Fhdr:        LBLRTM_Fhdr object for the current Layer.
```

```
                         UNITS:      N/A
                         TYPE:       LBLRTM_Fhdr_type
                         DIMENSION:  Conformable with LBLRTM_Layer
                         ATTRIBUTES: INTENT(OUT)

      n_Spectra:         Number of spectra (or "panels").
                         Must be > 0.
                         UNITS:      N/A
                         TYPE:       INTEGER
                         DIMENSION:  Conformable with LBLRTM_Layer
                         ATTRIBUTES: INTENT(IN)
```

A.5.3 LBLRTM_Layer_DefineVersion interface

```
NAME:
      LBLRTM_Layer_DefineVersion

PURPOSE:
      Subroutine to return the version information for the
      definition module(s).

CALLING SEQUENCE:
      CALL LBLRTM_Layer_DefineVersion( Id )

OUTPUTS:
      Id:     Character string containing the version Id information for the
              definition module(s).
              UNITS:      N/A
              TYPE:       CHARACTER(*)
              DIMENSION:  Scalar
              ATTRIBUTES: INTENT(OUT)
```

A.5.4 LBLRTM_Layer_Destroy interface

```
NAME:
      LBLRTM_Layer_Destroy

PURPOSE:
      Elemental subroutine to re-initialize LBLRTM_Layer objects.

CALLING SEQUENCE:
      CALL LBLRTM_Layer_Destroy( LBLRTM_Layer )

OBJECTS:
      LBLRTM_Layer: Re-initialized LBLRTM_Layer instance.
                    UNITS:      N/A
                    TYPE:       LBLRTM_Layer_type
                    DIMENSION:  Scalar or any rank
                    ATTRIBUTES: INTENT(OUT)
```

A.5.5 LBLRTM_Layer_Frequency interface

```
NAME:
      LBLRTM_Layer_Frequency

PURPOSE:
      Subroutine to compute the frequency grid for a valid LBLRTM Layer object.

CALLING SEQUENCE:
      CALL LBLRTM_Layer_Frequency( LBLRTM_Layer, Frequency )

OBJECTS:
      LBLRTM_Layer:  LBLRTM_Layer object to display.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Layer_type
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN)

OUTPUTS:
      Frequency:     Frequency grid for the current LBLRTM Layer object.
                     UNITS:      Inverse centimetres (cm^-1)
                     TYPE:       REAL(DP)
                     DIMENSION:  Rank-1
                     ATTRIBUTES: INTENT(OUT), ALLOCATABLE
```

A.5.6 LBLRTM_Layer_Inspect interface

```
NAME:
      LBLRTM_Layer_Inspect

PURPOSE:
      Subroutine to print the contents of an instance of an LBLRTM_Layer
      object to stdout.

CALLING SEQUENCE:
      CALL LBLRTM_Layer_Inspect( LBLRTM_Layer )

OBJECTS:
      LBLRTM_Layer:  LBLRTM_Layer object to display.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Layer_type
                     DIMENSION:  Scalar
```

```
                        ATTRIBUTES: INTENT(IN)
```

A.5.7 LBLRTM_Layer_IsValid interface

```
NAME:
      LBLRTM_Layer_IsValid

PURPOSE:
      Elemental function to test if the LBLRTM_Layer object contains
      valid data.

CALLING SEQUENCE:
      Status = LBLRTM_Layer_IsValid( LBLRTM_Layer )

OBJECTS:
      LBLRTM_Layer:  Instance which is to have its status tested.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Layer_type
                     DIMENSION:  Scalar or any rank
                     ATTRIBUTES: INTENT(IN)

FUNCTION RESULT:
      Status:        The return value is a logical value indicating
                     if the object contains valid data.
                     UNITS:      N/A
                     TYPE:       LOGICAL
                     DIMENSION:  Same as input
```

A.5.8 LBLRTM_Layer_SetValid interface

```
NAME:
      LBLRTM_Layer_SetValid

PURPOSE:
      Elemental subroutine to mark an instance of an LBLRTM_Layer object
      as containing valid data.

      Valid flag is set only if the LBLRTM_Layer object is allocated AND
      if the embedded LBLRTM_Fhdr object is also valid.

CALLING SEQUENCE:
      CALL LBLRTM_Layer_SetValid( LBLRTM_Layer )

OBJECTS:
      LBLRTM_Layer:  Instance which is to have its validity set.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Layer_type
                     DIMENSION:  Scalar or any rank
                     ATTRIBUTES: INTENT(IN)
```

A.5.9 LBLRTM_Layer_IOVersion interface

```
NAME:
      LBLRTM_Layer_IOVersion

PURPOSE:
      Subroutine to return the version information for the module.

CALLING SEQUENCE:
      CALL LBLRTM_Layer_IOVersion( Id )

OUTPUTS:
      Id:     Character string containing the version Id information for the
              module.
              UNITS:      N/A
              TYPE:       CHARACTER(*)
              DIMENSION:  Scalar
              ATTRIBUTES: INTENT(OUT)
```

A.5.10 LBLRTM_Layer_Read interface

```
NAME:
      LBLRTM_Layer_Read

PURPOSE:
      Function to read an LBLRTM Layer from an LBLRTM format
      file.

CALLING SEQUENCE:
      Error_Status = LBLRTM_Layer_Read( &
                       LBLRTM_Layer, &
                       FileId      , &
                       EOF         , &
                       Double_Panel = Double_Panel, &
                       Quiet        = Quiet         )

OBJECTS:
      LBLRTM_Layer:  LBLRTM Layer object to hold the data.
                     UNITS:      N/A
                     TYPE:       LBLRTM_Layer_type
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(OUT)
```

```
INPUTS:
      FileId:        The unit number for the already open LBLRTM file.
                     UNITS:      N/A
                     TYPE:       INTEGER
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN)

OUTPUTS:
      EOF:           Integer flag indicating end-of-file status for the
                     LBLRTM format file after the read. Valid return values
                     are defined in the LBLRTM_Parameters module.
                     If == LBLRTM_FILE_EOF:   End-Of-File has been reached.
                                              The file is then closed.
                        == LBLRTM_FILE_OK:    No EOF or EOL condition. The
                                              next read should be the panel
                                              data.
                        == LBLRTM_FILE_UNDEF: An error occurred. The file is
                                              closed.
                     UNITS:      N/A
                     TYPE:       INTEGER
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(OUT)

OPTIONAL INPUTS:
      Double_Panel:  Set this logical argument to indicate a double-panel file.
                     If == .FALSE., the file is assumed to be single panel [DEFAULT].
                        == .TRUE.,  the file is assumed to be double panel.
                     If not specified, default is .FALSE.
                     UNITS:      N/A
                     TYPE:       LOGICAL
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN), OPTIONAL

      Quiet:         Set this logical flag to suppress INFORMATION
                     messages being printed to stdout
                     If == .FALSE., INFORMATION messages are OUTPUT [DEFAULT].
                        == .TRUE.,  INFORMATION messages are SUPPRESSED.
                     If not specified, default is .FALSE.
                     UNITS:      N/A
                     TYPE:       LOGICAL
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:
      Error_Status:  The return value is an integer defining the error status.
                     The error codes are defined in the Message_Handler module.
                     If == SUCCESS the LBLRTM layer read was successful
                        == FAILURE an error occurred
                     UNITS:      N/A
                     TYPE:       INTEGER
                     DIMENSION:  Scalar

SIDE EFFECTS:
      If an error occurs or the end-of-file is encountered, the input file is
      closed.
```

A.5.11 LBLRTM_Layer_Write interface

```
  NAME:
        LBLRTM_Layer_Write

  PURPOSE:
        Function to write an LBLRTM Layer to an LBLRTM format
        file.

  CALLING SEQUENCE:
        Error_Status = LBLRTM_Layer_Write( &
                         LBLRTM_Layer, &
                         FileId      , &
                         No_EoL = No_EoL, &
                         Quiet  = Quiet   )

  OBJECTS:
        LBLRTM_Layer:  LBLRTM Layer object to write to file.
                       UNITS:      N/A
                       TYPE:       LBLRTM_Layer_type
                       DIMENSION:  Scalar
                       ATTRIBUTES: INTENT(IN)

  INPUTS:
        FileId:        The unit number for the already open LBLRTM file.
                       UNITS:      N/A
                       TYPE:       INTEGER
                       DIMENSION:  Scalar
                       ATTRIBUTES: INTENT(IN)

  OPTIONAL INPUTS:
        No_EoL:        Set this logical flag to indicate an End-Of-Level (EoL)
                       marker should NOT be written to the output LBLRTM file.
                       If == .FALSE., an EoL marker is written [DEFAULT]
                          == .TRUE.,  an EoL marker is NOT written
                       If not specified, default is .FALSE.
                       UNITS:      N/A
                       TYPE:       LOGICAL
                       DIMENSION:  Scalar
                       ATTRIBUTES: INTENT(IN), OPTIONAL

        Quiet:         Set this logical argument to suppress INFORMATION
                       messages being printed to stdout
                       If == .FALSE., INFORMATION messages are OUTPUT [DEFAULT].
                          == .TRUE.,  INFORMATION messages are SUPPRESSED.
                       If not specified, default is .FALSE.
                       UNITS:      N/A
```

```
                    TYPE:       LOGICAL
                    DIMENSION:  Scalar
                    ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:
      Error_Status: The return value is an integer defining the error status.
                    The error codes are defined in the Message_Handler module.
                    If == SUCCESS the LBLRTM layer write was successful
                       == FAILURE an error occurred
                    UNITS:      N/A
                    TYPE:       INTEGER
                    DIMENSION:  Scalar

SIDE EFFECTS:
      If an error occurs, the output file is closed.
```

A.6 `File` Structure

```
TYPE :: LBLRTM_File_type
  ! Allocation and valid data indicator
  LOGICAL :: Is_Allocated = .FALSE.
  LOGICAL :: Is_Valid     = .FALSE.
  ! Dimensions
  INTEGER  :: n_Layers  = 0 ! K
  ! The layer data
  TYPE(LBLRTM_Layer_type), ALLOCATABLE :: Layer(:)
END TYPE LBLRTM_File_type
```

Figure A.5: LBLRTM_File_type structure definition.

A.6.1 LBLRTM_File_Associated interface

```
NAME:
      LBLRTM_File_Associated

PURPOSE:
      Elemental function to test the status of the allocatable components
      of the LBLRTM_File object.

CALLING SEQUENCE:
      Status = LBLRTM_File_Associated( LBLRTM_File )

OBJECTS:
      LBLRTM_File:   Structure which is to have its member's
                     status tested.
                     UNITS:      N/A
                     TYPE:       LBLRTM_File_type
                     DIMENSION:  Scalar or any rank
                     ATTRIBUTES: INTENT(IN)

FUNCTION RESULT:
      Status:        The return value is a logical value indicating if the
                     object has been allocated.
                     UNITS:      N/A
                     TYPE:       LOGICAL
                     DIMENSION:  Same as input
```

A.6.2 LBLRTM_File_Create interface

```
NAME:
      LBLRTM_File_Create

PURPOSE:
      Elemental subroutine to create an instance of an LBLRTM_File object.

CALLING SEQUENCE:
      CALL LBLRTM_File_Create( LBLRTM_File, &
                               n_Layers     )

OBJECTS:
      LBLRTM_File:        LBLRTM_File object structure.
                          UNITS:      N/A
                          TYPE:       LBLRTM_File_type
                          DIMENSION:  Scalar or any rank
                          ATTRIBUTES: INTENT(OUT)

INPUTS:
      n_Layers:           Number of layers of spectral data.
                          Must be > 0.
```

```
                              UNITS:      N/A
                              TYPE:       INTEGER
                              DIMENSION:  Conformable with LBLRTM_File
                              ATTRIBUTES: INTENT(IN)
```

A.6.3 LBLRTM_File_DefineVersion interface

```
NAME:
      LBLRTM_File_DefineVersion

PURPOSE:
      Subroutine to return the version information for the
      definition module(s).

CALLING SEQUENCE:
      CALL LBLRTM_File_DefineVersion( Id )

OUTPUTS:
      Id:     Character string containing the version Id information for the
              definition module(s).
              UNITS:      N/A
              TYPE:       CHARACTER(*)
              DIMENSION:  Scalar
              ATTRIBUTES: INTENT(OUT)
```

A.6.4 LBLRTM_File_Destroy interface

```
NAME:
      LBLRTM_File_Destroy

PURPOSE:
      Elemental subroutine to re-initialize LBLRTM_File objects.

CALLING SEQUENCE:
      CALL LBLRTM_File_Destroy( LBLRTM_File )

OBJECTS:
      LBLRTM_File: Re-initialized LBLRTM_File instance.
                   UNITS:      N/A
                   TYPE:       LBLRTM_File_type
                   DIMENSION:  Scalar or any rank
                   ATTRIBUTES: INTENT(OUT)
```

A.6.5 LBLRTM_File_Inspect interface

```
NAME:
      LBLRTM_File_Inspect

PURPOSE:
      Subroutine to print the contents of an instance of an LBLRTM_File
      object to stdout.

CALLING SEQUENCE:
      CALL LBLRTM_File_Inspect( LBLRTM_File )

OBJECTS:
      LBLRTM_File:  LBLRTM_File object to display.
                     UNITS:      N/A
                     TYPE:       LBLRTM_File_type
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN)
```

A.6.6 LBLRTM_File_IsValid interface

```
NAME:
      LBLRTM_File_IsValid

PURPOSE:
      Elemental function to test if the LBLRTM_File object contains
      valid data.

CALLING SEQUENCE:
      Status = LBLRTM_File_IsValid( LBLRTM_File )

OBJECTS:
      LBLRTM_File:  Instance which is to have its status tested.
                     UNITS:      N/A
                     TYPE:       LBLRTM_File_type
                     DIMENSION:  Scalar or any rank
                     ATTRIBUTES: INTENT(IN)

FUNCTION RESULT:
      Status:        The return value is a logical value indicating
                     if the object contains valid data.
                     UNITS:      N/A
                     TYPE:       LOGICAL
                     DIMENSION:  Same as input
```

A.6.7 LBLRTM_File_SetValid interface

```
NAME:
      LBLRTM_File_SetValid

PURPOSE:
      Elemental subroutine to mark an instance of an LBLRTM_File object
      as containing valid data.

      Valid flag is set only if the LBLRTM_File object is allocated AND
      if the embedded LBLRTM_Layer object array is also valid.

CALLING SEQUENCE:
      CALL LBLRTM_File_SetValid( LBLRTM_File )

OBJECTS:
      LBLRTM_File:   Instance which is to have its validity set.
                     UNITS:      N/A
                     TYPE:       LBLRTM_File_type
                     DIMENSION:  Scalar or any rank
                     ATTRIBUTES: INTENT(IN)
```

A.6.8 LBLRTM_File_IOVersion interface

```
NAME:
      LBLRTM_File_IOVersion

PURPOSE:
      Subroutine to return the version information for the module.

CALLING SEQUENCE:
      CALL LBLRTM_File_IOVersion( Id )

OUTPUTS:
      Id:     Character string containing the version Id information for the
              module.
              UNITS:      N/A
              TYPE:       CHARACTER(*)
              DIMENSION:  Scalar
              ATTRIBUTES: INTENT(OUT)
```

A.6.9 LBLRTM_File_Read interface

```
NAME:
      LBLRTM_File_Read
```

```
PURPOSE:
      Function to read an LBLRTM format file.

CALLING SEQUENCE:
      Error_Status = LBLRTM_File_Read( &
                       LBLRTM_File , &
                       Filename    , &
                       n_Layers     = n_Layers    , &
                       Double_Panel = Double_Panel, &
                       Quiet        = Quiet        )

OBJECTS:
      LBLRTM_File:   LBLRTM File object to hold the data.
                     UNITS:      N/A
                     TYPE:       LBLRTM_File_type
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(OUT)

INPUTS:
      Filename:      The name of the LBLRTM file to read
                     UNITS:      N/A
                     TYPE:       CHARACTER(*)
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN)

OPTIONAL INPUTS:
      n_Layers:      Number of layers of spectral data to read.
                     If not specified, the number of layers read is 1.
                     UNITS:      N/A
                     TYPE:       INTEGER
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN), OPTIONAL

      Double_Panel:  Set this logical argument to indicate a double-panel file.
                     If == .FALSE., the file is assumed to be single panel. [DEFAULT]
                        == .TRUE.,  the file is assumed to be double panel.
                     If not specified, default is .FALSE.
                     UNITS:      N/A
                     TYPE:       LOGICAL
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN), OPTIONAL

      Quiet:         Set this logical argument to suppress INFORMATION
                     messages being printed to stdout
                     If == .FALSE., INFORMATION messages are OUTPUT [DEFAULT].
                        == .TRUE.,  INFORMATION messages are SUPPRESSED.
                     If not specified, default is .FALSE.
                     UNITS:      N/A
                     TYPE:       LOGICAL
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:
      Error_Status:  The return value is an integer defining the error status.
                     The error codes are defined in the Message_Handler module.
                     If == SUCCESS the LBLRTM file read was successful
                        == FAILURE an error occurred
                     UNITS:      N/A
                     TYPE:       INTEGER
                     DIMENSION:  Scalar
```

A.6.10 LBLRTM_File_Write interface

```
  NAME:
      LBLRTM_File_Write

  PURPOSE:
      Function to write an LBLRTM format file.

  CALLING SEQUENCE:
      Error_Status = LBLRTM_File_Write( &
                       LBLRTM_File , &
                       Filename    , &
                       Quiet = Quiet )

  OBJECTS:
      LBLRTM_File:   LBLRTM File object to write to file.
                     UNITS:      N/A
                     TYPE:       LBLRTM_File_type
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN)

  INPUTS:
      Filename:      The name of the LBLRTM file to write.
                     UNITS:      N/A
                     TYPE:       CHARACTER(*)
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN)

  OPTIONAL INPUTS:
      Quiet:         Set this logical argument to suppress INFORMATION
                     messages being printed to stdout
                     If == .FALSE., INFORMATION messages are OUTPUT [DEFAULT].
                        == .TRUE.,  INFORMATION messages are SUPPRESSED.
                     If not specified, default is .FALSE.
                     UNITS:      N/A
                     TYPE:       LOGICAL
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN), OPTIONAL

  FUNCTION RESULT:
      Error_Status:  The return value is an integer defining the error status.
                     The error codes are defined in the Message_Handler module.
```

```
If == SUCCESS the LBLRTM file write was successful
   == FAILURE an error occurred
UNITS:      N/A
TYPE:       INTEGER
DIMENSION:  Scalar
```

A.7 `LBLRTM_Utility` Module

A.7.1 LBLRTM_EoF_Message interface

```
NAME:
      LBLRTM_EoF_Message

PURPOSE:
      Pure function to return a message describing the LBLRTM file EOF status.

CALLING SEQUENCE:
      msg = LBLRTM_EoF_Message( eof )

INPUTS:
      eof:       LBLRTM file EoF status specifier.
                 UNITS:      N/A
                 TYPE:       INTEGER
                 DIMENSION:  Scalar
                 ATTRIBUTES: INTENT(IN)

FUNCTION RESULT:
      msg:       Character string describing the LBLRTM file EoF status.
                 UNITS:      N/A
                 TYPE:       CHARACTER(*)
                 DIMENSION:  Scalar
```

A.7.2 LBLRTM_EoL_Write interface

```
NAME:
      LBLRTM_EoL_Write

PURPOSE:
      Function to write an end-of-layer (EoL) marker to an output
      LBLRTM format file.

CALLING SEQUENCE:
      Error_Status = LBLRTM_EoL_Write( FileID )

INPUTS:
      FileId:        The unit number for the already open LBLRTM file.
                     UNITS:      N/A
                     TYPE:       INTEGER
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN)

FUNCTION RESULT:
      Error_Status: The return value is an integer defining the error status.
                    The error codes are defined in the Message_Handler module.
                    If == SUCCESS the LBLRTM file EOL write was successful
```

```
                         == FAILURE an unrecoverable error occurred
                      UNITS:      N/A
                      TYPE:       INTEGER
                      DIMENSION:  Scalar

SIDE EFFECTS:
      If an error occurs writing to the file, it is closed.
```

A.7.3 LBLRTM_File_Open interface

```
NAME:
      LBLRTM_File_Open

PURPOSE:
      Function to open an LBLRTM format data file. If the file is opened for
      reading, then a check is performed to determine if the file is of the
      right byte-sex.

CALLING SEQUENCE:
      Error_Status = LBLRTM_File_Open( &
                       Filename, &
                       FileId  , &
                       For_Output = For_Output, &
                       Quiet      = Quiet       )

INPUTS:
      FileName:      The LBLRTM formay datafile to open.
                     UNITS:      N/A
                     TYPE:       INTEGER
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN)

OUTPUTS:
      FileId:        The unit number for file access.
                     UNITS:      N/A
                     TYPE:       INTEGER
                     DIMENSION:  Scalar
                     ATTRIBUTES: INTENT(IN)

OPTIONAL INPUTS:
      For_Output:    Set this logical argument to open a new file for
                     writing. Default action is to open an existing file
                     for read access.
                     If == .FALSE., existing file is opened for READ access [DEFAULT]
                                   ACTION='READ', STATUS='OLD'
                        == .TRUE. , new file is opened for WRITE access.
                                   ACTION='WRITE', STATUS='REPLACE'
                     If not specified, the default is .FALSE.
                     NOTE: If the file already exists and it is opened with
                           this keyword set to .TRUE., the file is OVERWRITTEN.
                     UNITS:      N/A
```

```
                  TYPE:       LOGICAL
                  DIMENSION:  Scalar
                  ATTRIBUTES: INTENT(IN), OPTIONAL

      Quiet:      Set this logical argument to suppress INFORMATION
                  messages being printed to stdout
                  If == .FALSE., INFORMATION messages are OUTPUT [DEFAULT].
                     == .TRUE.,  INFORMATION messages are SUPPRESSED.
                  If not specified, default is .FALSE.
                  UNITS:      N/A
                  TYPE:       LOGICAL
                  DIMENSION:  Scalar
                  ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:
      Error_Status: The return value is an integer defining the error status.
                    The error codes are defined in the Message_Handler module.
                    If == SUCCESS the LBLRTM file open was successful
                       == FAILURE an unrecoverable error occurred
                    UNITS:      N/A
                    TYPE:       INTEGER
                    DIMENSION:  Scalar
```

A.7.4 LBLRTM_UtilityVersion interface

```
NAME:
      LBLRTM_UtilityVersion

PURPOSE:
      Subroutine to return the version information for the module.

CALLING SEQUENCE:
      CALL LBLRTM_UtilityVersion( Id )

OUTPUTS:
      Id:     Character string containing the version Id information for the
              module.
              UNITS:      N/A
              TYPE:       CHARACTER(*)
              DIMENSION:  Scalar
              ATTRIBUTES: INTENT(OUT)
```

A.7.5 LBLRTM_n_Points interface

```
NAME:
      LBLRTM_n_Points
```

```
PURPOSE:
      Pure function to compute the number of points in an LBLRTM spectrum.

CALLING SEQUENCE:
      n_Points = LBLRTM_n_Points( f1, f2, df )

INPUTS:
      f1:        Beginning frequency of the spectral data.
                 UNITS:      cm^-1
                 TYPE:       REAL(DP)
                 DIMENSION:  Scalar
                 ATTRIBUTES: INTENT(IN)

      f2:        Ending frequency of the spectral data.
                 UNITS:      cm^-1
                 TYPE:       REAL(DP)
                 DIMENSION:  Scalar
                 ATTRIBUTES: INTENT(IN)

      df:        Frequency spacing of the spectral data.
                 The value of the data type kind, FP, is can
                 indicate either single or double precision
                 depending on how the Type_Kinds.fpp module was
                 preprocessed for compilation.
                 UNITS:      cm^-1
                 TYPE:       REAL(FP)
                 DIMENSION:  Scalar
                 ATTRIBUTES: INTENT(IN)

FUNCTION RESULT:
      n_Points:  The return value is an integer containing the
                 number of points in the spectrum.
                 UNITS:      N/A
                 TYPE:       INTEGER
                 DIMENSION:  Scalar

PROCEDURE:
      The number of points is calculated from the begin and end frequencies,
      f1 and f2, and the frequency interval, df, by:

                  ( f2 - f1        )
        n = FLOOR (--------- + 1.5 )
                  (   df           )
```