Study:

The Scientific Analysis of Limb Sounding Observations of the Upper Troposphere

MARSCHALS Level 2
Software User and Reference Manual

Issue 3 – Revision 1

10 April 2007

Prepared by:

Name	Institute
G. Bazzini	ASPER
B. Carli	IFAC-CNR
C. Cecchi Pestellini	IFAC-CNR
S. Del Bianco	ASPER
B.M. Dinelli	ISAC-CNR
M. Gai	ASPER
L. Santurri	IFAC-CNR

For the comments of:

Name	Institute
S. Baronti	IFAC-CNR

This page is intentionally left blank.

CHANGE RECORD			
ISSUE	*DATE*		*REASON FOR CHANGE AND AFFECTED SECTIONS*
0.1	15/01/2004	First internal draft	First temporary draft.
0.2	10/08/2004	Second internal draft	The whole document has been changed according to a new software arrangement.
1.0	10/08/2004	First issue	Revision according to some internal comments.
2.0	10/05/2006	Second issue	Full revision and arrangement according to substantial changes in the code.
3.0	10/04/2007	Third issue	Revision according to some changes in the code.
3.1	10/04/2007	Forth issue	Some minor changes and corrections following MARC developer team observations.

This page is intentionally left blank.

TABLE OF CONTENTS

This page is intentionally left blank.

1 Introduction

1.1 Purpose

The MARSCHALS L2 software is a suite of three codes developed for a **Linux platform** and devoted to the retrieval of the atmospheric constituents from a set of spectroscopic measurements coming from the Level 1b (L1b) of MARSCHALS project. Together with the retrieved profiles, also some auxiliary results are provided by the MARSCHALS Level 2 (L2) software.

The aim of this document is to provide the information needed to install and use the software (MARSCHALS Level 2 software) that has been developed in the framework of MARSCHALS L2 Project.
Please notices that the MARSCHALS Level 2 software is a suite composed by three codes, that have to be run in a sequence; this document is devoted to the description of the use of the overall suite and of all the single codes.

1.2 How to read this document

In the following chapters of this document information about the installation, the setting and the execution of the codes composing the MARSCHALS L2 software are given. In Chapter (0) the list of the applicable and reference documents is presented, together with the acronyms and the conventions used in this document. In Chapter 2 an overview of the MARSCHALS L2 software is provided; the list of the source files is also presented in Paragraph 2.4, with the whole list of the files the code needs of to perform the retrieval procedure. Chapter 3 is devoted to the description of the codes installation procedures, with the specific hardware and software requirements. A detailed description about the use of the codes is then given in chapter 4, 5 and 6; in these chapters the parameters setting opportunities are also explained. In chapter 7 a tool provided to visualize the results of the retrieval procedure id described.
The run of the MARSCHALS L2 software requires also the execution of the external (not developed by the MARSCHALS L2 team) SBDART code. In annex 1 (chapter 8) the original documentation of SBDART code is reported.

Please notices that, according to the contract, the MARSCHALS L2 software is provided to ESA in electronic form by means of a computer in which the source files are stored and compiled.
Thus, together with the source files, also an executable version of the software the user can directly run is provided. To use this executable version, it is not needed any installation and compilation procedure; in this case the user can skip the reading of the paragraph 2.2, 2.3 and 2.4 and the whole Chapter 3 (related to the installation) and the paragraph 4.1, 5.1 and 6.1 (related to the compilation of the codes).

1.3 Relevant Document

RD 1	MARSCHALS Level 1 Product Format Specification (Issue 1 rel. 3, 29/09/2003)
RD 2	MARSCHALS Level 2 Algorithm Theoretical Baseline Document (IFAC_GA_2007_01_LS), issue 7 rel. 1, S. Baronti et al., 15 January 2007
RD 3	MARSCHALS Level 2 Architectural Design Document (IFAC_GA_2007_03_LS, issue 4 rel. 1, S. Baronti et al., 25 March 2007)
RD 4	MARSCHALS Level 2 Theoretical Retrieval Study (IFAC_GA_2007_05_LS, issue 5 rel. 1, G.Bazzini et al., 05 April 2007)
RD 5	Characterization of Millimetre-Wave Spectroscopic Signatures, ESTEC Contract No 16377/02/NL/FF
RD 6	Ricchiazzi et al 1998. (Bulletin of the American Meteorological Society, October 1998)

1.4 Acronyms

Acronyms list	
ADD	**A**dvanced **D**etailed **D**escription
ATBD	MARSCHALS Level 2 **A**lgorithm **T**heoretical **B**aseline **D**ocument
ESA	**E**uropean **S**pace **A**gency
IDL	**I**nteractive **D**ata **L**anguage
FORTRAN	**FOR**mula **TRAN**slation
L1b	**L**evel **1 b** (of MARSCHALS project)
L2	**L**evel **2** (of MARSCHALS project)
MARC	**M**illimetre-wave **A**tmospheric **R**etrieval **C**ode
MARSCHALS	**M**illimetre wave **A**irborne **R**eceivers for **S**pectroscopic **CH**aracterization of **A**tmospheric **L**imb **S**ounding
MFM	**M**arschals **F**orward **M**odel
MSSF	**M**ie **S**cattering **S**ource **F**unction
OCM	**O**ptical **C**loud **M**onitor
OFM	**O**CM **F**orward **M**odel
OSM	**O**ptical **S**ounding **M**odule
RTM	**R**adiative **T**ransfer **M**odel
SAMM	**S**upervising **A**nalyzer of **M**ARSCHALS **M**easurements
SBDART	**S**anta **B**arbara **DI**SORT **A**tmospheric **R**adiative **T**ransfer
VMR	**V**olume **M**ixing **R**atio

1.5 Conventions

General Identifier String	
§	*identifier indicating a generic MARSCHALS scan number*
β	*identifier indicating a generic string among the three band names (band_B, band_C, band_D)*
$	*identifier indicating a generic MARSCHALS band (a letter among the band letters (B,C,D)*
%	*identifier indicating a generic flight (campaign) of the instrument*
target	*identifier indicating a generic retrieved quantity*
species	*identifier indicating a generic molecule symbol of a retrieved species (e.g. H_2O)*
*****	*identifier indicating a generic namefile without extension*

This documentation uses the following general conventions:

Italic Font
: Denotes the name of files, directories and subdirectories, codes or modules of codes.

Bold Font
: Represents menus, windows names and tool buttons.

Bold Italic Font
: Denotes book titles.

`Plain Typewriter Font`
: Denotes code fragments, command lines, contents of files and command names.

`Italic Typewriter Font`
: Represents a variable for which an actual value should be substituted.

Paths are in UNIX notation (Forward Slashes).

This page is intentionally left blank.

2 Overview of MARSCHALS L2 software

The MARSCHALS L2 software is a suite of codes oriented to the retrieval of the atmospheric constituents from a set of spectroscopic measurements. These measurements come from the L1b of MARSCHALS project and different elaborations are necessary in order to retrieve all the requested profiles, such as: a preliminary treatment of measurement data, the simulation of synthetic spectra, the iterative execution of an "inversion algorithm", a modeling of scattering contribution due to clouds, both in millimetre region and in near-infrared spectral range.
A set of modules have been thus developed, and arranged in executable codes.

2.1 Software modules and codes

The MARSCHALS L2 software is basically constituted by five modules:

- *Forward Model* module;
- *Retrieval* module;
- *Scattering Source Function* module;
- *Pre-Processor* module;
- *Optical Sounding* module.

- The *Forward Model* module is dedicated to the simulation of synthetic spectra, at a given atmospheric state; it is devoted to the computation of a whole set of spectroscopic limb measurements (as they would be acquired by the MARSCHALS instrument) starting from a given profile of atmospheric constituents, pressure and Temperature, and starting from a model of the instrument itself;

- the *Retrieval* module is devoted to the retrieval of the profile of atmospheric constituents, by processing a set of spectroscopic measured (or simulated) data;

- the *Scattering Source Function* module is dedicated to the computation of additional Source Function, due to the presence of aerosols;

- the *Pre-Processor* module is dedicated to a preliminary overview of measurement data, and to rearrange the data themselves in a suitable format;

- an *Optical Sounding* module is dedicated to the simulation of radiance, as it is received by OCM instrument.

The five modules are integrated in three different codes:

- *SAMM* code;
- *MARC* code;
- *OFM* codes;

- the SAMM code, whose name stands for ***S**upervising **A**nalyzer of **M**ARSCHALS **M**easurements* reads the L1b data and arranges measurement data in the proper format requested by MARC code; it works as a pre-processor of MARC, providing also some auxiliary data. It constitutes the Pre-Processor Module;

- the MARC code, whose name stands for ***M**illimetre-wave **A**tmospheric-**R**etrieval **C**ode* is based on the *Retrieval* module, and performs the retrieval procedure. Actually the retrieval procedure uses the *Forward Model* module (performed by the *MFM – MARSCHALS Forward Model* routine), to generate the synthetic spectra necessary to minimize the residuals; in case of retrieval on a "cloud-contamined" measurement, the *Scattering Source Function* module (performed by the *MSSF – Mie Scattering Source Function* routine) is also called;

- the OFM codes, whose name stands for ***O**CM **F**orward **M**odel,* performs a simulation of the radiance collected by the OCM instrument, and constitutes the *Optical Sounding* module.

The MARC code is the principal code of the MARSCHALS L2 suite, and performs the retrieval procedure on the data pre-processed by the SAMM code; it uses the *Retrieval* module and the *Forward* module. When necessary it uses also the results of the OFM code and the auxiliary *MSSF* module.

A *Visualization Tool* is also provided in order to plot the results of the retrieval.

The Figure 2-1 provides an overview of the developed modules, suggesting how these modules have been integrated to generate the MARSCHALS codes.

A retrieval procedure is performed by running the SAMM code on the L1b data, and then the MARC code on the output of the SAMM execution; in some conditions, also the execution of the OFM code is needed.

To compile and execute the SAMM code and the MARC code see Chapter 4 and Chapter 5 respectively. The compilation and execution of the OFM code is described in Chapter 6.

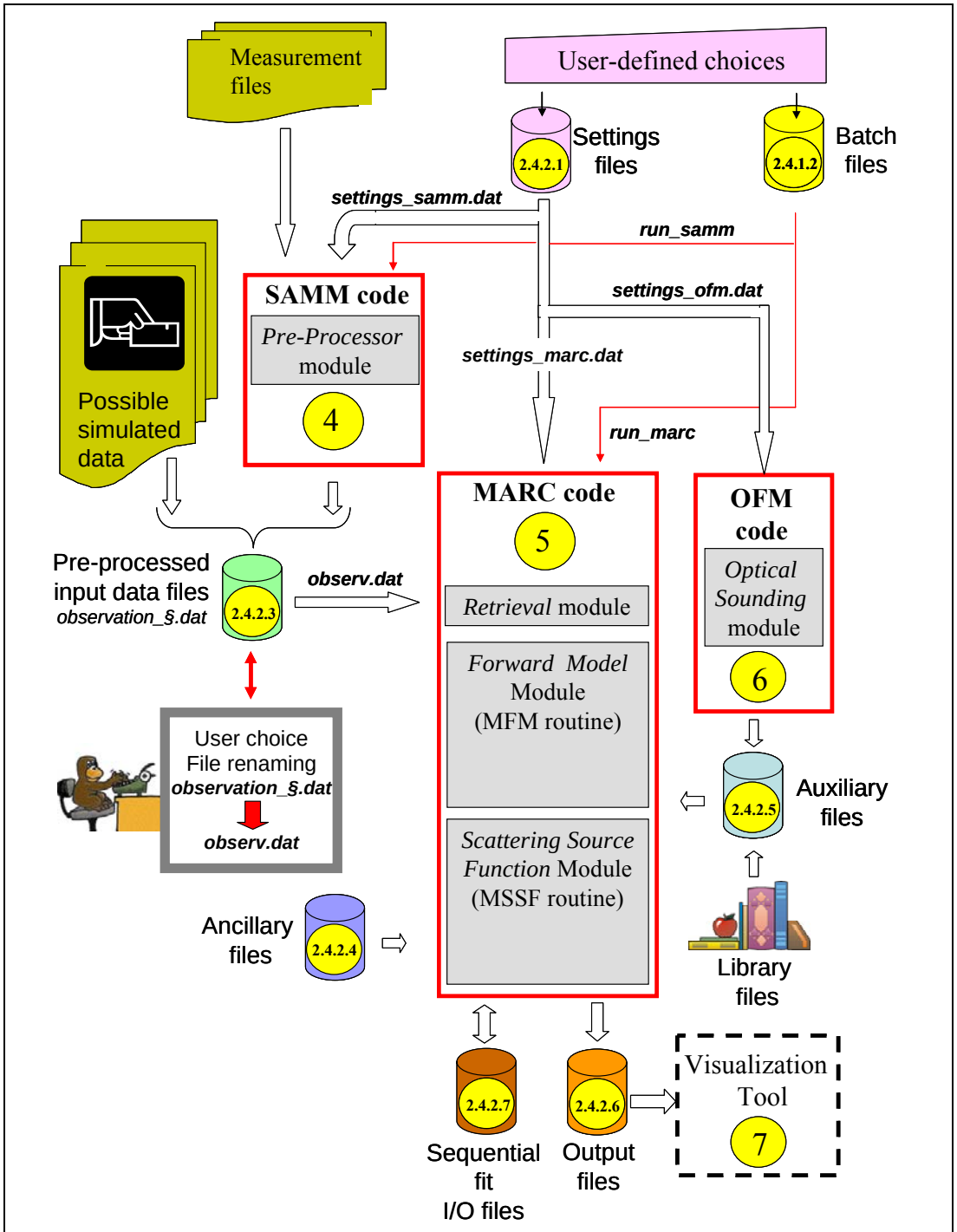

Figure 2-1. The overall scheme of the retrieval procedure performed by the suite of MARSCHALS L2 codes; grey boxes represent single modules, whereas red contoured boxes indicate the codes. The numbers indicates the section of this document in which the related code, module or files are described.

As previously noticed, the aim of the MARSHALS L2 developed software is to provide a retrieval of the atmospheric profile, that is the output of the MARC code; thus the simulated measurement produced by the forward model during the retrieval procedure would not be required as an output. Anyway, in order to better perform the test on the adopted forward model, the aforementioned simulated measurement can be obtained as an output and saved on a file by adequately setting the MARC configuration parameters; in this case the MARC code acts as a Forward Model, executing only the part of the code related to the *Forward Model* module. The setting of the MARC configuration is described in 5.4.

2.2 The used programming language

SAMM code has been developed in standard C language, whereas MARC and OFM codes are written in FORTRAN 90. About MARC and OFM codes the developers followed a given (FORTRAN 90 compatible) syntax in order to make possible the use of a tool for the automatic documentation of such codes.
The visualization tool is written in IDL language.

2.3 Hardware and system requirement

The MARSCHALS L2 codes have been developed in order to run under a LINUX operating system.

The MARSCHALS L2 codes have been also tested on a Windows platform. To run under a Microsoft Windows Operating System, some changes have to be made in the sources of the code, in order to take into account for the different system variables.
The description of these changes is out of the purpose of this document.

Some requirements have been identified for the computer devoted to run the MARSCHALS L2 codes.
The only mandatory requirement concerns the RAM memory that has to be not less then 2 Gbyte. Such an amount of RAM is the maximum amount of memory that a Linux system on a 32-bit machine can efficiently manage. The 2 Gbyte RAM should guarantee that paging (system swapping on disk) is minimized thus allowing for the maximum speed when running the code. An important suggestion concerns the clock speed. Due to the massive amount of computations such a speed has to be as high as possible (>3 GHz). The other hardware requirements (disk, CD-writer, network board, display, graphic card) are non-critical.

HARDWARE requirements:
RAM > 2 Gbyte
Processor Speed > 3 Ghz*

* Suggested

As an example, the MARSCHALS codes has been executed successfully on a computer with the following characteristics:

Processor: INTEL Pentium IV
Memory: 2 Gbyte RAM
Operating System: Linux DEBIAN 2.4.23

At the present the code works under a LINUX operating system. A Windows version is also available; the two versions differ only for some commands inside some code routines.

2.4 The MARSCHALS L2 software files

In this chapter, the set of the files needed to run the MARSCHALS L2 codes is introduced.

The executable version on the computer delivered to ESA is provided with all the files needed to run the codes.

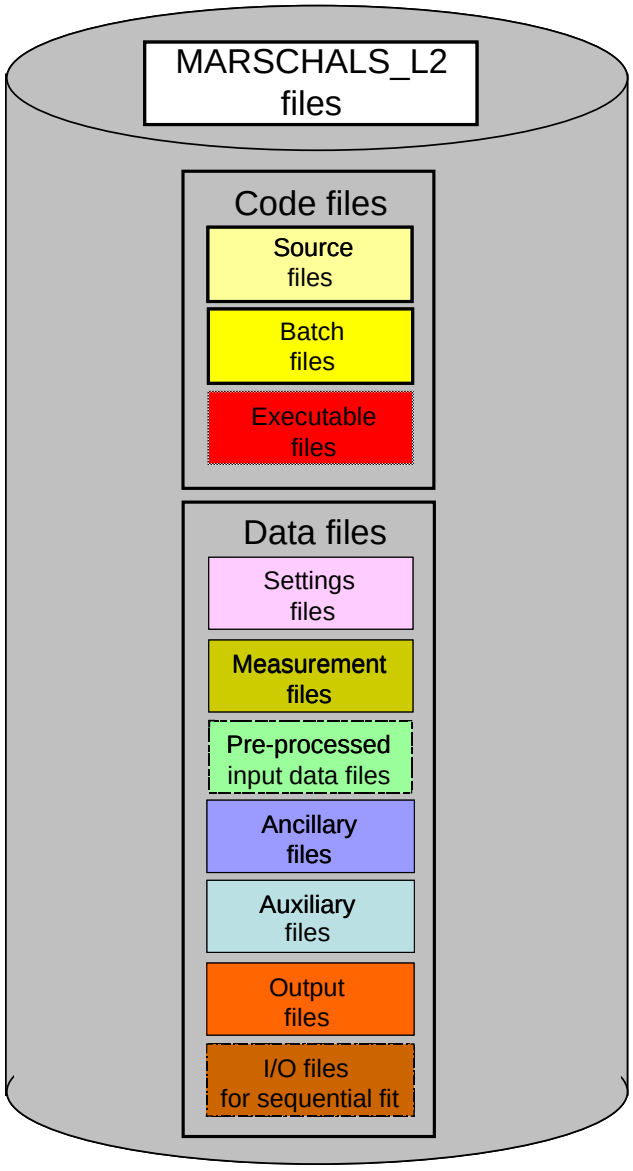

Figure 2-2. The files needed to run the MARSCHALS L2 codes.

As specified in Section 2.1, the MARSCHALS L2 suite is composed by three codes; each of thise is provided, in source version, by mean of a set of source files. Among the code files, also some batch files are provided to help the users in compiling and running the codes.
An executable version of the code (i.e. already set up and compiled) is also provided; trivially, the executable version is not "strictly" needed to run the code, since it can be originated by compiling the source codes. On the opposite hand, the source files are not requested if the executable version is used.
Each code operates on data stored in different data files. Trivially, to run the codes the user needs both the source files to be compiled and the adequate input data files.
The Pre-processed input data files are needed to run the MARC code, but, as for the executable files, they are not strictly needed because they can be originated by a run of the SAMM code on the measurement files.
The files needed to run the codes are briefly presented in the following Paragraphs, where files are introduced grouped according to their role in the execution of the codes; this functional subdivision of the MARSCHALS L2 files is schematized in Figure 2-2. A detailed list of the files needed to run the code is reported in Chapter 3.
As shown in Figure 2-2, the MARSCHALS suite is based on some code files, i.e. on a set of source and batch files; these source files are introduced in Paragraph 2.4.1. MARSCHALS L2 codes require some setting and parameters files (see 2.4.2); the data coming from the MARSCHALS measurements are trivially requested, and provided by the L1b measurement file (see 2.4.2.2). Furthermore, a set of adequate data files coming from the pre-processing of a L1b data file are requested as input file of the MARC code (see 2.4.2.3). A set of auxiliary files with auxiliary data (e.g. meteorological data) is also requested (see 2.4.2.5).

2.4.1 The code files

The set of code files is composed by the source files (see 2.4.1.1), to be compiled to generate the executable files, three makefiles (one for each code) that can be used to compile the codes (see 2.4.1.2) and three executable files, that is source codes compiled by the MARSCHALS L2 team (see 2.4.1.3).

2.4.1.1 The source files

The MARSCHALS L2 codes are based on a set of source files written in FORTRAN language (MARC code and OFM codes) and C language (SAMM code).
The names of these files that have to be compiled to run the related codes are described in RD 3.

2.4.1.2 The batch files

The batch files are composed by three makefiles and three run files. The three makefile files are provided (one for each of the three codes of the MARSCHALS suite) to compile the source files of the related code. More details can be found in 4.1 (makefile to compile the SAMM code), 5.1 (makefile to compile the MARC code) and 6.1 (makefile to compile the OSM code).
The three run batch files (one for each code) are provided to run the codes. They are basically shell scripts where some indication of file names and paths is addressed. More details can be found in 4.2 (batch for the SAMM code), 5.2 (batch for the MARC code) and 6.2 (batch for the OSM code).

2.4.1.3 The executable files

Three executable files (one for each code) are provided. They are the results of a compilation of the source files of the code performed by the MARSCHALS L2 team.

2.4.2 The data files

The data files contain the data processed by the MARSCHALS L2 code, together with some user-defined settings concerning the run of the codes.
The set of data files is composed by:

- the settings files with the user-defined set-up of the code (see 2.4.2.1),
- the L1b measurement file with the measurement data coming from the MARSCHALS L1b team, (see 2.4.2.2),
- the Pre-processed input data files with the aforementioned L1b measurement file elaborated by the SAMM code, (see 2.4.2.3);
- the ancillary files, (see 2.4.2.4);
- the auxiliary files, (see 2.4.2.5);
- the output files, (see 2.4.2.6);
- the I/O files for the sequential fit (see 2.4.2.7).

2.4.2.1 The settings files

The MARSCHALS L2 software is built in order to offer to the user the opportunities to set many options and to activate some extra features, for the SAMM, the MARC and the OFM codes. These user-defined choices are stated by editing some ASCII settings files described in RD 3.

2.4.2.2 The L1b measurement files

The files used as input of the MARSCHALS L2 software (i.e. as input of the SAMM Pre-Processor) are the L1b data files coming from the MARSCHALS measurements. These files, listed in paragraph 4.3.4, are processed by the SAMM Pre-Processor, and thus they can be considered as input files of the overall MARSCHALS L2 software (together with the auxiliary and ancillary data files).

2.4.2.3 The pre-processed data files

At the present, the retrieval procedure performed by MARC code processes the single scan separately. Once the SAMM Pre-Processor is executed, the file *observations_$.dat* related to the scan [$] to be processed must be manually renamed as *observ.dat*. Since the pre-processor can be run separately, the *observ.dat* file can be considered as the measurement input for MARC code. The same procedure has to be followed for the *OCM_$.dat* file related to the scan under consideration, that has to be manually renamed as *OCM.dat*.

2.4.2.4 The ancillary data files

The MARC retrieval code of the MARSCHALS software needs also some ancillary files related to the processed measure. These files, described in RD 3, contain information provided by level 1 Analysis.

> Please notice that a set of ancillary data files are provided together with the other files of the MARSCHALS L2 suite. They are referred to the provided test data; if the codes of the MARSCHALS L2 suite is run with different measurement data, the provided ancillary data could not be adequate to the new measurement scenario. In this case, a new and suitable set of ancillary data files have to be provided by the user.

2.4.2.5 The auxiliary data files

Some auxiliary files are needed to run MARC retrieval code of the MARSCHALS L2 suite. These files are not provided by the pre-processor, and have to be provided by the user. These files are described in RD 3.

> Please notice that a set of auxiliary data files are provided together with the other files of the MARSCHALS L2 suite. They are referred to the provided test data; if the codes of the MARSCHALS L2 suite is run with different measurement data, the provided auxiliary data could not be adequate to the new measurement scenario. In this case, a new and suitable set of auxiliary data files have to be provided by the user.

2.4.2.6 The output data files

The results of the retrieval procedure are stored in some output data files; the format of these files are described in RD 3.

2.4.2.7 The I/O files for the sequential fit

If a sequential fit is adopted, the output data to be used as input data of the following step are stored in some I/O data files. These files are described in RD 3.

2.5 *The key-word syntax of the files with the user defined settings*

The settings files used by the MARSCHALS L2 codes contain data that can be edited or read by the user. To make the editing and the reading easier, the most part of these files are written in ASCII format, and are characterised by the following syntax (hereinafter *key-word syntax*).

> In the settings file characterized by the *key-word syntax*, data are written in ASCII format and arranged in rows; inside a single data row, all data (in ASCII format) are separated by some blanks. The data the row is related to is specified in the previous row with a key-word.
> Possible commented rows are preceded by a #.
> Data and commented rows can be separated by some blank rows that will not be considered.

A single data is specified in the following way:

a commented row (starting with a #) introduces the variable under definition; then a string (hereinafter *key-word*) identifying the variable to be defined is written in a not commented row between square brackets; the related quantity is reported in the following (not commented) data rows. Notice that the variable could be vectorial (i.e. a single data specification can occupied more than a single row in case of matricial data); moreover the format of the data can be different among the row elements (e.g. in the same row it is possible to find integer data together with real data and so on).
An example of a definition of a matricial quantity is reported in the following:

```
# comments
[key-word]
a11    a12    ...    a1n
...    ..     ...    ...
am1,   am2,   ...    amn
```

Because of the presence of the key-word string, the order of the definition among different data is not matter of care; in any case.

3 Installation of the MARSCHALS L2 codes

Please notices that the executable version provided to ESA with the delivered computer has been already installed and compiled by the MARSCHALS L2 team, and does not require any further installation procedure. The reading of this chapter is thus not needed to run the executable codes since the installation have to be performed only to transfer the software on another computer.

The MARSCHALS L2 software is composed by the SAMM code (written in C language), the MARC and the OFM codes; these last written in FORTRAN language.

- The MARSCHALS L2 codes are provided by means of their source files.
- To run the MARSCHALS L2 codes the user needs of the source files (to be compiled) but also adequate sets of input data files, ancillary files and auxiliary files are requested.

The files needed to run the MARSCHALS L2 codes has been listed in 2.4.

To install the codes it is sufficient to copy the source files, the input data files, the auxiliary files and the ancillary files on the hard disk, according to a given arrangement described in Sect. 3.1.

All the aforementioned files have to be placed in a dedicated root directory, adequately arranged in subdirectories. The name of the root directory can be chosen by the user; hereinafter it will be referred as *MARSCHALS_L2* directory.
MARSCHALS software files can be arranged in a personalized manner on the hard disk, also different from the one proposed in Figure 3-2, Figure 3-3 and Figure 3-4.

The default arrangement of the files needed to run the three MARSCHALS L2 codes can be changed by the user; in this case, the run file of the related code have to be changed accordingly (see 3.1.4).

The name of the subdirectories in which the files are arranged have to be specified by the user in the *run* files of the related codes (see 2.4.1.2).

Please notice that all the addresses specified in the default batch files are relative addresses referred to the subdirectory of the MARSCHALS_L2 root directory in which the source files are placed.

Once the source files are stored, they have to be compiled with the related compiler, according to the procedure described in 4.1 (SAMM code) 5.1 (MARC code) and 6.1 (OFM codes).

3.1 The default arrangement of the files

As a default option, we propose the following architecture for the arrangement of the files needed to run the three codes of the MARSCHALS L2 suite.

The user can change the default arrangement of the files needed to run the MARSCHALS L2 software by changing the name of the subdirectories (see 3.1.4).

According to this default arrangement, depicted in Figure 3-1, the files of the SAMM, the MARC and the OSM codes have to be stored on the hard disk in three dedicated subdirectory (named respectively *SAMM, MARC* and *OFM*) placed in a main MARSCHALS L2 directory named *MARSCHALS_L2*.

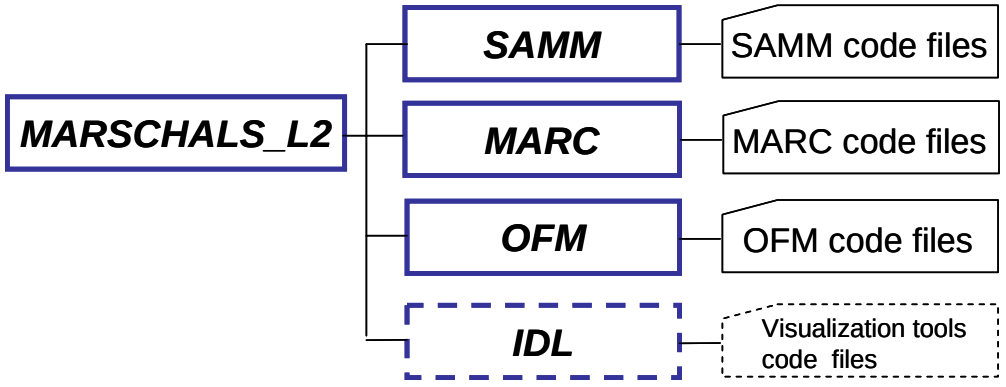

Figure 3-1. The overall default arrangement in three main subdirectories of the files composing the MARSCHALS L2 suite (in blue the subdirectories).

For each of the three proposed subdirectories a default arrangement of the files in a tree structure subdirectories is proposed, as reported in the following chapter 3.1.1 (SAMM code), 3.1.2 (MARC code) and 3.1.3 (OFM codes).

Please notice that a further IDL subdirectory is present. This subdirectory contains the IDL code files of a visualization tool that has been developed in the context of the MARSCHALS project by the MARSCHALS L2 team to make easier the presentation and the analysis of the retrieval results obtained by the MARSCHALS software. This tool is provided as it is as an IDL code, and requires an IDL licence in the computer in which it is executed. The description of this tools is provided in Chapter 7.

3.1.1 The default arrangement of the SAMM files

The default arrangement of the *SAMM* directory, containing the files needed to run the SAMM code, is depicted in Figure 3-2.

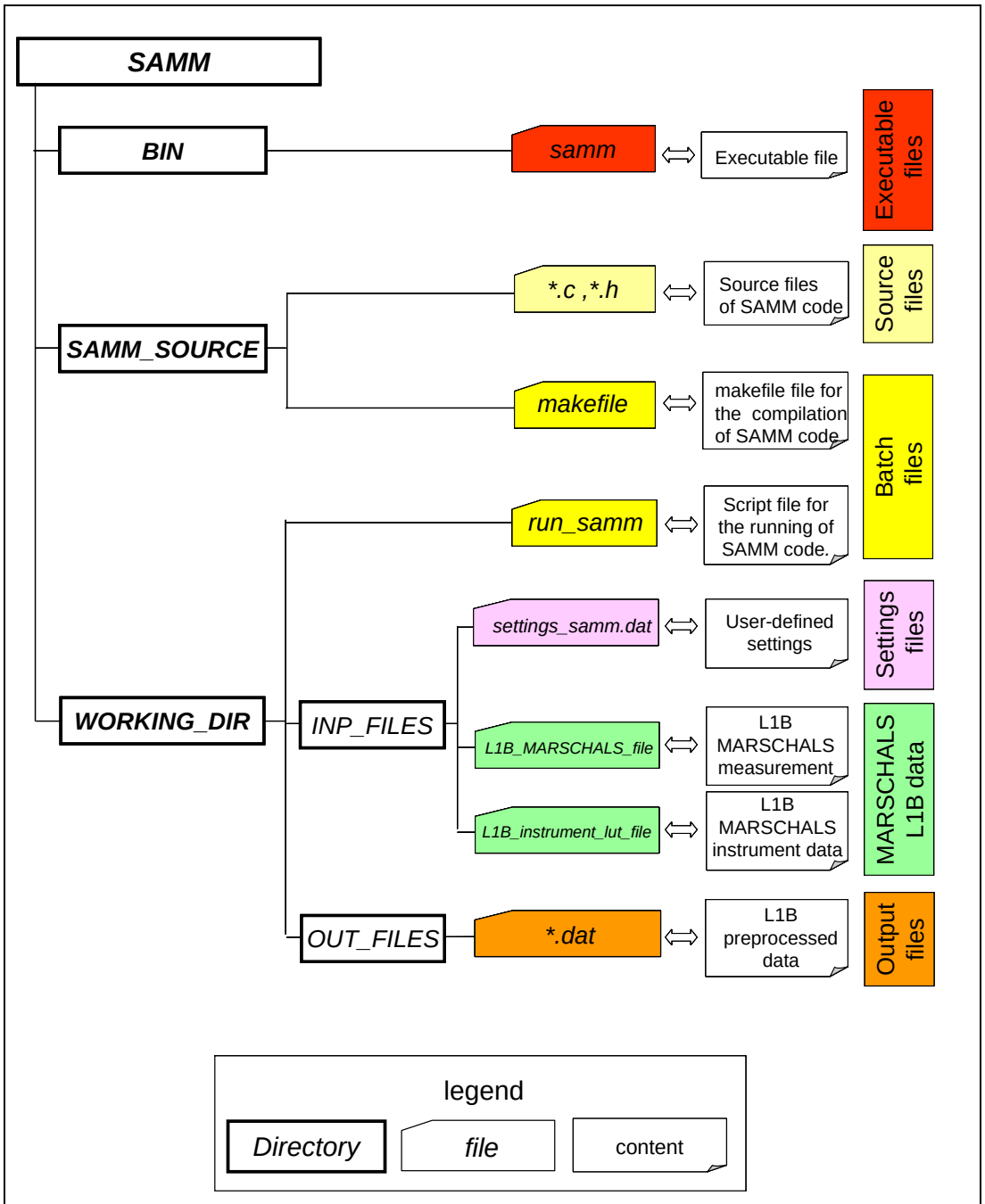

Figure 3-2. The arrangement of the SAMM code data and source files. The names of the subdirectories are those of the default arrangement.

The *SAMM* directory presents three subdirectories:

- the *BIN* subdirectory;
- the *SAMM_SOURCE* subdirectory;
- the *WORKING_DIR* subdirectory.

The *BIN* directory contains:

- the *samm* executable files of the SAMM code;

The *SAMM_SOURCE* directory contains:

- the source files of the SAMM code;
- the *makefile* file to be used to compile the aforementioned source file.

The *WORKING_DIR* directory contains:

- the *run_samm* script file to be used to run the SAMM code;
- the *INP_FILES* subdirectory;
- the *OUT_FILES* subdirectory.

The *INP_FILES* directory contains:

- the *settings_samm* file to be used for the set up of the preprocessing procedure (see 4.3.7);
- the L1b_MARSCHALS file with the data coming from the L1b measurement.
- the *l1b_instrument.lut* ASCII file containing some instrument characterization needed to the pre-processing operations.

The *OUT_FILES* directory contains:

- the *flight_view.dat* ASCII file reporting general information about the flight campaign under consideration (see 4.5.1);
- the *scan_§_view.dat* ASCII file reporting general information about the scan (§) (see 4.5.2);
- the *OCM_§.dat* ASCII file containing OCM images and additional information about a given scan (§) in the given campaign. (see 4.5.3);
- the *observations_§.dat* ASCII file containing all measurement data referred to the scan (§); it is used as input of MARC.

3.1.2 The default arrangement of the *MARC* files

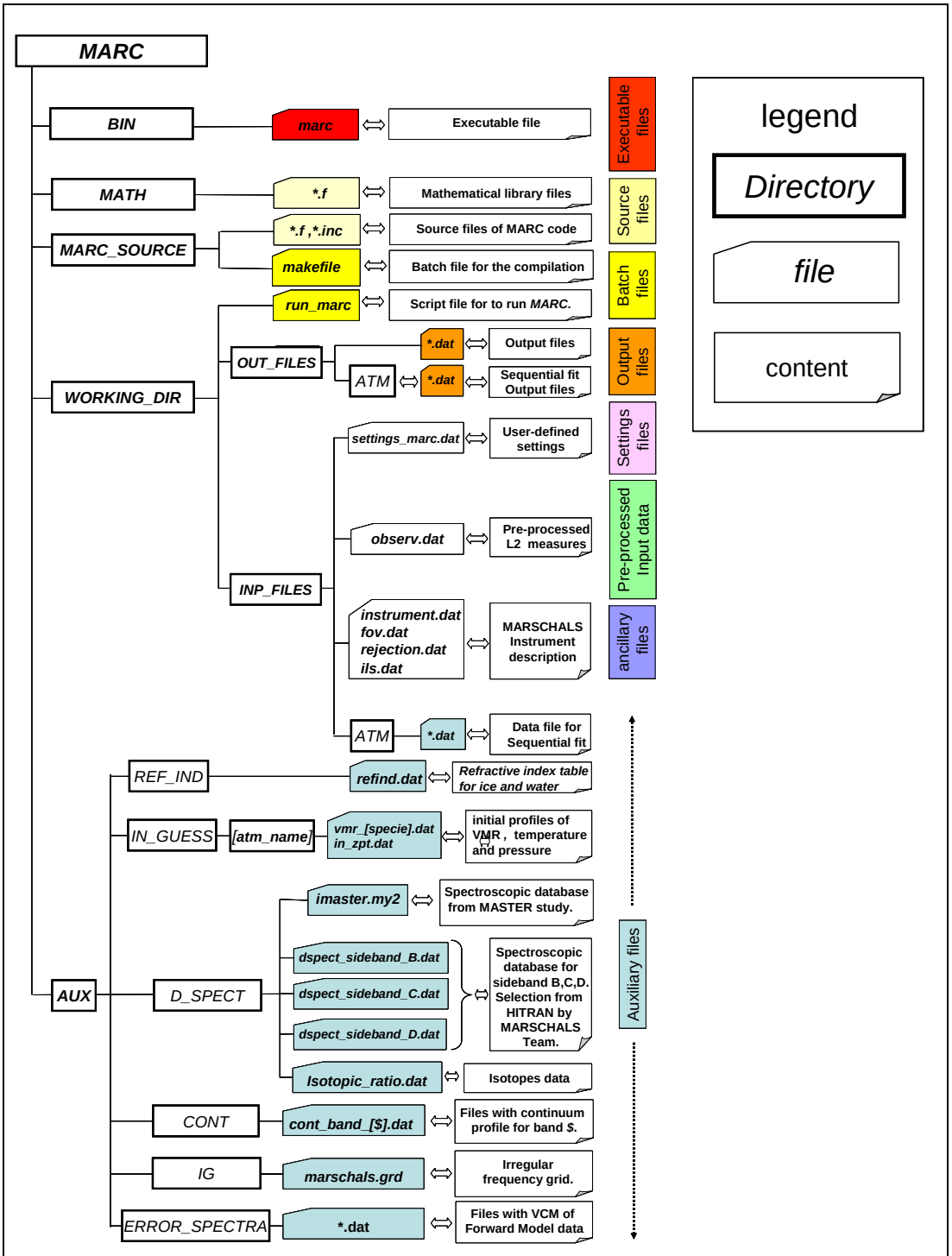

Figure 3-3. The arrangement of the MARC code data and source files. The names of the subdirectories are those of the default arrangement.

The default arrangement of the *MARC* directory containing the files needed to run the MARC code is depicted in Figure 3-3. It is characterised by 5 subdirectories:

- The *BIN* subdirectory;
- the *MATH* subdirectory;
- the *MARC_SOURCE* subdirectory;
- the *WORKING_DIR* subdirectory;
- the *AUX* subdirectory.

The *BIN* directory contains:

- the *marc* executable files of the MARC code;

The *MATH* directory contains:

- a set of mathematical library files.

The *MARC_SOURCE* directory contains:

- the source files of the MARC code;
- the *makefile* file to be used to compile the aforementioned source files.

The *WORKING_DIR* directory contains:

- the *run_marc* script file to be used to run the MARC code;
- the *INP_FILES* subdirectory;
- the *OUT_FILES* subdirectory.

The *INP_FILES* subdirectory contains:

- the *settings_marc.dat* file (with the data to set the retrieval procedure);
- the *observ.dat* file with the pre-processed MARSCHALS L1b data;

and four files with data about the instrument:

- *instrument.dat* file with general information about MARSCHALS instrument;
- *fov.dat* file with general information about the FOV of the MARSCHALS instrument;
- *rejection.dat* file with general information about the rejection factor for the image band of the MARSCHALS instrument;
- *ils.dat* file with general information about the ILS of the MARSCHALS instrument.

Furthermore the *INP_FILES* directory contains the *ATM* subdirectory with a set of files containing data for the possible sequential fit.

The *OUT_FILES* subdirectory contains:

- the *output files* (with the results of he retrieval);
- the *ATM subdirectory.*

The *ATM* directory contains:

- files with output data to be used when the sequential fit is adopted.

The *AUX* directory contains the auxiliary and the ancillary data files arranged in the following subdirectory:

- *REF_IND;*
- *IN_GUESS*
- *D_SPECT;*
- *CONT;*
- *IG;*
- *ERROR_SPECTRA.*

The *REF_IND* directory contains the *refind.dat* file with the refractive index table for ice and water.

The *IN_GUESS* directory contains a-priori data about the atmosphere; different atmospheres can be used, and for sake of clarity different subdirectories can be used for atmospheres with different characteristics. At the present it contains:

- the *Mid_Lat* subdirectory with data about a mid latitude atmosphere. The *MidLat* subdirectory contains:
  - a set of *vmr_[species]* files, (one for each species *species*) , containing the a-priori vmr profiles of the species;
  - the file *in_zpt.dat* with the a-priori values of pressure and temperature profiles.

The *D_SPECT* directory contains:

- the file *imaster.my2* with the spectroscopic database coming from the MASTER study;
- the *dspect_sideband_B.dat*, *dspect_sideband_C.dat* and *dspect_sideband_D.dat* files with the spectroscopic database from HITRAN to be used for the *sideband B*, *sideband C* and *sideband D* respectively;
- the *isotopic_ratio.dat* with data about the isotopes.

The *CONT* directory contains three files:

- cont_band_B, cont_band_C and cont_band_D with the continuum profiles for the three bands.

The *IG* directory contains:

- the *marschals.grd* file with the irregular frequency grid.

The *ERROR_SPECTRA* directory contains:

- a set of files with information about error spectra data.

3.1.3 The default arrangement of the OFM files

The default arrangement of the *OFM* directory containing the files needed to run the OFM codes is depicted in Figure 3-4.

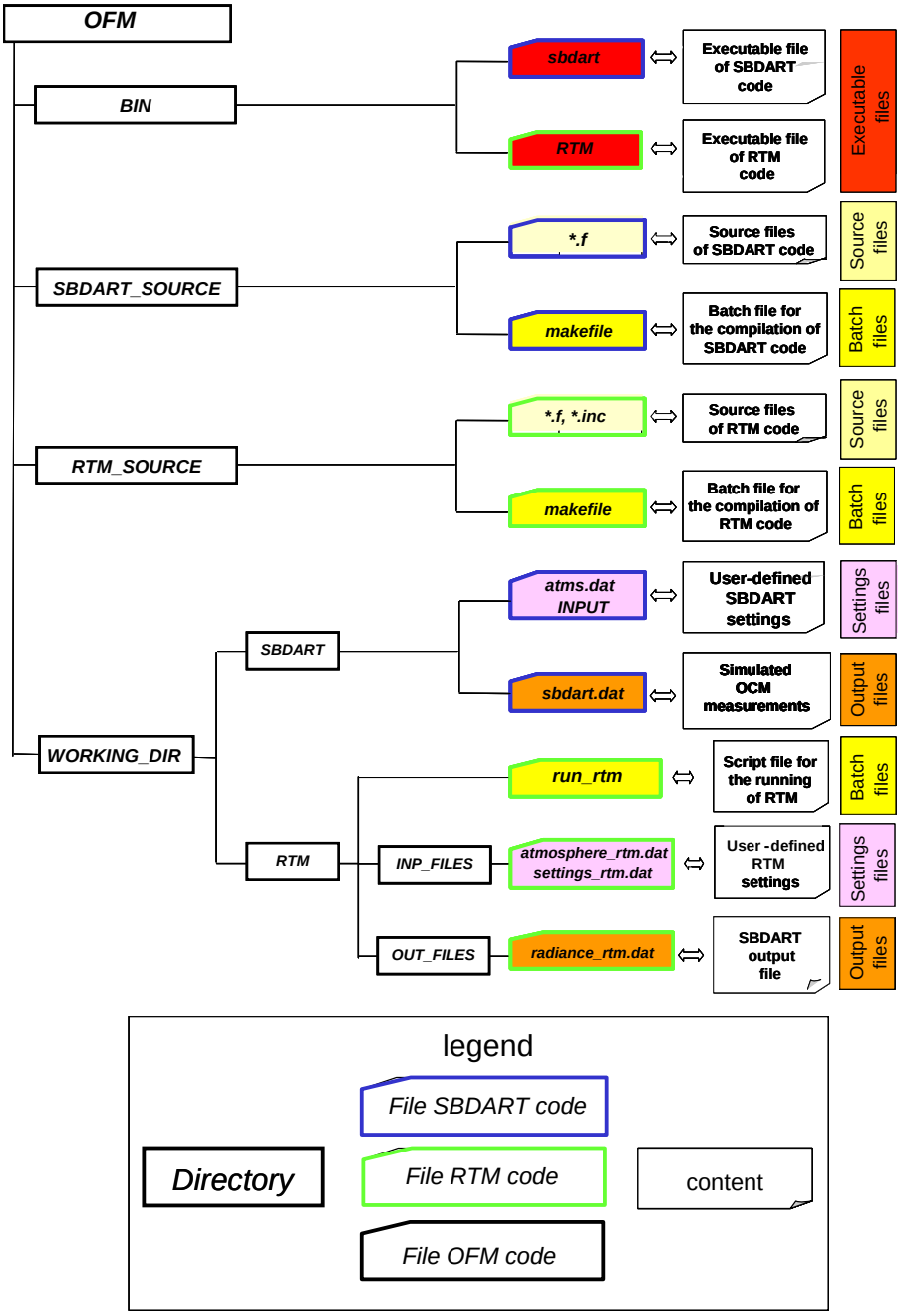

Figure 3-4. The arrangement of the OFM codes data and source files. The names of the subdirectories are those of the default arrangement.

The OFM directory contains 4 subdirectories:

- the *BIN* subdirectory;
- the *SBDART_SOURCE* subdirectory;
- the *RTM_SOURCE* subdirectory;
- the *WORKING_DIR* subdirectory.

The *BIN* directory contains 2 executable files:

- the *sbdart* executable file of the SBDART code;
- the RTM executable file of the RTM code.

The *SBDART_SOURCE* directory contains:

- the source files of the SBDART code;
- the *makefile* batch file for the compilation of the SBDART code.

The *RTM_SOURCE* directory contains:

- the source files of the SBDART code;
- the *makefile* batch file for the compilation of the RTM code.

Please notices that the OFM is composed by two subcodes, the SBDART code and the RTM code, that have to be run in sequence. Where possible, the related files are maintained separated in different subdirectories.

The *WORKING_DIR* directory contains two files and two subdirectories:

- the SBDART subdirectory;
- the RTM subdirectory.

Please notices two subdirectories with the same name (SBDART) are present at different level inside the OFM directory.

The SBDART subdirectory placed in the *WORKING_DIR* directory contains:

- the *atmos.dat* and the INPUT files with the user-defined SBDART settings;
- the *sbdart.dat* file with simulated OCM measurement.

The RTM subdirectory placed in the *WORKING_DIR* directory contains:

- the *run_rtm* batch file for the execution of the RTM code;
- the *INP_FILES* subdirectory;
- the *OUT_FILES* subdirectory.

The *INP_FILES* directory contains:

- the *atmosphere_rtm.dat*
- the *settings_rtm.dat* file with the user-defined RTM settings.

The *OUT_FILES* directory contains:

- the *radiance_rtm.dat* OFM output file with the simulated OCM radiance produced by the RTM.

3.1.4 How to change the default names of the subdirectories

The user can change the default arrangements of the subdirectories by moving and changing the name of the subdirectories.

Please notice that if a change is made in the default arrangement of the files, some changes have to be made also to the provided batch files: the batch files to be used to run the codes, i.e. *run_marc* (see 5.2), stored as a default in the *MARC_working* sub-directories, have to be changed accordingly by the user.
If used, also the *makefile* files that are provided together with the source codes of the MARSCHALS L2 codes in order to make easier the compilation of the codes have to be changed accordingly (see 4.1 for the SAMM code, 5.1 for the MARC code, 6.1 for the OFM code).

4 SAMM code

This chapter aims to describe the use of SAMM Pre-Processor code with its implemented features. The SAMM Pre Processor is intended as a tool to plan the activity of data analysis and should be used to perform a first selection of the MARSCHALS measurements and transferring them into a suitable format, readable by MARC code.

> The SAMM Pre-Processor will allow tackling changes in the format of the input data without any changes in the MARC code; in such a way a new validation of the Retrieval Code will be not necessary in case of changes in input data format.

An important functionality of SAMM is to provide to the user an overview of the measurement scans available from the analysed campaign data, comprehensive of those information which could help the user in the selection of a specific scan. Subsequently the user can indicate which measurement he is interested in, evaluating a set of parameters, such as quality indicators from Level 1, OCM information, and so on.
In order to reach this goal, in addition to the main functionality of selection/transcription of data, SAMM Pre-Processor performs the elaboration of other information, not properly necessary for the execution of MARC retrieval, but useful to plan the strategy of MARC elaboration. Some examples are: measurement geolocation from Level 1 and rejection of lines of sight from scan measurements due to low contrast in spectrum.
The SAMM code can also be used to produce the ancillary data files (i.e. *fov.dat*, *ils.dat*, *instrument.dat* and *rejection.dat*) required by the MARC code, by considering some data files that are provided by the L1b team with the related measurement data files. The aforementioned ancillary files can be also directly created by the user; in any case they must be present to run the MARC code.
In the following paragraphs the input files necessary for SAMM Pre-Processor execution (see 4.3), the output files resulting from the Pre Processing procedure (see 4.5), a list of modalities and functionalities of the code and some indications on the available user-defined choices (see 4.3.5) will be addressed, together with the instruction to compile (see 4.1) and run (see 4.2) the code.

4.1 How to compile the source files of the SAMM code

The SAMM code is composed by some source files written in standard C language.

> **!** Please notices that the executable code installed on the computer delivered to ESA has been already compiled by the MARSCHALS L2 team; if no change has been made to the source codes, to run the executable version a compilation is not required.

The source files of the SAMM code can be compiled in the usual way by means of whatever C compiler. The Linux operating system residing on the computer delivered to ESA is provided with the standard *gcc* compiler that is able to perform this compilation task.
Once the compilation is completed, the executable code has to be moved in the adequate directory, that is, in the proposed standard configuration, the *BIN* one.

> The compilation procedure can be performed by using an adequate *makefile* file provided by the MARSCHALS L2 team together with the code files; this *makefile* file, residing in the *SAMM_SOURCE* subdirectory, can be executed by means of the usual `make` command.
> Once the compilation is performed, the *makefile* automatically moves the executable file in the right standard subdirectory (*BIN*).

To use the *makefile* file, please move to the directory in which the code files and the *makefile* file are stored (the *SAMM_SOURCE* subdirectory in the standard configuration) and, at the prompt, type:

```
make
```

and then press the "**`enter`**" button.

> **Please notice that the `make` commands works according to the standard arrangement of the file (as reported in 3.1.1); the user can change this arrangement, but in this case, to use the `make` command, the *makefile* file have to be edited accordingly to the new user defined arrangement of the files.**

4.2 How to run SAMM code

Together with the executable MARSCHALS L2 software installed on the computer delivered to ESA a batch file is provided to run the SAMM code

The user can run the SAMM code by using the batch file *run_samm*, stored as a default in the *SAMM/WORKING_DIR* subdirectory.

To run the SAMM code by directly using the *samm* executable file, please move to the work directory (the *SAMM/WORKING_DIR* in the default arrangement) and at the prompt type:

```
../BIN/samm name_file_settings
```

where `name_file_settings` is the name of the settings file with the relative path with respect to the work directory
(i.e. *./INP_FILES/settings_samm.dat* in the default arrangement).

Alternatively, to run the SAMM code by using the batch file, please move to the subdirectory in which the *run_samm* batch file is stored (the *SAMM/WORKING_DIR* in the default arrangement) and, at the prompt, type:

```
./run_samm
```

and then press the "**`enter`**" button.

The user can change the default arrangements of the subdirectories in which the files needed to execute the *SAMM* code are stored.

If a change is made the in the default arrangement of the files (as reported in 3.1.1), the batch file *run_samm*, stored as a default in the *SAMM/WORKING_DIR* subdirectory, have to be changed accordingly by the user.

Please notice that SAMM code files are arranged in subdirectories. These subdirectories are contained in a "main directory" (named, as a default, "*SAMM*") that is user defined; all the paths are thus defined in a relative way with respect to the work directory (that is the *SAMM/WORKING_DIR* one in the default configuration).
The *run_samm* environment file contains information about the name of the setting file to be considered

4.3 The files needed to run the SAMM code

To run the SAMM code some files of different kind are requested. These files are listed and classified in Table 4-1.

The data file needed to run the SAMM code	
File classification	**Name of file**
source files(1)	**.c* (source files) and **.h* (include files) standard C language files. The whole list is reported in RD 3.
batch file	*makefile* - makefile to compile the SAMM source files (see 4.1 for the SAMM compilation procedure and RD 3 for the format file).
	run_samm(2) - batch file to run the SAMM code.
executable file	*samm* - executable file of he SAMM code.
measurement files	*L1B_MARSCHALS_file* (the name of this file changes according to an internal MARSCHALS L1B syntax).
ancillary data files	*L1B_instrument.lut* file
auxiliary data files	The pre-processor can be used to generate the files (*fov.dat*, *ils.dat*, *instrument.dat* and *rejection.dat*) containing the instrument characterizations needed by the retrieval code. In this case the following L1b auxiliary files are needed: • files containing *spectral response* (for each channel and each band); • files containing *the fov (acap)* (for each band).
settings file	*settings_samm.dat*
(1) These files are needed to create the *samm* executable code; anyway a *samm* executable file has been already created and installed in the delivered package; thus these files are not requested to run the provided default version of the SAMM code . (2) Actually this file is not mandatory to run the code, since it is only a batch file to run in an easier way the related codes.	

Table 4-1. The data file needed to run the SAMM code.

The aforementioned files have to be stored in the hard disk of the computer devoted to run the SAMM code, according to a given files arrangement. A default arrangement suggested by the MARSCHALS L2 team is described in paragraph 3.1.

> The computer delivered to ESA is provided with the entire set of files needed to run the SAMM code, stored according to the default arrangement presented in paragraph 3.1.

4.3.1 The source files

The SAMM code is based on a set of source files (**.c* source files , **.h* include files) written in standard C language. The whole list is reported in RD 3.

4.3.2 The batch file

The *makefile* batch file can be used to compile the source files and, after the compilation of the source files, to move the executable in the right directory, according to the default arrangement described in chapter 3.1.1 (for the compilation procedure of the SAMM code, please see 4.1)

> Please notices that the *samm* executable file has been already produced and installed in the computer delivered to ESA; thus to run the proposed default code a compilation is not required.

The batch file *run_samm* is provided for the running of the SAMM executable code stored in the *samm* file.

4.3.3 The executable file

The *samm* executable file, obtained by compiling the SAMM code source files according to the default setting, is provided inside the computer delivered to ESA (stored in the *SAMM/BIN* directory) and in the related package.

4.3.4 The measurement files

SAMM receives as input the data from Level 1B. These data are expected to be stored in the following data file:

- L1B_MARSCHALS_file : this is a binary file containing the MARSCHALS L1B measurement; its name follows an internal L1B syntax (see RD 1);

Together with the input data coming from the L1b, also some auxiliary (see 4.3.5) and ancillary (see 4.3.6) files are needed to run the SAMM code. Trivially, to execute the SAMM code, the executable file is needed.

4.3.5 The ancillary data files

Together with the L1B data file, the L1 team provides some auxiliary information about the instrument; these information are stored in the following file:

- *L1B_instrument.lut* file : this is an ASCII file containing some instrument characterization needed to the pre-processing operations.

4.3.6 The auxiliary data files

No auxiliary data files are requested for the basic run of the SAMM code. If SAMM code is used to produce the MARC ancillary files (i.e. *fov.dat*, *ils.dat*, *instrument.dat* and *rejection.dat* files) the following L1b data files are needed:

- files containing *spectral response* (for each channel and each band);
- files containing *the fov (acap)* (for each band).

4.3.7 The settings file

The user-defined options available for the run of the SAMM code are specified in the *settings_samm* file. These options are described in the following paragraph 4.4.

4.4 How to set the SAMM pre-processing

SAMM Pre-Processor provides many options and possible settings of the Pre-Processing procedure that can be specified by the user.
As an example, the user can define:

- The name of the input/output files (see 4.4.1);
- The quantities to be extracted from the L1b data files (see 4.4.2);
- The set of MARSCHALS observation to be considered (see 4.4.2).

The user-defined choices are addressed to the code by means of a setting file (*setting_samm.dat*) in ASCII format, that can be easily edited by the user. In this file the possible choices are defined by means of parameters value specified in some dedicated rows; each row containing parameters is preceded by a row with a key-word inside square brackets (whose appearance is thus *[key-word]*), specifying the quantity defined in the following row.
The definition of a parameter will be thus as in the following example:

```
[keyword]
parameter_value
```

Due to the presence of the key-word, the order of the couples of rows specifying the parameters is meaning-less, while the absence of a key-word will abort the process.
In the following the description of all the available user-defined choices is reported,

4.4.1 How to set the input/output namefiles

The name of the input/output files can be defined by the user by setting the following parameters:

Key-words	related quantity	dim.	data type
[NAME_L1B_FILE] input_file_name	The string *input_file_name*, labelled by the key-word *[NAME_L1B_FILE]* specifies the name of the binary file (L1B_MARSCHALS_file) as received from Level 1B.	dimless	char (max 200 char)
[NAME_OUTPUT_DIRECTORY] output_directory_name	The variable *output_directory_name* specifies the directory in which SAMM will save the output files (path is requested).	dimless	char (max 200 char)
[NAME_LUT_FILES] name_file_L1B_LUT	name of the L1B file containing the instrument characterization.	dimless	char (max 200 char)

4.4.2 How to define the quantities to be extracted from the L1b data files.

The set of data to be extracted from the L1b data files are specified by defining the following parameters.

Key-words	related quantity	dim.	data type
[EXTRACTION_CRITERION] criterion_number	The variable *criterion_number* is a code number that indicates the criterion that the Pre Processor will use to extract data. The possible criteria to select are: • *criterion_number* =1 Scan ordinal number. The Pre Processor will select the measurements on the basis of the sequential number of the scan. The range of the sequential numbers is specified by the parameter labelled by the key-word *[Scans_range_selection]* defined in the following. • *criterion_number* = 2 Aircraft coordinates. The Pre Processor will select the measurements on the basis of the aircraft coordinates. The range of coordinates is specified by the parameter labelled by the key-word *[Scans_coordinate_selection]* defined in the following.	dimless	int

Key-words	related quantity	dim.	data type
[SCANS_RANGE_SELECTION] scan_number_1, scan_number_2	The SAMM Pre Processor will extract those scans whose ordinal number lays in the range from *scan_number_1* to *scan_number_2*. This array defines the numerical range of the scans to be extracted. This field is read only if *extraction_criterion*==1.	dimless	2×int
[SCANS_COORDINATE_ SELECTION] latitude_1, longitude_1, latitude_2, longitude_2.	This array defines the coordinate range of the scan to be extracted. The SAMM Pre Processor will extract those scans for which aircraft coordinates lay in the indicated range. This field is read only if *extraction_criterion*==2.	degrees	4×double
[BAND_SELECTION] selector_first_band, selector_second_band, selector_third_band.	This array defines the band to be considered. The Pre Processor will select those scans whose measurements are performed inside the indicated band. All other measurements are discarded. All band combinations are possible; the combination is specified by three logical numbers, being the first one referred to the first band, the second to the second band and the third to the third band. The aforementioned order is that used in the L1B data file (please notices that the first band could not be the band B). If the logical number is set to one, the related band is considered, otherwise it is not selected.	dimless	3×int

Key-words	related quantity	dim.	data type
[LOS_RANGE_SELECTION] altitude_los_1, altitude_los_2.	This array defines the altitude range of the scans to be extracted. The Pre Processor will extract data related to the Lines Of Sight whose tangent altitudes lay in the range from *altitude_los_1* to *altitude_los_2*.	Km	double double
[CONTRAST_LEVEL_THRESHOLD] threshold	This value defines the acceptable contrast range. The Pre Processor will select those scans whose spectra show a contrast level greater than *threshold*. The contrast level value is defined in the following. All the other measurements will be discarded.	dimless	double
[NOISE_LEVEL_THRESHOLD] threshold	This value defines the acceptable noise range. The Pre Processor will select those scans whose spectra show a noise level greater than *threshold*. All the other measurements will be discarded.	dimless	double
[QUALITY_NUMBER_THRESHOLD] threshold	At the moment, this settings option is not available.	dimless	int

4.4.3 How to select the files generated by the pre-processor.

The pre-processor can extract some general information about the measurement campaign, as well as the measurement data related to a selected scan. The different modalities of extraction with the related value of the extraction type variable are specified by the means of the following parameter.

Key-words	related quantity	dim.	data type
[EXTRACTION_TYPE] extraction_type	The value *extraction_type* is an integer that indicates the set of data the Pre processor will extract. The different modalities of extraction are reported in the following list, where *§* indicates the scan inside the considered flight. *extraction_type* = 1 The Pre processor performs an overview, extracting only general data referred to the flight. It generates the file *flight.view* (see 4.5.1). *extraction_type* = 2 The Pre processor performs an overview, extracting detailed data referred to the scan. It generates the file *scan_§.view* (see 4.5.2). *extraction_type* = 3 The Pre processor extracts MARSCHALS data. It generates all the files needed by MARC code, that is the *observation_§.dat* file (see 4.5.4); it also generates the file *scan_§.view* (see 4.5.2). *(continue in the next page)…*	dimless	int

Key-words	related quantity	dim.	data type

	`extraction_type = 4`
	The Pre processor extracts OCM data. It generates the file *OCM_§.dat*
	(see 4.5.3)
	`extraction_type = 5`
	The Pre processor generates all files		

4.4.4 How to enable the output of files containing instrumental information.

The pre-processor can be used to generate some files containing instrument information to be used as auxiliary files for the retrieval code; particularly the pre-processor can generate the following files:

- *instrument.dat*;
- *rejection.dat*;
- *fov.dat*;
- *ils.dat*.

This output is optional because these files contain instrument characterization data can be produced once for each flight.

Key-words	related quantity	dim.	data type
[CREATE_INSTRUMENT_DAT] flag	Flag used to create the auxiliary file "*instrument.dat*" 1 : the file is created; 0 : the file is NOT created. Data are extracted from *l1b_instrument.lut* file.	dimless	int
[CREATE_REJECTION_DAT] flag	Flag used to create the auxiliary file "*rejection.dat*": 1 : the file is created; 0 : the file is NOT created. Data are extracted from *l1b_instrument.lut* file.	dimless	int
[CREATE_ILS_DAT] flag	Flag used to create the auxiliary file "*ils.dat*". 1 : the file is created; 0 : the file is NOT created.	dimless	int
[CREATE_FOV_DAT] flag	Flag used to create the auxiliary file "*fov.dat*": 1 : the file is created; 0 : the file is NOT created.	dimless	int
[LIST_ SPECTRAL_RESPONS_B] name_file_list_B	Name of the file with the list of spectral response in band B. This file is generated by the user and contains the list of the L1B spectral response files. The pre-processor analyses only the files summarized in this list. Note that this field is read only if CREATE_ILS_DAT == 1	dimless	char (max 200 char)

Key-words	related quantity	dim.	data type
[LIST_ SPECTRAL_RESPONS_C] name_file_list_C	name of the file with the list of spectral response in band C. This file is generated by the user and contains the list of the L1B spectral response files The pre-processor analyses only the files summarized in this list. Note that this field is read only if CREATE_ILS_DAT == 1	dimless	char (max 200 char)
[LIST_ SPECTRAL_RESPONS_D] name_file_list_D	name of the file with the list of spectral response in band D. This file is generated by the user and contains the list of the L1B spectral response files The pre-processor analyses only the files summarized in this list. Note that this field is read only if CREATE_ILS_DAT == 1	dimless	char (max 200 char)
[FILE_FOV_B] name_file_fov_B	name of the L1B file containing the instrumental Field of View in band B. Note that this field is read only if CREATE_FOV_DAT == 1	dimless	char (max 200 char)
[FILE_FOV_C] name_file_fov_C	name of the L1B file containing the instrumental Field of View in band C. Note that this field is read only if CREATE_FOV_DAT == 1	dimless	char (max 200 char)
[FILE_FOV_D] name_file_fov_D	name of the L1B file containing the instrumental Field of View in band D. Note that this field is read only if CREATE_FOV_DAT == 1	dimless	char (max 200 char)

4.4.5 SAMM extra-features

To allow, during the first part of the code development, the testing of MARC code in presence of temporary input data characterized by some problems in the reported measurements, a series of extra-feature of the SAMM preprocessor code have been developed such as:

- manual selection of the LOS:
  the user can exclude one or more LOS from the file of observations; this manual selection has been added to exclude LOS having large values of chi-square.

- computation of the average spectral error:
  for each LOS, the preprocessor computes an average (rms) value for the spectral error; the user can decide to use the original error or the averaged one in the retrieval.

- treatment of spectra containing un-realistic values:
  some spectra in band B contains channels having un-realistic values (NULL or very large values) in spectral data and/or in spectral data error; when the file of observations is produced, these channels can be replaced with values selected by the user. This feature of the preprocessor allows to exclude the un-realistic values by introducing a large value in the spectral error. The channels marked as un-realistic are not taken into account when the average spectral error is computed.

These extra-feature have not to be used with the “normal” MARSCHALS L1b data the code is developed for, and thus the description of their usage is out of the purpose of this document.

4.4.6 The contrast level

The contrast level related to a given spectrum is defined as follows:

$$Contrast\_Level = \frac{I_{peak}}{I_{valley}}$$

that is the ratio between two reference intensities, I_{peak} and I_{valley}.
Reference intensities are evaluated by averaging the spectral intensity over given spectral intervals, each related to a specific band, as specified in the following.

For band B

I_{peak} is evaluated as the mean value on the interval [301.6 - 302.0] GHz, for a total of three spectral points. This interval contains the strongest O_3 line.
I_{valley} is evaluated as the mean value on the interval [295.6 - 296.4] GHz, for a total of five spectral points (see Figure 4-1).

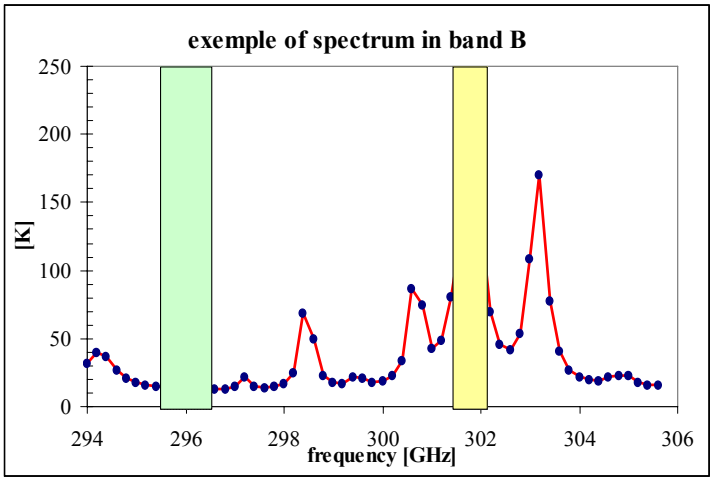

Figure 4-1. The range in which the I_{valley} (green zone) and the I_{peak} (yellow zone) values are evaluated in the band B.

For band C:

I_{peak} is evaluated as the mean value on the interval [324.9 - 325.3] GHz, for a total of three spectral points. This interval contains the strongest H_2O line.
I_{valley} is evaluated as the mean value on the interval [322.5 - 323.3] GHz, for a total of five spectral points (see Figure 4-2).

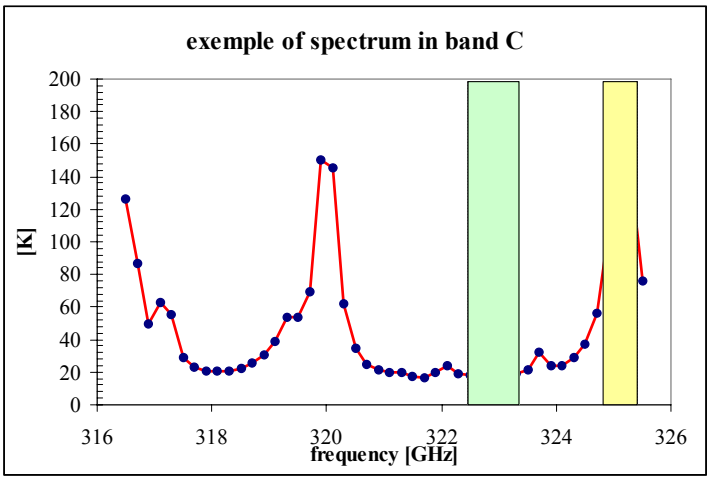

Figure 4-2. The range in which the I_{valley} (green zone) and the I_{peak} (yellow zone) values are evaluated in the band C.

For band D :

I_{peak} is evaluated as the mean value on the interval [343.0 - 343.4] GHz, for a total of three spectral points. This interval contains the strongest O_3 line.
I_{valley} is evaluated as the mean value on the interval [347.6 - 348.4] GHz, for a total of five spectral points.

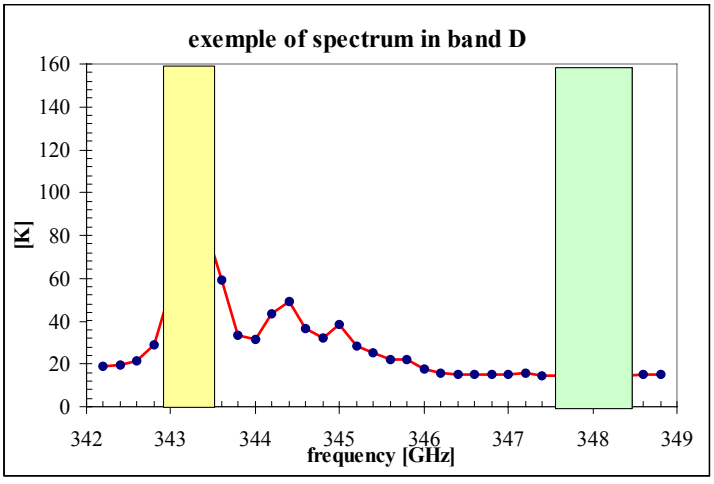

Figure 4-3. The range in which the I_{valley} (green zone) and the I_{peak} (yellow zone) values are evaluated in band D.

4.5 Output files

According to the user-defined setting specified by the parameter [**EXTRACTION_TYPE**] in the *settings_samm* file (see 4.4.3), the SAMM Pre Processor code is able to produce a set of output files, listed in **Table 4-2**, together with the related contents and the definition of the special character (§) used inside the filenames.
This paragraph is devoted to the introduction of these output files.

Pre-processor product	
flight_view.dat	ASCII file reporting general information about the flight campaign under consideration (see 4.5.1).
scan_§_view.dat	ASCII file reporting general information about the scan (§) (see 4.5.2).
OCM_§.dat	ASCII file containing OCM images and additional information about a given scan (§) in the given campaign. (see 4.5.3).
observations_§.dat	ASCII file containing all measurement data referred to the scan (§); it is used as input of MARC. (see 4.5.4).
where:	
§	Identifier of the scan (number of the scan).
Pre-processor auxiliary product	
instrument. dat	ASCII file containing some instrumental parameters; it is used as input of MARC (ancillary file).
rejection. dat	ASCII file containing some instrumental parameters; it is used as input of MARC (ancillary file).
ils. dat	ASCII file containing the frequency response of the channels; it is used as input of MARC (ancillary file).
fov. dat	ASCII file containing the antenna Field of View. it is used as input of MARC (ancillary file).

Table 4-2. The output files generated by the SAMM preprocessor.

4.5.1 The *flight_view.dat* file with the general information about the flight campaign

General information about the flight campaign are reported in the file named *flight_view.dat*. It reports information about the whole set of scans collected during the campaign, giving indication about which scans meet the selection criteria defined by the user. The *flight_view.dat* is an ASCII file in raw format. The content and the organization of this file is described in Table 4-3.

4.5.2 The *scan_§_view.dat* file with the general information about the selected scan

General information about the selected scan are stored in the ASCII file named *scan_§_view.dat*, where § is a string referred to the specific scan. This file reports information related to the selected scan, giving also the indication about which lines of sight meet the selection criteria defined by the user.
The content and the organization of the file are detailed in Table 4-4.

4.5.3 The *OCM_§.dat* file with the OCM images

The OCM data are stored in the file named *OCM_§.dat*, where § indicates the specific scan.
A dedicate elaboration is necessary for extracting information from OCM images.
The Pre-Processor arranges a wrapped profile of radiance, made by segments of profiles provided by each individual OCM image referred to a specific line of sight.
For more details see RD 3.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
		Average location of the instrument			Average location of the tangent point					Lines Of Sight					
Id	time	mean altitude of the receiver	mean latitude of the receiver	mean longitude of the receiver	maximum altitude of the refracted tangent point	minimum altitude of the refracted tangent point	azimuth of the mean refracted tangent point	latitude of the mean refracted tangent point	longitude of the mean refracted tangent point	number of selected lines of sight	number of LOS tangent point inside altitude range	number of LOS constrast level greater than the threshold	number of LOS noise level less than the threshold	number of LOS quality flags greater than the threshold	total numbers of atmospheric lines of sight
Scan number	mean time stamp														

-1 Scan number;
-2 mean time stamp;
-3 mean altitude of the receiver during the scan [Km];
-4 mean latitude of the receiver during the scan [deg];
-5 mean longitude of the receiver during the scan [deg];
-6 maximum altitude of the refracted tangent point during the scan [Km];
-7 minimum altitude of the refracted tangent point during the scan [Km];
-8 azimuth of the mean refracted tangent point during the scan [deg];
-9 latitude of the mean refracted tangent point during the scan [deg];
-10 longitude of the mean refracted tangent point during the scan [deg];
-11 number of selected lines of sight (los) of the scan;
-12 number of los whose tangent point falls inside the selected altitude range;
-13 number of los whose constrast level is greater than the selected threshold;
-14 number of los whose noise level is less than the selected threshold;
-15 number of los whose quality flags are in number greater than the selected threshold
-16 total numbers of atmospheric lines of sight

Table 4-3. The structure of the *flight_view.dat* file.

Table 4-4. The structure of the *scan_$_view.dat* file.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	LOS identifier			ponting	Instrument location			Line of Sight				Tangent point				Quality flags			
Id	LOS index	L1B record index	Time stamp	Zenith nominal .	Altit.	Latit.	Longit .	azimut h point.	azimut h stdv	Zenith point.	Zenith stdv	Altit.	range	Latit.	Longit .	Conti n.	Contr ast	Noise	L1B flags

-1 counter
-2 index of the line of sight
-3 index of the record in L1B file
-4 L1B time stamp [s]
-5 nominal zenithal pointing angle [deg]
-6 altitude of receiver [Km]
-7 latitude of receiver [deg]
-8 longitude of receiver [deg]
-9 mean value of azimuth angle of the line of sight [deg]
-10 standard deviation of azimuth angle of the line of sight [deg]
-11 zenithal pointing angle [deg]
-12 standard deviation of zenithal pointing angle [deg]
-13 altitude of refracted tangent point [Km]
-14 range variability of the geometrical tangent point during the acquisition [Km]
-15 latitude of refracted tangent point [deg]
-16 longitude of refracted tangent point [deg]
-17 value of the continuum level [K]
-18 value of the contrast level
-19 value of the noise level [K]
-20 number of the positive quality flags as reported by level 1B

4.5.4 The *observation_§.dat* files with the measured data

Measurement data are reported in the file named *observation_§.dat,* where § is a string referred to the specific scan. This file contains all fundamental data necessary for the execution of MARC retrieval process. It collects in a unique solution spectral data and geometric information of measurement, separated for band, and, inside a single band, arranged for line of sight. The file refers to a single measurement scan.
The following list recalls the structure of data inside the file; a detailed description of the format is given in RD 3. At the beginning of the file, an auxiliary block of data is placed: it provides a technical description of the scan and summarizes the number of bands extracted from L1b data and the number of lines of sight contained in the scan itself. Then for each band the file reports the number of spectral points; furthermore for each line of sight (LOS) the following data are listed in the file:

- the index of LOS;
- the altitude of the instrument;
- the tangent altitude;
- the latitude and longitude of the instruments;
- the latitude and longitude of the tangent point;
- the zenithal pointing angle;
- the index of the line of sight that shares the calibration;

and finally the measured spectral points are reported in the file; for each spectral point the following information are extracted from L1B:

- frequency [GHz];
- spectrum intensity [K];
- error [K];
- gain;
- gain error;
- offset [K];
- offset error [K];
- gain-offset covariance;
- spectral data covariance..

.

Please notices that he pre-processed data are constituted by the L1b measured spectra rearranged by the pre-processor according to a format suitable for the MARC code. As a matter of fact, the pre-processor produces a set of files named *observation_§.dat* (see RD 3) by varying the scan number *§*. At the present, if the sequential fit is not set, the MARC code performs the retrieval procedure on a single scan, i.e. on data contained in one file of the set composed by the *observation_§.dat files*; once the scan is chosen, the related *observation_§.dat* file has to be renamed by the user as *observ.dat*, since MARC code in the standard configuration reads the measurement data from such a file.

This page is intentionally left blank.

5 MARC code

This chapter aims to describe the use of the MARC code with its implemented features. The MARC code is the core of the MARSCHALS suite, and it constitute the retrieval code, that performs the retrieval of the specified minor atmospheric constituents from the MARSCHALS L2 previously rearranged in a suitable way by the SAMM code.
The retrieval is implemented as an iterative procedure based on a Forward Model, and together with the MARSCHALS L1b data, it requires also some auxiliary and ancillary data: The retrieval is characterized by a high flexibility, and thus, many parameters can be user-defined, and some feature can be adopted or not according to the user-defined settings.

The marc code is flexible, and an exhaustive user-defined setting of the retrieval is possible and easily achieved, according to user needs and measurement data quality. A default setting is in any case provided by means of a suitable setting file..

In the following paragraphs the input files necessary for MARC retrieval execution (see 5.3), the output files resulting from the Pre Processing procedure (see 5.5), a list of modalities and functionalities of the code and some indications on the available user-defined choices (see 5.4) will be addressed, together with the instruction to compile (see 5.1) and run (see 5.2) the code.

5.1 How to compile the source files of the MARC code

The MARC code is composed by some source files written in standard Fortran language.

> **!** Please notices that the executable code installed on the computer delivered to ESA has been already compiled by the MARSCHALS L2 team; if no change has been made to the source codes, to run the executable version a compilation is not required.

The source files of the MARC code can be compiled in the usual way by means of a whatever FORTRAN compiler. The computer delivered to ESA is provided with a FORTRAN compiler (from Portland company) that is able to perform this compilation task.
Once the compilation is completed, the executable code has to be moved in the adequate directory, that is, in the proposed standard configuration, the *BIN* one.

> The compilation procedure can be performed by using an adequate *makefile* file provided by the MARSCHALS L2 team together with the code files; this *makefile* file, residing in the *MARC_SOURCE* subdirectory, can be executed by means of the usual `make` command.

To use the *makefile* file, please move to the directory in which the source files and the *makefile* file are stored (the *MARC_SOURCE* directory in the standard configuration) and at the prompt type:

```
make
```

and then press the "**`enter`**" button.

> **Please notice that the `make` command works according to the standard arrangement of the file; the user can change this arrangement, but in this case, to use the make and the make install command, the makefile file have to be edited accordingly to the user defined arrangement of the files.**

5.2 How to run MARC code

Together with the executable MARSCHALS L2 software installed on the computer delivered to ESA, a batch file is provided to run the MARC code.

The user **must** run the MARC code by using the batch file *run_marc*, stored as a default in the MARC/WORKING_DIR subdirectory.

Please notice that the MARC code cannot be run directly from command line by means of itse executable file; as a matter of fact the use of the *run_marc* batch file is mandatory, since in such a file some user-defined parameter needed to run the code are specified.

To run the MARC code, please move to the subdirectory in which the *run_marc* batch file is stored (the MARC/WORKING_DIR one in the default arrangement) and at the prompt type:

```
./run_marc
```

and then press the "**enter**" button.

The user can change the default arrangements of the subdirectories in which the files needed to execute the *MARC* code are stored.

If a change is made in the default arrangement of the files (as reported in 3.1.2), the batch file *run_marc*, stored as a default in the MARC/WORKING_DIR subdirectory, have to be changed accordingly by the user.

Please notice that MARC code files are arranged in subdirectories. These subdirectories are contained in a "main directory" (named, as a default, "*MARC*") that is user defined; all the paths are thus defined in a relative way with respect to the subdirectory in which the executable files are contained, (that is the MARC/WORKING_DIR one in the default configuration).
The *run_marc* environment file contains information about the internal tree structure of the subdirectories in which the files needed to run the MARC code reside, thus, if a change is made to the delivered default configuration (as reported in 3.1.2), the *run_marc* file have to be changed accordingly by the user.

In the following the *run_marc* file is presented; each definition is shown and explained in the order of appearance inside the file. A default arrangement of the input files is described in 3.1.2.

5.2.1 The *run_marc* file

line in *run_marc* **file**	Related setting
`export MARC_IODIR=[ ]`	The `MARC_IODIR` variable indicates the path of the I/O files directory. The default configuration is: `export MARC_IODIR= ./` and indicates that the INP_FILES and OUT_FILES directory containing the I/O files will be placed directly in *the MARC/WORKING_DIR* directory.
`export MARC_PT=[ ]`	The *MARC_PT* variable indicates the path of the directory containing the file related to the temperature and pressure profiles of the atmosphere, that is the *in_zpt.dat* file. The default configuration is: `export MARC_PT=../AUX/IN_GUESS/Mid_Lat/` and indicates that the files will be placed in the *MARC*/AUX/*IN_GUESS/Mid_Lat* directory.
`export MARC_VMR= [ ]`	The *MARC_VMR* variable indicates the path of the directory containing the files related to the atmospheric VMR profiles of the species, that is set of *vmr_[species].dat* files. As an example: `export MARC_VMR=..AUX/IN_GUESS/Mid_Lat/` indicates that the atmospheric VMR profile files will be placed directly in the *MARC/AUX/IN_GUESS/Mid_Lat* directory.
`export MARC_SPECT=[ ]`	The *MARC_SPECT* variable indicates the path of the directory containing the files related to the spectroscopic database. As an example: `export MARC_SPECT=../AUX/DSPECT/` indicates that the data files will be placed in the *MARC/AUX/DSPECT* directory.

line in *run_marc* **file**	Related setting
`export MARC_CONT=[ ]`	The *MARC_CONT* variable indicates the path of the directory containing the files related to the continuum used in the MARC retrieval, i.e. *cont_band_[$].dat.* As an example: `export MARC_CONT=../AUX/CONT/` indicates that the continuum data files will be placed in the *MARC/AUX/CONT* directory.

`export MARC_Error_Spectra=[]`	The *MARC_Error_Spectra* variable indicates the path of the directory containing the files related to the error spectra used in the MARC retrieval. As an example:
	`export MARC_ Error_Spectra =`
	`=../AUX/ERROR_SPECTRA/`
	indicates that the error data files will be placed in the *MARC/AUX/ERROR_SPECTRA* directory.

`export IG_DIR= [ ]`	The *IG_DIR* variable indicates the path of the directory containing the file related to the irregular grid to be used (when specified) in the MARC retrieval, i.e. the *marschals.grd* file. As an example: `export IG_DIR=../AUX/IG/` indicates that the irregular grid data files will be placed in the *MARC/AUX/IG* directory.

`export MARC_REFIND=[ ]`	The *MARC_REFIND* variable indicates the path of the directory containing the file *refind.dat* related to the refractive indexes of water and ice used in the MARC retrieval. As an example:
	`export MARC_ REFIND =`
	`=../AUX/REF_IND/`
	indicates that the refractive indexes file will be placed in the MARC/AUX/*REF_IND* directory.

`export VERSION= [ ]`	The *VERSION* variable indicates the version of the code.

line in *run_marc* **file**	Related setting
`../BIN/marc > log_marc.txt`	This line specifies the standard output. In the case reported on the left, the standard output is redirected on `log_marc.txt` ASCII file.

Table 5-1. The parameters defined in the run file with their environment variables, and run of the MARC code.

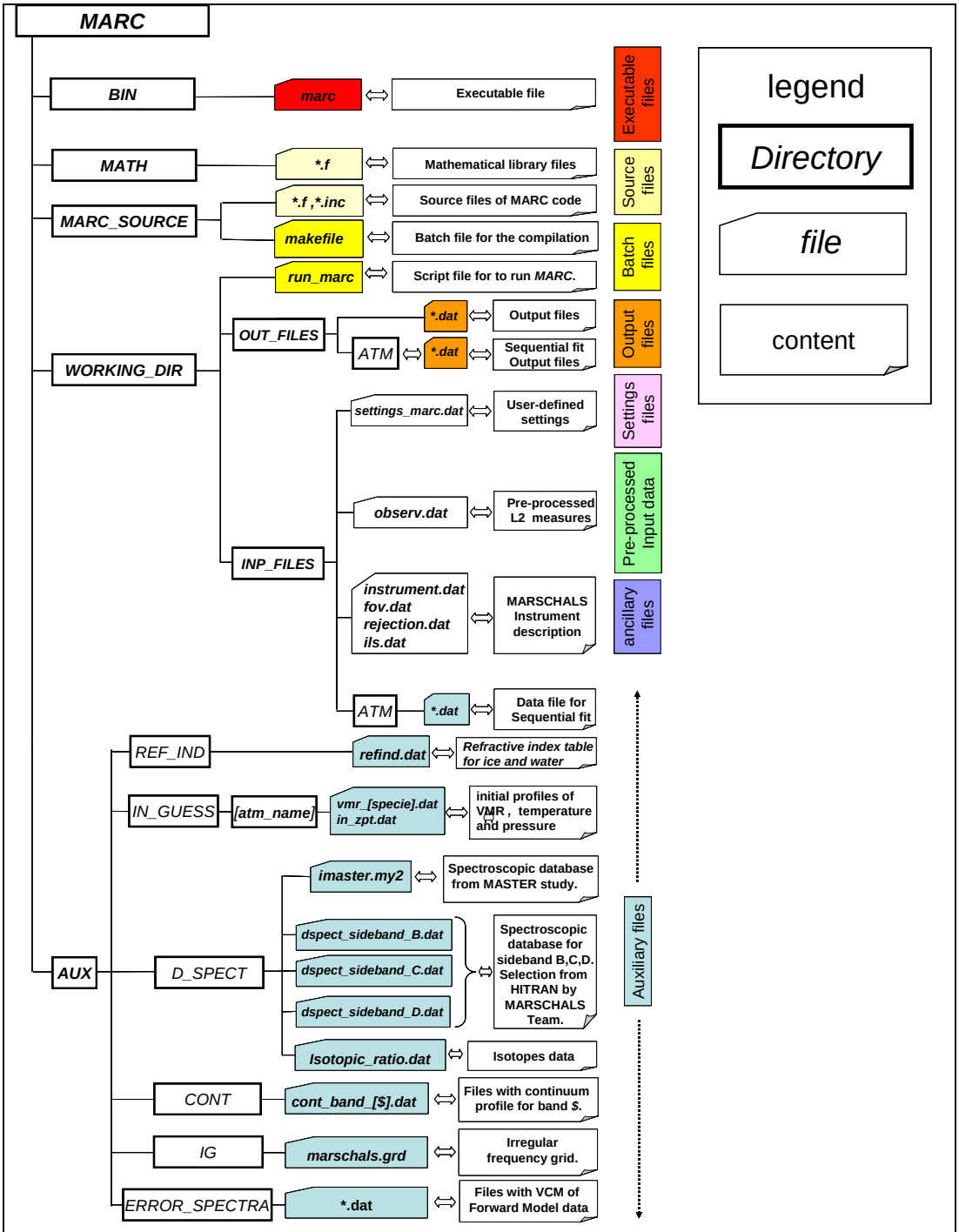

Figure 5-1. The default arrangement in subdirectories of the MARSCHALS L2 MARC files as specified in the delivered *run* file.

5.3 The files needed to run the MARC code

To run the MARC code some files of different kind are requested. These files are listed and classified in:

The data file needed to run the MARC code	
File classification	**Name of files**
source files(1)	**.f* (source files) and **.inc* (include files) standard FORTRAN language files. The whole list is reported in RD 3.
batch file	*makefile* - makefile to compile the MARC source files (see 5.1 for the MARC compilation procedure and RD 3 for the file format).
	run_marc - batch file to run the MARC code (see 5.3.2 and RD 3 for the file format).
executable file	*marc* - executable file of he MARC code. (see 5.3.3 and RD 3 for the file format).
pre-processed data files	*observ.dat* file with the pre-processed L1b data (see 5.3.4 and RD 3 for the file format).
ancillary data files	files with ancillary data (see 5.3.5 and RD 3 for the file format). • *instrument.dat* • *fov.dat* • *rejection.dat* • *ils.dat*
auxiliary data files	files with auxiliary data (see 5.3.6 and RD 3 for the file format). • *refind.dat* • *imaster.my2* • *dspect_sideband_$* • *isotopic_ratio.dat* • *marschals.grd* • *vmr_[species].dat* • *in_zpt.dat* • *cont_band_$* • *T_band_[$].dat* • *[species] _band_[$].dat* • *[species]_g_band_[$].dat* • *continuum_band_[$].dat* • *pointing_band_[$].dat.* • Error spectra data files (see RD 3)

The data file needed to run the MARC code	
File classification	**Name of files**
setting files	*settings_marc* file to set the MARC code run (see 5.3.7)
(1) These files are needed to create the *marc* executable file; anyway a *marc* executable file has been already created and installed in the computer delivered to ESA; thus these files are not requested to run the provided default version of the MARC code	

Table 5-2. The data file needed to run the MARC code.

The aforementioned files have to be stored in the hard disk of the computer devoted to run the MARC code, according to a given files arrangement. A default arrangement suggested by the MARSCHALS L2 team is described in paragraph 3.1.

The computer delivered to ESA is provided with the entire set of files needed to run the MARC code, stored according to the default arrangement presented in paragraph 3.1

5.3.1 The source files

The MARC code is based on a set of source files (*.*f* source files , *.*inc* include files) written in standard FORTRAN language. The whole list is reported in RD 3.

5.3.2 The batch files

The *makefile* file can be used to compile the source files and, after the compilation, to move the executable in the right directory, according to the default arrangement described in chapter 3.1.2 (for the compilation procedure of the MARC code see 5.1).

> Please notices that a *marc* executable file has been already produced and installed in the computer delivered to ESA; thus to run the proposed default code a compilation is not required.

The batch file *run_marc* is provided for the running of the executable code stored in the *marc* file (see 5.2).

> The *run_marc* file the running of the executable code is provided with the other MARC files inside the computer delivered to ESA, stored (in the default configuration) in the *MARC/WORKING_DIR* subdirectory.

5.3.3 The executable file

The *marc* executable file, obtained by compiling the SAMM code source files according to the default setting is provided inside the computer delivered to ESA (stored in the *MARC/WORKING_DIR* subdirectory).

5.3.4 The pre-processed data files

The MARC code operates on the data coming from the L1b measurement, preprocessed by the SAMM code. As a matter of fact, it operates on the data stored in the *observ.dat* file, a file that is obtained by the user from the set of the *observation_§.dat* files produced by the SAMM code (being § the scan) by choosing one of these file (that related to the desired scan) and renaming it with the name *observ.dat*.
For more details on these files see RD 3.

5.3.5 The ancillary data files

The MARC code of the MARSCHALS software needs also some information related to the processed measure, i.e. to some parameters and data related to the acquisition of the data under consideration. This information is stored in four files:

- *instrument.dat*
- *fov.dat*
- *rejection.da*
- *ils.dat*

that, in the default configuration, are stored in the MARC/WORKING_DIR/INP_FILES subdirectory.

Please notice that a set of ancillary data files are provided together with the other files of the MARSCHALS L2 suite. They are referred to the provided test data; if the codes of the MARSCHALS L2 suite are run with different measurement data, the provided ancillary data could not be adequate to the new measurement scenario. In this case, a new and suitable set of ancillary data files have to be provided by the user.

5.3.6 The auxiliary data files

Some auxiliary files are needed to run MARC retrieval code. These files are not provided by the pre-processor, and have to be provided by the user. These files are listed in Table 5-3 and described in RD 3.

> Please notice that a set of ancillary data files are provided together with the other files of the MARSCHALS L2 suite.

MARC auxiliary data files	
refind.dat	File with refractive index table for ice and water.
imaster.my2	file with the spectroscopic database for the main bands.
dspect_sideband_B *dspect_sideband_C* *dspect_sideband_D*	file with the spectroscopic database for the image bands.
isotopic_ratio.dat	file with isotopic ratio information.
vmr_[species].dat	files with information about VMR profile.
in_zpt.dat	file with information about pressure and temperature profile.
cont_band_B *cont_band_C* *cont_band_D*	files with information about atmospheric continuum profile.
T_band_[$].dat; *[species] _band_[$].dat*; *[species]_g_band_[$].dat*; *continuum_band_[$].dat*; *pointing_band_[$].dat;* Error spectra data files.	files with information about systematic errors.
marschals.grd	file with information about frequency irregular grid.

Table 5-3. The list of the MARC auxiliary data files.

5.3.7 The setting file

The user defined set up of the MARC code execution for the retrieval procedure are specified in the *settings_marc.dat* file.
The set up options offered to the user are detailed in paragraph 5.4.

5.4 How to set up the marc code execution

The retrieval performed by the MARC code offers many options that can be set by the user; at the same time, to run the code, the user has to fix many parameters to set the algorithms of the retrieval procedure.

> **The selection of the possible options and the setting of the retrieval procedure has to be made by editing the related settings files in ASCII format with a common text editor.**

In the following paragraphs of this chapter a selection of the most important settings is provided. As a general rule, it is possible to highlight that in the *settings_marc.dat* file the data are arranged in rows; all data are in ASCII format and are separated by a number of blanks. Commented rows are preceded by a #. Blank rows have not to be considered. The format of the data is the following:

> a commented row (starting with a #) introduces the variable under definition; then a string that identifies the variable that is defined is written in a not commented row between square brackets; the related quantity is reported in the following single non commented row. Notice that the variable could be vectorial.
> A string `"[end_file]"` ends the file.

Because of the presence of the key-word string, the order of the definition is not matter of care; in any case, for sake of clarity, the parameters have been grouped according to their meanings.

5.4.1 How to set the release of the run of the retrieval code

The keyword ***[settings_key*** is used to define a string containing the file release.

Key-words	related quantity	dim.	data type
[settings_key]	String specifying the release of the file	dimless	char*10

Since the name of the setting file has to be the one specified in the *run_marc* file, it would not be easy to change this name inside a series of executions of the code. Thus, in order to allow a tracking of different runs of the code with different parameters (i.e. with different *settings_marc.dat* files), the key-word `[settings_key]` has been introduced in the file to distinguish different set up (i.e. different file with the same name *settings_marc.dat*). When a run of the code is repeated with a different set up specified in the *settings_marc.dat* file, the user can change the string associated to the key-word `[settings_key]`. A `[settings_key]` key-word is provided for each input file, the main code is able to take into account the version of all the files that are read during the run, and an array with these version numbers is available in the output data.

5.4.2 How to run the MARC code as a retrieval procedure or a simple forward model

The MARC code is designed to perform a retrieval procedure, but it can be used also as a simple Forward Model.
The **[lmfm]** flag must be set to decide if the code has to be run as a simple Forward Model or a Retrieval Code.

Key-words	related quantity	dim.	data type
[lmfm]	Flag (F/T) to define if the code is used as retrieval code or forward model: T → MARC works in forward model modality; F → MARC works in retrieval model modality.	dimless	logical

5.4.3 How to set the bands to be used in the retrieval procedure

The MARC code performs a multi-band retrieval on the three MARSCHALS band (B, C and D), but it can also performs the retrieval on a whatever selection of the aforementioned bands. To select the bands on which the retrieval has to be performed the **[iband]** flag has to be set.

Key-words	related quantity	dim.	data type
[iband]	Array with three logical values (T/F) specifying the bands on which the retrieval is performed: T → the related band is considered; F → the related band is not considered. the order of the bands inside the array is the following: 1 → band_B ; 2 → band_C ; 3 → band_D . As an example, the setting: … # [iband] TFT … means that bands B and D are considered in the retrieval, whereas C is not considered.	dimless	3×logical

This setting is considered also when the MARC code works as a simple Forward Model.

5.4.4 How to set the quantities to be retrieved

The MARC code is able to perform a multi-target retrieval of various atmospheric species and temperature at the same time. The quantities to be retrieved can be specified by settings the values of some key-words in the setting file.
The specification is made by setting the following keyword:

Key-words	related quantity	dim.	data type
[ltempfit]	Switch (logical value F/T) for the retrieval of the temperature values: T → the temperature is retrieved; F → temperature is not retrieved.	dimless	logical
[lvmrfit]	Switch (F/T) for the retrieval of the VMR values: T → at least the VMR of one species is retrieved; F → VMRs are not retrieved.	dimless	logical
[NfitVMR]	Number of gases whose VMR is retrieved. **Note** that the *NfitVMR* value is read only if *lvmrfit* is *True*.	dimless	integer
[codefitVMR]	Array of integers with the MARSCHALS/MASTER codes of the molecules to be retrieved. Note that only the first *nfitvmr* codes of the arrays are read. The MARSCHALS/MASTER code is composed by numbers that indicate the species: matching number of the species is specified in the following Table 5-4. Retrieval tests have been performed for H_2O, O_3, HNO_3, CO, N_2O, CH_3Cl, (see RD 4).	dimless	Nfitvmr × integer

The MARSCHALS/MASTER code corresponds to the HITRAN code, and is composed by numbers that indicate the species: matching number <-> species is specified in the Table 5-4:

Key-words	related quantity	dim.	data type
[lgvmrfit]	Switch (logical value F/T) for the retrieval of the gradients values: T → at least the VMR gradient of one species is retrieved; F → the VMR gradients are not retrieved.	dimless	logical
[NfitGRAD]	Number of species whose gradient is retrieved. Note that the *NfitGRAD* value is read only if *lgvmrfit* = = "T". If *lvmrfit* is true, it has to be *NfitGRAD* ≤ *NfitVMR* .	dimless	integer*4
[codefitGRAD]	Array of integers containing the MARSCHALS/MASTER code of the species whose gradients have to be retrieved. **Note** that only the first *NfitGRAD* codes of the arrays are read. The HITRAN code is that specified in Table 5-4. If *lvmrfit* is true, *codefitGRAD* has to be a subset of *codefitVMR*. Retrieval tests have been performed for gradient of O_3 and gradient of H_2O (see RD 4).	dimless	nfitgrad × integer

Key-words	related quantity	dim.	data type
[lpoifit]	Switch (logical value F/T) for pointing bias retrieval: T → retrieval of pointing bias is performed; F → retrieval of pointing bias is not performed.	dimless	logical
[ig_pointing]	Initial guess of pointing angle bias.	deg.	real
[lexcfit]	Switch (logical value F/T) for external continuum retrieval: T → retrieval of un-accounted continuum is performed; F → retrieval of un-accounted continuum is not performed.	dimless	logical
[loffsfit]	Switch for instrumental offset retrieval: T → retrieval of offset is performed; F → retrieval of offset is not performed.	dimless	logical
[ig_offset]	initial guess of the instrumental offset for each fitted band.	K	3 × real
[lgainfit]	Switch for gain retrieval: T → retrieval of gain is performed; F → retrieval of gain is not performed.	dimless	logical
[ig_gain]	initial guess of gain for each fitted band.	dimless	3 × real

This setting is considered also when the MARC code works as a simple Forward Model.

MARSCHALS/MASTER (HITRAN) code				
1 = H_2O	2 = CO_2	3 = O_3	4 = N_2O	5 = CO
6 = CH_4	7 = O_2	8 = NO	9 = SO_2	10 = NO_2
11 = NH_3	12 = HNO_3	13 = OH	14 = HF	15 = HCl
16 = HBr	17 = Hl	18 = ClO	19= OCS	20 = H2CO
21 = HOCl	22 = N_2	23 = HCN	24= CH_3C	25 = H_2O_2
26 = C_2H_2	27 = C_2H_6	28 = PH_3	29= COF_2	30 = SF_6
31 = H_2S	32 = HCOOH	33 = HO_2	34 = O	35 = $ClONO_2$
36 = NO+	37 = HOBr	38 = C_2H_4	39= CH_3OH	40 = Br0

Table 5-4: the MARSCHALS/MASTER (HITRAN) code for the main molecules

Please notice that the order of the species to be retrieved is rearranged by the code using the following modality.
The list starts with the species for which both the VMR and the gradient are to be retrieved; among these species, the order is that specified by *codefitGRAD*. Then, the list goes on with the species whose gradient is not to be retrieved, listed according to the order specified in *codefitVMR*.
Trivially, if any gradient is not retrieved (*lgvmrfit* = F), the order is that specified by the user with *codefitVMR*; on the contrary, if for all the retrieved specie both the VMR and the gradient are retrieved, the order is the one specified in by *codefitGRAD*.
As an example of a generic case, if

codefitVMR = 12 3 1 5 4

and

codefitGRAD = 1 3

the resulting order will be:

1 3 12 5 4

5.4.5 How to set the retrieval grid

The MARC retrieval can be performed at the tangent altitudes or on a user defined altitude grid. In the last case, the user has to specify an overall altitude grid that is common for all the retrieved quantities; then a dedicated retrieval grid can be specified for each retrieved quantity as a subset of the common grid previously defined, by specifying this subset with a "mask" array.

The setting of the retrieval grid is made by set the following key-words:

Key-words	**related quantity**	**dim.**	**data type**
[ltangaltret]	Switch (F/T) to specify the choice of the retrieval altitude grid: T → retrieval at tangent altitude; F → retrieval at user defined altitude grid. Please notice: in Forward Model option, when MARC is used in Forward Modality, the *ltangaltret* switch must be set to "F".	dimless	logical
[nretrievalgrid]	Number of altitude points of the retrieval grids (user-defined and tangent-altitude grids).	dimless	integer
[rretrievalgrid]	Array (values) with the altitudes of the retrieval grid in decreasing order; available range (0,*rulatm*)). Note: only the first *nretrievalgrid* values in the array are read.	Km	nretrievalgrid × real

Key-words	related quantity	dim.	data type
[ltmask]	Array (T/F) with *nretrievalgrid* flag (one for each altitude of the retrieval grid) specifying the mask for temperature retrieval; T → the temperature is retrieved at the related altitude; F → the temperature is not retrieved at the related altitude. note: only the first *nretrievalgrid* codes in the array are read.	dimless	nretrievalgrid × logical
[lvmrmask]	Matrix with *NfitVMR* rows, one for each retrieved species (in the rearranged order specified in 5.4.4). Each row is composed by *nretrievalgrid* flag (one for each altitude of the retrieval grid) specifying the mask for the VMR retrieval: T → the VMR, the row is related to, is retrieved at the related altitude; F → the VMR, the row is related to, is not retrieved at the related altitude. Note: • only the first *nretrievalgrid* codes in the input line are read • only *NfitVMR* lines corresponding to the fitted VMR are read.	dimless	Nfitvmr × nretrievalgrid × logical

Key-words	related quantity	dim.	data type
[lvmrgmask]	Matrix (logical values F/T) with `NfitGRAD` rows, one for each retrieved gradient (in the rearranged order specified in 5.4.4). Each row is composed by *nretrievalgrid* flag (one for each altitude of the retrieval grid) specifying the mask for the retrieval of the gradient of the VMRs: T → the VMR gradient, the row is related to, is retrieved at the related altitude; F → the VMR gradient, the row is related to is not retrieved at the related altitude. Note: • only the first *nretrievalgrid* codes in the input line are read • only *NfitGRAD* lines corresponding to the fitted gradient are read.	dimless	logical
[lcmask]	Array (logical values F/T). This row is composed by *nretrievalgrid* flag (one for each altitude of the retrieval grid) specifying the mask for the retrieval of the un-accounted continuum: T → the un-accounted continuum is retrieved at the related altitude; F → the un-accounted continuum is not retrieved at the related altitude.	dimless	nretrievalgrid × logical

This setting is considered also when the MARC code works in Forward modality.

5.4.6 How to set the atmospheric parameters

The retrieval procedure is based on a forward model that uses an atmospheric model; the user can define some features and parameters of this atmospheric model, such as the used reference system, the boundaries of the atmosphere, the presence of horizontal gradient, the possibility to take into account the continuum of certain species, the imposition of the hydrostatic equilibrium, the presence of scattering phenomena and some instrument parameters.
The aforementioned settings have to be specified by defining the values related to the following key-words in the setting file.

Key-words	related quantity	dim.	data type
[hydroeq]	Switch (T/F) to impose (if True) the hydrostatic equilibrium: T → hydrostatic equilibrium is imposed; F → hydrostatic equilibrium is not imposed.	dimless	logical
[rzpr]	Number specifying the reference altitude for imposition of hydrostatic equilibrium.	Km	real
[lgrad]	Switch (T/F) for the use of the Horizontal Gradients: T → horizontal gradients are used: (2D atmosphere); F → horizontal gradients are not used: (1D atmosphere).	dimless	logical
[lextcont]	Switch (T/F) for the use of external continuum in the Forward Model procedure: T → the external continuum is used; F → the external continuum is not used.	dimless	logical

Key-words	related quantity	dim.	data type
[nullcont_above]	Altitude above which the atmospheric continuum is forced to 0. It has to be: *nullcont_above* < *rulatm*	Km	real
[originref]	Array defining the Reference Systems Origin ([*izref*]]; [*rzref*]). It is composed by one or two values; the former being [*izref*], the possible latter being [*rzref*]; both of them are defined in the following.		
[izref]	Integer specifying the kind of Reference System among some possibilities, according to the following code: 1 = Reference Vertical passing through the instrument location; 2 = Reference Vertical passing through the lowest tangent point, in geometrical approximation (greater then 0); 3 = Reference Vertical at intermediate tangent altitude; (in geometrical approximation); 4 = user defined (with respect to the instrument location). If the option 4 (user defined vertical reference) is specified, the user defined displacement with respect to the vertical of the instrument has to be defined; it is specified in the second element *[rzref]* of the array. Trivially, this element is present only if the option 4 is specified; if *izref*<4 *rzref* is ignored.	dimless	integer

Key-words	related quantity	dim.	data type
`[rzref]`	Displacement of the Reference Vertical with respect to the vertical of the instrument (in the sense of the line of sight).	degrees	real
`[lmssf]`	Switch (T/F) for the use of the Mie Scattering Source function (MSSF): T → MSSF data are used; F → MSSF data are not used.	dimless	logical

This setting is considered also when the MARC code works in Forward modality.

5.4.7 How to set up the retrieval modality and techniques

The MARC code allows performing the retrieval procedure by using different techniques and modalities; in addiction to the Marquardt iteration techniques, also Regularization or an Optimal Estimation technique can be set. The user can specify the use of these techniques by defining the values related to the following key-words in the setting file.

Key-words	related quantity	dim.	data type
[imode]	Switch to select the retrieval mode; available range (1, 2, 3): 1 → Marquardt only; 2 → Regularization + Marquardt; 3 → Optimal Estimation + Marquardt. If option 2 is chosen, the set up of the regularization has to be performed, by defining the related parameters.	dimless	integer*4

Then, the chosen modalities have to be set by defining the related parameters; these parameters have to be specified by setting the related key-words.

- To set up the Gauss and Marquardt iterations see 5.4.8;
- To set up the Regularization technique see 5.4.9;
- To set up the Optimal Estimation technique see 5.4.10.

5.4.8 How to set up the Gauss and Marquardt iterations

The retrieval procedure is based onto two nested iterations, the Gauss one (macro-iteration) and the Marquardt one (micro-iteration).
The following key-words have to be defined to set the Marquardt and the Gauss iterations:

Key-words	related quantity	dim.	data type
[rlambdat]	Initial value for lambda parameter in Marquardt iterations used for the retrieval of the Temperature.	dimless	real
[rlambdav]	Array containing the initial values for λ parameters in Marquardt iterations used for the VMRs retrieval; the λ parameter has been considered as gas dependent, thus a values is given for each retrieved VMR; the order inside this array is that defined in 5.4.4.	dimless	Nfitvmr × real
[rlambdavg]	Array containing the initial values for λ parameters in Marquardt iterations used for the retrieval of the VMR gradients; the λ parameter has been considered as gas dependent, thus a value is given for each retrieved gradient; the order inside this array is that defined in 5.4.4.	dimless	Nfitgrad × real
[rlambdapoi]	Initial value for the λ parameter in the Marquardt iterations used for the retrieval of the pointing angle offset.	dimless	real
[rlambdaexc]	Initial value for λ parameter in the Marquardt iterations used for the retrieval of the external continuum.	dimless	real
[rlambdaoff]	Initial value for λ parameter in the Marquardt iterations used for the retrieval of the instrumental offset.	dimless	real

Key-words	related quantity	dim.	data type
[rlambdagain]	Initial value for λ parameter in the Marquardt iterations used for the retrieval of the instrumental gain.	dimless	real
[rgaussDumpingFactor]	Dumping Factor for Gauss iterations (macro-iteration).	dimless	real
[rMarquardtDumpingFactor]	Dumping Factor for Marquardt iterations (micro-iteration).	dimless	real
[Threshold_on_Weighted_Chisq_increment]	Threshold to activate the Marquardt micro-iteration.	dimless	real

5.4.9 How to set up the regularization procedure

The MARC retrieval code can be performed by considering a Regularization procedure. The Regularization can be separately applied to single retrieved quantities. To set up the Regularization procedure the following key-words have to be defined.

Key-words	related quantity	dim.	data type
[reg_tuning]	Parameter for tuning profile regularization.	dimless	real

TEMPERATURE

Key-words	related quantity	dim.	data type
[reg_diagT]	Switch (T/F) for the use of regularization matrix in temperature retrieval: T → regularization matrix is used in temperature retrieval; F → regularization matrix is not used in temperature retrieval.	dimless	logical

VMR

Key-words	related quantity	dim.	data type
[reg_diagVMR]	Array with the flags (T/F) related to the use of regularization matrix in VMR retrieval. A value is provided for each retrieved species, in the order specified in sect. 5.4.4: T → regularization matrix is used in VMR retrieval of the related species; F → regularization matrix is not used in VMR retrieval of the related species.	dimless	logical

VMR GRADIENT

Key-words	related quantity	dim.	data type
[reg_diagGRAD]	Array with the flags (T/F) related to the use of regularization matrix in VMR gradient retrieval. A value is provided for each retrieved species, in the order specified in 5.4.4: T → regularization matrix is used in VMR gradient retrieval of the related species; F → regularization matrix is not used in VMR gradient retrieval of the related species.	dimless	logical

UN-ACCOUNTED CONTINUUM

Key-words	related quantity	dim.	data type
[reg_diagCONT]	Array with the flags (T/F) related to the use of regularization matrix in un-accounted continuum retrieval. A value is provided for each band: T → regularization matrix is used in un-accounted continuum retrieval of the related band; F → regularization matrix is not used in un-accounted continuum retrieval of the related band.	dimless	logical

5.4.10 How to set up the optimal estimation procedure

To set up the optimal estimation procedure, the following parameters have to be defined:

Key-words	related quantity	dim.	data type
[opt_pointing]	Absolute error on a priori information on pointing angle.	deg.	real
[opt_extCONT]	Absolute error on a priori information on Un-accounted Continuum.	cm^2x10^{27}	real
[opt_gain]	Absolute error on a priori information on gain.	dimless	real
[opt_ofs]	Absolute error on a priori information on offset.	K	real

5.4.11 How to set the convergence and stop criteria for retrieval

The MARC code performs a retrieval process based on a iterative procedure; five convergence criteria have been defined to check the convergence of the iterations. The user can choose which convergence criterion has to be used; a combination of more than one convergence criteria can be also defined (the selected criteria can be checked according to user-defined AND or OR logic operator). The user has to choose the convergence criteria that have to be used, and the logical combination of these criteria. The user has also to set some parameters of the selected convergence criteria. In order to avoid endless loop, a stop criterion to terminate the iterative procedure after a given number of Gauss or Marquardt iterations is also provided; the corresponding maximum number of allowed iterations is user-defined. When the maximum number of Marquardt iteration is reached, the code moves itself to the next Gauss iteration. Conversely, if the max number of Gauss iteration is reached, the iteration procedure is stopped even if the convergence is not reached.
The five implemented convergence criteria are the following:

1st convergence criterion (relative variation of χ^2)

The relative variation of the χ^2 function obtained in the present iteration with respect to the previous iteration is less than a fixed threshold t_1 , i.e.:

$$\left| \frac{\chi^2(\mathbf{x}^{iter-1}) - \chi^2(\mathbf{x}^{iter})}{\chi^2(\mathbf{x}^{iter})} \right| < t_1$$

where *iter* is the current iteration index.
A drawback of this criterion is that small variations of the χ^2 function, and even growth, may be caused by non-linearity.

2nd convergence criterion (maximum relative correction)

The maximum correction that has to be applied to the elements of the state vector is below a fixed threshold t_2 i.e.:

$$\mathrm{Max}_j \left| \frac{(\mathbf{x}^{iter-1})_j - (\mathbf{x}^{iter})_j}{(\mathbf{x}^{iter})_j} \right| < t_2$$

Different levels of sophistication can be applied to this criterion. A different threshold can be used for the different elements of the state vectors. The absolute variations can be considered instead of the relative variations, whenever an absolute accuracy requirement is present for the elements of the state vector. A distinction can be made between target elements (quantities for the determination of which the measurements are performed) and non-target elements (quantities that are determined in order to improve the quality of the fit, but are not the objective of the measurement, e.g. the instrument parameters).

3rd convergence criterion (maximum relative correction referred to error)

Since expression of the 2nd convergence criterion is singular whenever an element of the state vector is equal to zero, a modified expression that can be considered is:

$$\mathrm{Max}_j \left| \frac{(\mathbf{x}^{iter-1})_j - (\mathbf{x}^{iter})_j}{\sqrt{(\mathbf{S}^x_{iter})_{j,j}}} \right| < t_3$$

where $\sqrt{\left(\mathbf{S}^x_{iter}\right)_{j,j}}$ represents the error associated with the element of state vector $(\mathbf{x}^{iter})_j$ at iteration *iter*. The consideration that may prevent the unconditional use of the above expression instead of that of the 2nd criterion is that $\sqrt{\left(\mathbf{S}^x_{iter}\right)_{j,j}}$ does not really represent the error of the element $(\mathbf{x}^{iter})_j$ when the retrieval is still far from convergence.

4th convergence criteria (relative difference between χ^2 and linear χ^2

The difference between the current chi-square $\chi^2\left(\mathbf{x}^{iter}\right)$ and the chi-square computed in the linear approximation (χ^2_{LIN}) is less than a fixed threshold t_4:

$$\left| \frac{\chi^2(\mathbf{x}^{iter}) - \chi^2_{LIN}(\mathbf{x}^{iter})}{\chi^2(\mathbf{x}^{iter})} \right| < t_4$$

Where χ^2_{LIN} is computed using the expression:

$$\left((\mathbf{I}-\mathbf{KG})\left(\mathbf{y}-\mathbf{F}\left(\mathbf{b},\tilde{\mathbf{x}}\right)\right)\right)^T \left(\mathbf{S}^M\right)^{-1} \left((\mathbf{I}-\mathbf{KG})\left(\mathbf{y}-\mathbf{F}\left(\mathbf{b},\tilde{\mathbf{x}}\right)\right)\right)$$

5th convergence criteria (Marquardt parameter)

The value of Marquardt parameter (λ_M) have to be less than a fixed threshold (t_5), for each retrieved parameter.

$$\lambda_M i < t_5 \,,\ \forall i \qquad (5.4.1)$$

Convergence is reached only if each Marquardt parameter is less than the set threshold

Combination of the convergence criteria

The overall convergence criterion is obtained as a suitable combination of the above conditions.
In some case the convergence criteria can be satisfied also when the iterative process is still far from reaching convergence. In order to avoid this problem, if the computing time of an extra iteration does not pose problems, the requirement of a double contiguous occurrence of satisfied criterion may be required for convergence.

To avoid endless loop, two stop criteria have been considered.
The adopted stop criteria is the following

Stop criteria

The code exit from the Marquardt iteration and moves itself to the next Gauss iteration when the maximum number of Marquardt iteration (iMaxiterM) is reached:

$$iterM \geq iMaxiterM \qquad (5.4.2)$$

Being *iterM* the number of Marquardt iteration:

The code exit from the Gauss iteration and ends the retrieval procedure even if the convergence is not reached when the maximum number of Marquardt iteration (`iMaxiterM`) is reached:

$$iterG \geq iMaxiterG \qquad (5.4.3)$$

being $iterG$ the number of Gauss iteration:

The setting of the aforementioned convergence and stop criteria is made by setting the values related to the following key-words in the setting file.

1st Convergence Criterium

Key-words	**related quantity**	**dim.**	**data type**
[conv_criterium1]	Array with two elements [[`rthreshold1`], [`lconv1`]].		
[rthreshold1]	Percent variation of the chisquare between two subsequent iterations.	dimless (%)	real
[lconv1]	Switch (T/F) for the convergence criterium: T → 1st Convergence Criterium is considered; F → 1st Convergence Criterium is not considered.	dimless	logical

2nd Convergence Criterium

Key-words	related quantity	dim.	data type
[conv_criterium2]	Array with two elements [[rthreshold2], [lconv2]].		
[rthreshold2]	Maximum percent correction to state vector (in percentage).	dimless (%)	real
[lconv2]	Switch (T/F) for the convergence criteria. T → 2nd Convergence Criterium is considered; F → 2nd Convergence Criterium is not considered.	dimless	logical

3rd Convergence Criterium

Key-words	related quantity	dim.	data type
[conv_criterium3]	Array with two elements [[rthreshold3], [lconv3]].		
[rthreshold3]	Maximum percent correction to modified state array modified (in percentage).	dimless (%)	real
[lconv3]	Switch (T/F) for the convergence criteria. T → 3rd Convergence Criterium is considered; F → 3rd Convergence Criterium is not considered;	dimless	logical

4th Convergence Criterium

Key-words	related quantity	dim.	data type
[conv_criterium4]	Array with two elements [[rthreshold4], [lconv4]].		
[rthreshold4]	Percent variation between chisquare and LINEAR chisq (in percentage).	dimless (%)	real
[lconv4]	Switch (T/F) for the convergence criterium. T → 4th Convergence Criterium is considered; F → 4th Convergence Criterium is not considered.	dimless	logical

5th Convergence Criterium

Key-words	related quantity	dim.	data type
[conv_criterium5]	Array with two elements [[rthreshold5], [lconv5]].		
[rthreshold5]	Threshold on Marquardt parameter. Convergence is reached only if each Marquardt parameter is less than the set threshold	dimless	real
[lconv5]	Switch (T/F) for the convergence criterium. T → 5th Convergence Criterium is considered; F → 5th Convergence Criterium is not considered.	dimless	logical

Stop Criteria

Key-words	related quantity	dim.	data type
[iMaxIterG]	Maximum number of allowed Gauss iterations (Macro-iteration).	dimless	int
[iMaxIterM]	Maximum number of allowed Marquardt iterations (Micro-iteration).	dimless	int

Combination of the Convergence Criteria

Key-words	related quantity	dim.	data type
[OR_AND]	Switch for an "AND" or an "OR" combination of the convergence criteria: 0 → "AND" logic operation is performed among the results of the selected Convergence Criteria; 1 → "OR" logic operation is performed among the results of the selected Convergence Criteria.	dimless	integer

5.4.12 How to set the use of the Variance Covariance Matrix (VCM) of the Forward Model

The use of the Variance Covariance Matrix can be specified by the user by setting the values related to the **[lvcmfm]** key-word in the setting file.

Key-words	related quantity	dim.	data type
[lvcmfm]	Switch (T/F) for the use of FM VCM: T → FM VCM is used; F → FM VCM is not used.	dimless	logical

If lvcmfm is TRUE, MARC computes the VCM of the Forward Model using the files listed in RD 3. The default directory where these files are located in reported in **Figure 5-1**.

5.4.13 How to set the sequential fit procedure

The MARC code is able to perform a sequential fit, i.e. to carries out a retrieval in which the initial guess is constituted by the result of the previous retrieval and a completed VCM of *a-priori* information is used.
To impose the sequential fit procedure, the user has to set to T the value related to the **`[lseq]`** key-word of the setting file.

Key-words	related quantity	dim.	data type
`[lseq]`	Switch (T/F) to select the sequential fit procedure: T → the sequential fit is performed; F → the usual (not sequential) fitting procedure is used.	dimless	logical

Notice that the sequential fit performs some actions:

- it exports each retrieved profile in input format;
- it exports VCM for each retrieved profile;
- it reads VCM for each retrieved profile if found.

The default directory where these files are located in reported in **Figure 5-1**.

5.4.14 How to set the internal Forward model of the MARC code

The retrieval procedure of the MARC code is based on an internal Forward Model, that can be set by the user by adequately specifying the values related to the following key-words in the setting file.
The key-words of this section are read only if the Forward Modality is set (`lmfm` = "F").

Key-words	related quantity	dim.	data type
[itot]	Number of Lines of Sight (LOS) to be simulated.	dimless	integer
[fly_altitude]	Array containing the altitudes of the instrument for each LOS. Note: only `itot` values are read in the array.	Km	itot×real
[fly_angles]	Array containing the pointing limb view angles for each LOS.	deg.	itot×real
[kind_of_noise]	Index to select the Noise Model to be superimposed on the simulated spectrum, according to the code above: 0 → no noise is superimposed; 1 → uniform completely random noise is superimposed; 2 → uniform standard random noise is superimposed; 3 → gaussian completely random noise is superimposed; 4 → gaussian standard random noise is superimposed.	dimless	integer
[lradius]	Switch (T/F), to specify the model of the Earth Radius: T → computed from latitude; F → read below.	dimless	logical
[rearad]	Local radius of the Earth.	Km	real

Key-words	related quantity	dim.	data type
[rlat]	Latitude of the simulations.	deg	real
[lref]	Switch (T/F) to define the export of the atmospheric profiles used for the simulations: T → external profiles are exported; F → external profiles are not exported.	dimless	logical

5.4.15 The setting of the MSSF module

The key-words of this section are read only if the Mie Scattering Source function is used (*lmssf* = "F").

Key-words	related quantity	dim.	data type
[ipol]	Switch polarization index for B, C, D band: -1 → polarization not defined; 0 → vertical polarization plane; 1 → horizontal polarization plane.	dimless	integer*4
[size1]	Radius of water particles..	cm	real*8
[size2]	Radius of ice particles.	cm	real*8
[chem]	Ice fraction.	dimless	real*8
[rzcloud]	Altitude of the top of the clouds.	Km	real*8
[rdzcloud]	Vertical geometrical thickness of the clouds.	Km	real*8
[rdncloud]	Numerical density of the particles inside the clouds.	cm^{-3}	real*8
[nag]	Parameter used to build the number of direction used to model the solid angle surrounding the clouds. As a matter of fact, the solid angle is sampled by means of $(2 \times nag+1)^2$ directions.	dimless	integer*4

5.4.16 How to define the parameters for the advanced settings

Some advanced settings can be selected in MARC code; these settings can be specified by the user by defining the values related to the following key-words in the setting file.

Key-words	related quantity	dim.	data type
[ldebug]	switch (T/F) to enable the DEBUG option: T → debug is ENABLED; F → debug is DISABLED.	dimless	logical
[rulatm]	Value specifying the altitude of the upper boundary of the atmosphere.	Km	real
[rdaup]	Altitude step to complete the base grid above retrieval altitudes.	Km	real
[rdzda]	Array with two components: - the former specifies the altitude step to be added to the maximum flight altitude to obtain the altitude under which the grid has to be thickened; - the latter specifies the maximum step in the thickened grid.	Km	real
[nfov]	Minimum number of pencil beams to be used in the FOV simulation inside the finer grid (should be an odd number). The value: 2*rfovamp/(nfov-1) gives the maximum distance between two pencil beams inside the higher precision region of the FOV.	dimless	integer

Key-words	related quantity	dim.	data type
[rfov]	Array with parameters related to the FOV function; it is constituted by three values: [rfov] = [[rfovampmax],[rfovamp] ,[rzdfov]].		
[rfovampmax]	Number specifying the maximum extension for the FOV convolution (Half Width).	deg.	real
[rfovamp]	Number specifying the amplitude of the finer description of the FOV-function (Half Width). (In the current version *rfovampmax* and *rfovamp* cannot be equal)	deg.	real
[rzdfov]	Number specifying step in altitude between pencil beam (at tangent point).	Km	real
[lcont]	Array containing three logical (T/F) switches to specify if the retrieval procedure has to be performed by taking into account for the continuum the contribute of the species H_2O, O_2 and N_2 respectively.	dimless	logical
[lirrgrd]	Switch (T/F) to choose the frequency grid for the fine spectrum simulation: T → irregular grid is used; F → regular grid is used.	dimless	logical
[lray]	Switch (T/F) to define the ray tracing computation modality: T → computation is done for each iteration; F → computation is done only for the first evaluation of the FM (using the initial guess atmosphere).	dimless	logical

Key-words	related quantity	dim.	data type
[deps]	Convergence criterion for the C.G. integrals.	dimless	real
[lainvbexp]	Switch (T/F) to define the export of the $A^{-1}B$ matrix: (A and B being the two matrices used in the retrieval according to the definition of the ATBD document): T → Matrix $A^{-1}B$ is exported; F→ Matrix $A^{-1}B$ is not exported. It is active only if *ldebug* is true (debug modality).	dimless	logical
[ljac]	Switch (F/T) to select the exporting of the Jacobian matrix on a file: T → Jacobian matrix is exported; F→ Jacobian matrix is not exported.	dimless	logical
[lhisto]	Switch (F/T) to select the exporting of the *histo* structure containing all the retrieval details: T → *histo* structure is exported; F→ *histo* structure is not exported.	dimless	logical

5.5 Output files

The MARC retrieval procedure generates as output some files that are detailed in the following.
The MARC code produces as output a set of files that are stored in the dedicated output directory (as default this directory is named OUT_FILES). Each output file is characterized by a string that gives information about the files used as input in the run of the code by which the output file has been produced(see 5.5.1). The set of output files change, according to the settings.
The whole set of output files can be arranged in some classes:

- The retrieved quantities files;
- The VCM files;
- The files with the data used to run the retrieval procedure;
- The results of the performed retrieval procedure.

In the following the aforementioned output files are introduced; a detailed description is presented in RD 3.

Standard output data when retrieval is performed without extra features

If the code is used as retrieval code without any extra features, the produced files are these described in Table 5-5:

The retrieved profiles are saved in a set of files, one for each retrieved species, named *[target].dat*, being *target* the chemical symbol of the related molecule (e.g. *h2o.dat* will contain the retrieved profile of the water). The *spectrum.dat* file contains the simulated measured spectrum (i.e. the foreseen of the spectrum that will be measured) related to the aforementioned retrieved profiles; this file also reports the simulated spectrum computed using the adopted initial guess atmosphere and the original measured spectrum.
The *iterationdetails.dat* file reports information about the details of the iterations of the retrieval procedure. The effectiveness of the single steps of the iteration is measured by the χ^2 value. Details about the observed χ^2 on each band and each geometry during the retrieval procedure are reported in the *chisquare.dat* file.
The retrieval procedure performs a check of some convergence criteria at the end of any iteration of the retrieval procedure; the *convergence.dat* file contains the values of the quantities that have been evaluated to verify the convergence criteria at the last iteration (both the values calculated over the retrieved data and the related thresholds used to evaluate the quality of the retrieval). Information about the Averaging Kernel Matrix and the related data are reported in the *AK_matrix.dat* file. Finally, the *total_correl.dat* file contains information about the total correlation existing between retrieved parameters and related data.

file	Data contained
AK_matrix.dat	information about the Averaging Kernel Matrix and the related data. For more details see RD 3.
chisquare.dat	information about the chi square on each band and each geometry during the retrieval procedure. For more details see RD 3.
convergence.dat	values that have been considered to verify the convergence criteria at each iteration. For more details see RD 3.
[target].dat	set of files (one for each retrieved target) with the retrieved information of the considered targets. For more details see RD 3.
iterationdetails.dat	information about the execution of the iterations of the iterative retrieval procedure. For more details see RD 3.
spectrum.dat	simulated/measured spectrum obtained from the retrieved atmosphere, measured spectrum obtained from the initial guess atmosphere and original measured spectrum. For more details see RD 3.
total_correl.dat	information about the total correlation between retrieved parameters and related data. For more details see RD 3.

Table 5-5. Standard output files of the MARC code when used as a retrieval code without any extra features.

The MARC code, when used in retrieval modality, generates a series of output files related to the targets chosen by the user.
The files the code is able to produce are reported in the following table, together with the related flags that have to be set to produce the aforementioned files.

file	contained data	flag to be set to produce the file
temperature.dat	retrieved profile of temperature and related data. For more details see RD 3.	`ltempfit`
[specie].dat	retrieved profile of gas specified by the string *[specie]* and related data. For more details see RD 3.	`lvmrfit`
[specie]_grad.dat	retrieved horizontal gradient of gas specified by the string *[specie]* and related data. For more details see RD 3.	`lgvmrfit`
pointing.dat	retrieved pointing value and related data. For more details see RD 3.	`lpoifit`
cont_band_[$].dat	Retrieved profile of un-accounted continuum in the band specified by the key-word *[$]* and related data. For more details see RD 3.	`lexcfit`
offset.dat	retrieved instrumental offset and related data. For more details see RD 3.	`lofsfit`
gain.dat	retrieved instrumental gain and related data. For more details see RD 3.	`lgainfit`

Table 5-6. Optional output files of the MARC code when used in Retrieval Modality.

Output data when sequential fit is performed

If the `lseq` flag is set *.T.*, the MARC code performs the sequential fit; in order to perform the retrieval in a sequential modality, some data have to be passed from the previously retrieval to the following one. To pass these data, for each retrieved quantities a set of files is produced. These files are output files of retrieval that begin input files of the following retrieval; they are introduced in the following and separately described in RD 3.

file	contained data
temperature_VCM.dat	VCM matrix of temperature and related data. For more details see RD 3.
[species]_VCM.dat	VCM matrix of species specified by the flag *[species]* and related data. For more details see RD 3.
[species]_grad_VCM.dat	VCM matrix of retrieved horizontal gradient of quantity specified by the *[species]* string and related data. For more details see RD 3.
cont_band_[$]_VCM.dat	VCM Matrix of continuum of band specified by the flag *[$]* and related data. For more details see RD 3.
vmr_[species].dat	Initial guess for the atmospheric species with their gradients; a dedicated file specified by the string *[species]* is provided. The errors of the VMR and gradient profiles are also reported. For more details see RD 3.
in_zpt.dat	Initial guess for the atmospheric profiles of temperature and pressure with the related gradients; the error of the temperature profile is also reported. For more details see RD 3.
cont_band_[$].dat	Profile (with the altitude) of the continuum value for the band specified by the string *[$]*. For more details see RD 3.

Table 5-7. Optional set of output files of the MARC code (one set for each retrieved quantity) when used in the Retrieval Modality and a sequential fit is adopted.

Notice that the *in_zpt.dat, vmr_[species].dat* and *cont_band_[$].dat* are replied with the same format of the input files with the same names used for the retrieval procedure, and do not complain the key-word syntax.
The other files are produced in order to complain with the key-word syntax.

Output data when self-standing Forward Model is performed

If the MARC code is used in forward modality, i.e. the flag `lmfm` is set ON, the MARC code produces in general only the *observ.dat* file as a standard output.
Other files are produced in the Forward Model modality if a further flag is set *on*:

if *lref* = TRUE — the *pt_ref.dat*, *[species]_ref.dat* and *cont_band_[$]_ref.dat* files are produced together with the observ.dat file;

if *lref* = FALSE — only the *observ.dat* file is produced.

Output files of the MARC code when used as Forward Model		
file	**data contained**	**flag**
observ.dat	Spectral measurement and some related parameters in the internal format of MARC code. For more details see RD 3.	*lmfm*
pt_ref.dat	For more details see RD 3.	*lmfm* and *lref*
[species]_ref.dat	For more details see RD 3.	*lmfm* and *lref*
cont_band_[$]_ref.dat	For more details see RD 3.	*lmfm* and *lref*

Table 5-8. Output files of the MARC code when used in the Forward Model modality.

5.5.1 The release string

All the output files are characterised by a data row that is preceded by the key-word *[input_key]*; e.g.:

```
[input_key]
|Simulation|001|001|001|001|001|
```

This data row reports the versions of the input files used in the run of the code that have originated the output data contained in the output file, according to the following list:

```
[input_key]
|k0|n1| n2| n3| n4| n5|
```

being:

k0	string with the name of the L1b data file
n1	number with the version of the *instrument.dat* file.
n2	number with the version of the *fov.dat* file.
n3	number with the version of the *ils.dat* file.
n4	number with the version of the *rejection.dat* file.
n5	number with the version of the *settings.dat.* file.

Table 5-9. Columns key-word for *[input_key]* data.

In case of use of simulated measurement this `k0` name is replaced with the string "`simulation`".

6 OFM

This chapter aims to describe the use of OFM with its implemented features.

The OFM is composed by two independents codes, being the first SBDART code and the second the RTM code developed by MARSCHALS team.
The output generated by SBDART module are used as input of the RTM code. The execution thus is performed in a sequence of the two components.

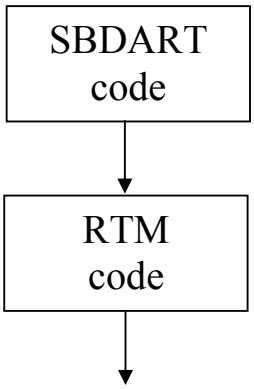

Figure 6-1. The overall scheme of the OFM with the two codes.

SBDART is a software tool that computes plane-parallel radiative transfer in clear and cloudy conditions within the Earth's atmosphere. For a general description and review of the program please refer to [RD 6]. The complete documentation and the original user manual of SBDART code is provided in the Annex 1 (Chapter 8).

The RTM code is developed by MARSCHALS team in order to use the output from SBDART and some other input data; then it computes the simulated radiance as it would received by the OCM receiver. In the Table 6-1 a list of all files needed for the execution of RTM code is presented.

The data file needed to run the RTM code	
File classification	Name of file
batch file	*run_rtm*(^); *makefile.*
source files	**.f* (source files) and **.inc* (include files) standard FORTRAN language; the whole list is reported in RD 3.
setting files	*settings_rtm.dat*
auxiliary data files	*atmosphere_rtm.dat*

(^)This file is not actually mandatory to run the code, since it is a batch file to run in an easier way the related code.

† These data are needed by the OCM code, but can be obtained by the output of the Pre-Processor; thus they are not really input data of the MARSCHALS software.

Table 6-1. The data file needed to run the RTM code.

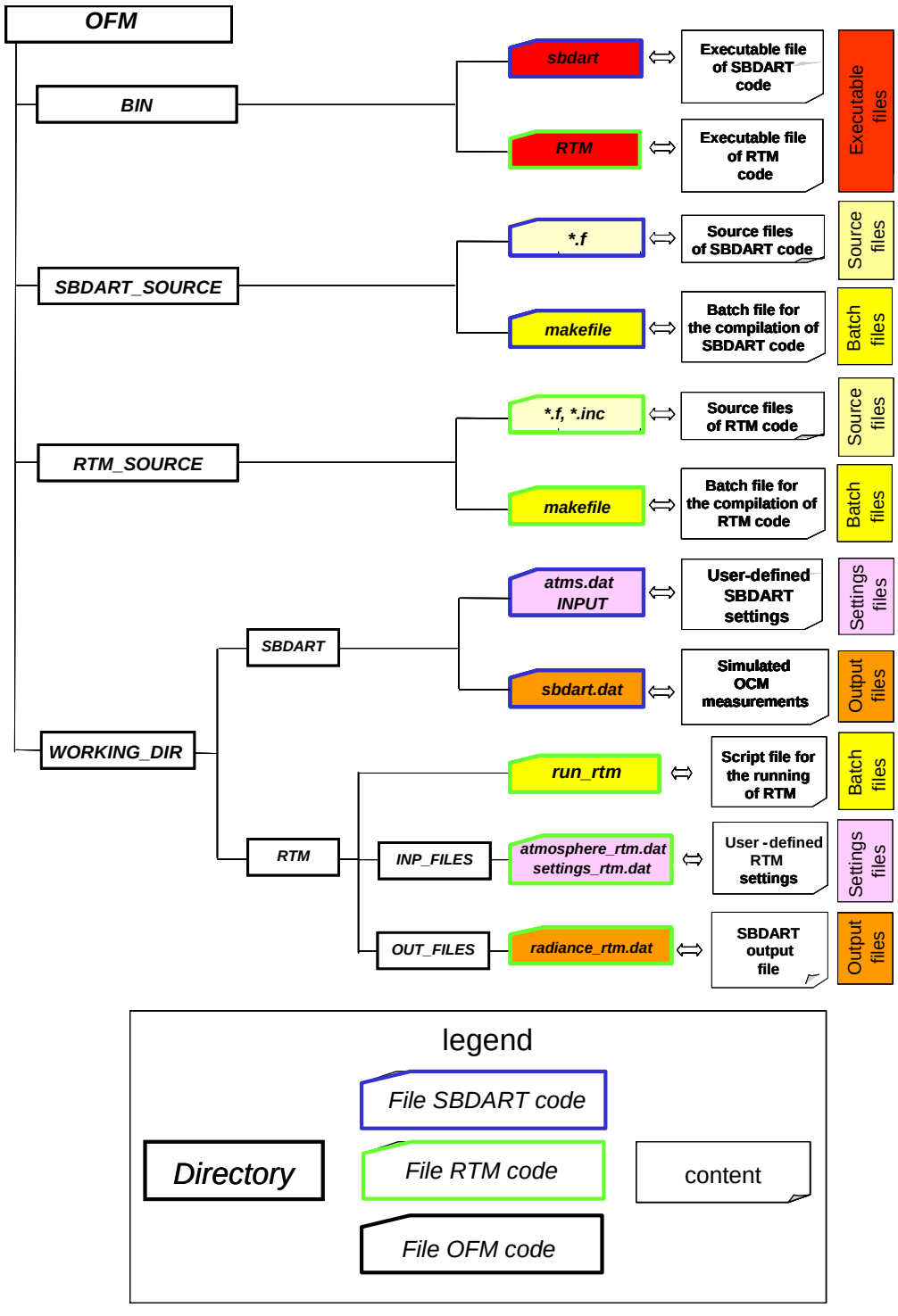

Figure 6-2. The default arrangement of the OFM codes data and source files. The names of the subdirectories are those of the default arrangement.

6.1 How to compile the source files of the SBDART and RTM code

The SBDART and RTM code is composed by some source files written in standard Fortran language

> **!** Please notices that the executable code installed on the computer delivered to ESA has been already compiled by the MARSCHALS L2 team; if no change has been made to the source codes, to run the executable version a compilation is not required

The source files of the SBDART and RTM code can be compiled in the usual way by means of a whatever FORTRAN compiler. The computer delivered to ESA is provided with a FORTRAN compiler (from Portland Company) that is able to perform this compilation task.
Once the compilation is completed, the executable code has to be moved in the adequate directory, that is, in the proposed standard configuration, the *BIN* one.

> The compilation procedure can be performed by using an adequate *makefile* file provided by the MARSCHALS L2 team together with the code files; this *makefile* file, residing in the *SBDART_SOURCE* and *RTM_SOURCE* subdirectory, can be executed by means of the usual `make` command.
> Once the compilation is performed, , the `make install` command can be used to move the executable version.

To use the *makefile* file, please move to the directory in which the source files and the *makefile* file are stored (the *SBDART_SOURCE* and *RTM_SOURCE* directory in the standard configuration) and at the prompt type:

```
make
```

and then press the "**enter**" button.

To use the `make install` command, after the compilation of the source files please move to the directory in which the compiled files are stored (the *BIN* directory in the standard configuration) and at the prompt type:

```
make install
```

and then press the "**enter**" button.

> **Please notice that the `make` and the `make install` commands works according to the standard arrangement of the file; the user can change this arrangement, but in this case, to use the make and the make install command, the makefile file have to be edited accordingly to the user defined arrangement of the files.**

6.2 How to run the OFM

OFM is not a single self-standing code. To simulate the OCM measurement, two independent codes have to be sequentially used. First of all, SBDART code have to be run to simulates the source function and the extinction coefficient profile in a plane-parallel atmosphere (see 6.2.1). Then the data produced by the SBDART code are used as input of the RTM code, that have to be run (see 6.2.2) on these data to simulate the OCM measurement in a spherical geometry.

6.2.1 How to run SBDART code

The settings of the SBDART code and the procedure to run it are described in Annex 1 (Chapter 0). Please notice that the SBDART code is a general tools, and in the MARSCHALS case it have to be set (by editing the *INPUT* file) in order to produced the desired source functions and the extinction atmospheric profiles.

The following suitable default INPUT file is provided with the MARSCHALS codes.

```
&INPUT
  idatm   =  0           ,
  wlinf   =  0.855       ,
  wlsup   =  0.855       ,
  sza     =  39.6483     ,
  isalb   =  0           ,
  albcon  =  0.0         ,
  zcloud  =  10.0, -12.0 ,
  tcloud  =  0.0, 1.0     ,
  nre     =  -10.0, -10.0 ,
  iout    =  22          ,
  nstr    =  40          ,
  uzen    =  90.         ,
  nphi    =  10          ,
  phi     =  0.0, 180.0   ,
  imomc   =  3           ,
  idb(8)  =  1           ,
  /
```

The SBDART code can be run in this way; at the prompt of the SBDART working directory (OFM/WORKING_DIR/SBDART in the default configuration) type:

```
../../BIN/sbdart > sbdart.dat
```

and then press the "**enter**" button.

The outputs are written, in the default configuration, in the **`sbdart.dat`** output file.

6.2.2 How to run the RTM code

The user can run RTM code by using the batch file *run_rtm*.
At the prompt of the RTM working directory (OFM/WORKING_DIR/RTM in the default configuration) type:

```
run_rtm
```

and then press the "**enter**" button.
The user can change the default arrangements of the subdirectory by editing the batch file *run_rtm* and inserting the personalized organization.

6.3 *Input files*

6.3.1 The input of the RTM code

RTM code reads as input the file *atmosphere_rtm.dat* and the file *settings_rtm.dat*.
The file *atmosphere_rtm.dat* contains data related to the vertical distribution of the pressure, temperature and water vapour; more on it contains the data elaborated by SBDART, such as the linear extinction coefficient and the radiance at a given altitude.

6.4 Settings file of RTM code

The user can set the procedure of the OFM by editing the setting file of RTM (called *atmosphere_rtm.dat*).

6.4.1 How to define the altitude of the receiver

The user can introduce a value to fix the altitude of the receiver.

[`receiver_altitude`] == real value (Km) to identify the altitude of the OCM receiver

6.4.2 How to define the pointing angles

The user can introduce one or more pointing angles, at which the simulation is performed.

[n_angles] == an integer to define the number of pointing angles;

[pointing_angles] == an array (real values) used to define the zenithalpointing angles (degrees).

6.4.3 How to define the receiver field of view

The user can define an uniform field of view by entering an angular value.

[`fov`] == a real value which define the half-aperture of the antenna

6.4.4 How to define the latitude of the observations

The user can define the latitude at which the simulation is carried out. This information is used by RTM code to compute the local earth radius.

[`latitude`] == a real value used to define the latitude (degrees)

6.4.5 How to insert an identifier of the computation

The user can introduce a character string to identify the current setting file. The string is reported in the output file.

[`setting_key`] == a string (3 characters) used to identify the file.

6.5 *Output files*

6.5.1 Output of RTM code

The RT code returns as output the file *radiance_rtm.dat*
This file contains the radiance simulated by RTM code for each pointing angle of the instrument.
Data are arranged in rows and columns, with a row for each pointing angle of the RTM code.
In each row the data are arranged in columns labelled by the following key-words.

Key-words	**related quantity**	**dim.**	**data type**
Ind	Sequential index of the element.	-	int
Ang	Pointing angle.	deg	real
Rztang	Value of tangent altitude [referred to pointing angle (*ang*)].	Km	real
Rheta	Angular coordinate of the tangent point [referred to pointing angle (*ang*)].	deg	real
radiance	Value of radiance simulated for the pointing angle (*ang*).	W/(m^2*sr*nm)	real

Table 6-2. Parameters defined in the *radiance_rtm.dat* file, listed with their key-words.

6.5.2 Log file

During the process, RTM elaborates a log-file, in which messages about the processing are reported. The content of the messages are self-explicative, and do not require further comments.

7 The visualization tool

The MARSCHALS suite of codes is provided with a visualization tool that can be used to visualize in a easier way the results of the retrieval perform by the MARSCHALS code.
The visualization tool is constituted by an IDL code. It does not require any installation, but an existing IDL license for the computer in which the MARSCHALS code is installed is needed to run the tool. Please notice that this license is not provided by the MARSCHALS team for the delivered computer, and must be arranged by the user.
In the default configuration, the files of the visualization tools are placed in the subdirectory IDL of the main directory MARSCHALS_L2, as depicted in Figure 7-1.

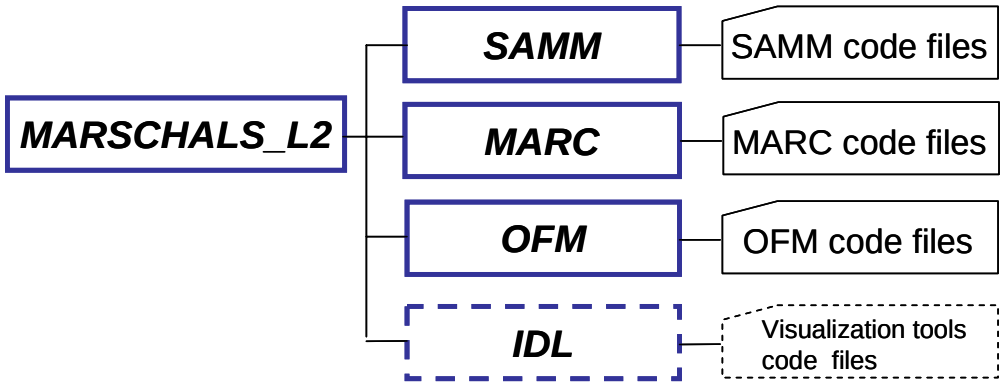

Figure 7-1. The overall default arrangement in three main subdirectories of the files composing the MARSCHALS L2 suite (in blue the subdirectories).

7.1 How to run the visualization tool

The visualization tools must be run by directly using the executable file *run* (stored as a default in the *MARSCHALS_L2/IDL* subdirectory) in a IDL environment.

To run the visualization tool code please follow the above steps:

1. move to the subdirectory in which the *run* IDL executable code is placed (the *MARSCHALS_L2/IDL* in the default arrangement);

2. launch the IDL environment by typing at the system prompt:

   ```
   idl
   ```

   and then press the "**enter**" button.

3. Inside the IDL platform, launch the visualization tool, by typing at the IDL prompt:

   ```
   @run
   ```

   and then press the "**enter**" button.
   A simple graphic user interface will be shown

4. In the graphic interface, by using the toolbar, choose the directory in which the MARSCHALS output data to be visualized are placed.

The visualization tool will produce a series of pictures (one for each retrieved target) with information about the related retrieved quantity. These picture are stored in files named *[target]_col.TIFF*, placed in the same directory of the considered MARSCHALS data.
The output of the visualization tool is described in a more detailed way the following paragraph.

7.2 The visualization tool output

The visualization tool is able to display information about all the MARC retrieved quantities. For each retrieved target the visualization tool automatically produces a picture containing the retrieved profile, its Expected Standard Deviation and the related quality parameters and averaging kernel.
An example of such a picture is depicted in Figure 7-2.
The aforementioned pictures are saved as a files named *[target]_col.TIFF* in the same directory of the considered MARC output data. These pictures are also directly shown in the default output (i.e. generally, the monitor of the computer).

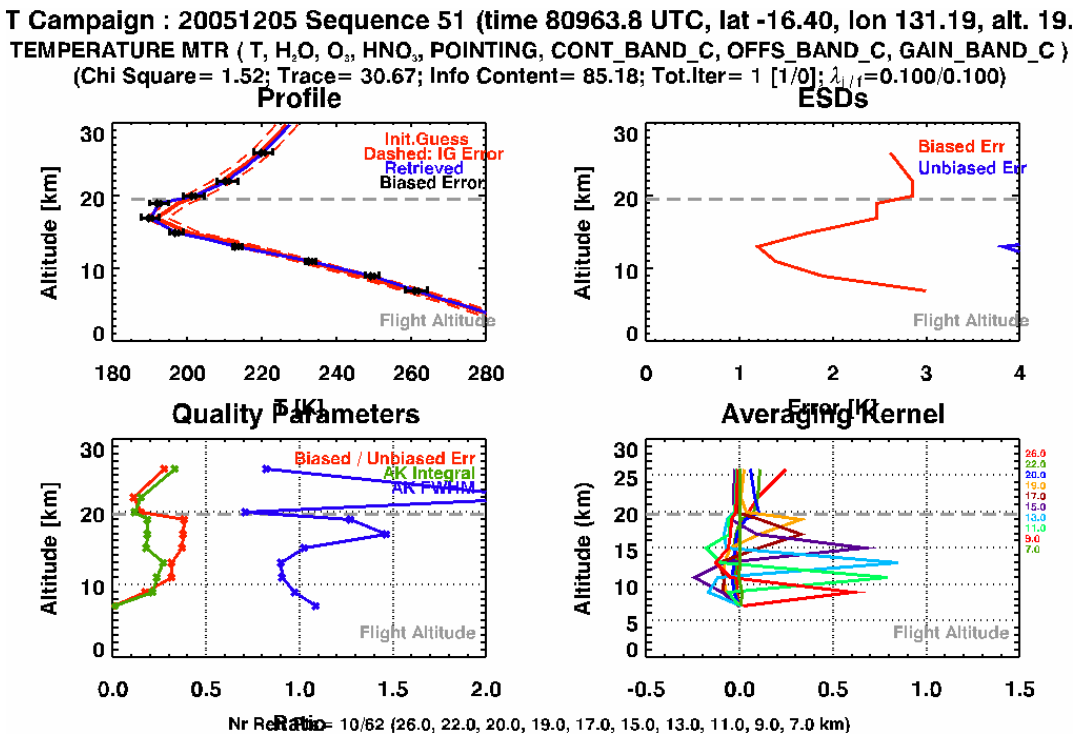

Figure 7-2. The output of the visualization tool.

This page is intentionally left blank.

8 ANNEX 1: SBDART documentation

Input documentation for SBDART (summer 2002 release)

Paul Ricchiazzi
Institute for Computational Earth System Science
University of California, Santa Barbara

This file documents input parameters for SBDART, (Santa Barbara DISORT Atmospheric Radiative Transfer). SBDART is a software tool that computes plane-parallel radiative transfer in clear and cloudy conditions within the Earth's atmosphere and at the surface. For a general description and review of the program please refer to Ricchiazzi et al 1998. (Bulletin of the American Meteorological Society, October 1998).

SBDART's main input file is called INPUT. This file contains a single NAMELIST input block also named INPUT. A significant advantage of NAMELIST input is that not all elements of an input block need be specified by the user. Since most of the code inputs have been initialized with reasonable default values, new users can start by specifying just a few interesting input parameters. The default state of input parameters may be determined by removing INPUT from the current working directory. When SBDART detects the absense of file INPUT, it will print the default settings of all input parameters. This output may be redirected to a file for editing.

The default configuration of INPUT is as follows:

```
==========================================================================

 &INPUT
  idatm   =  4         ,  amix    =  0.0       ,  isat    =  0         ,
  wlinf   =  0.550     ,  wlsup   =  0.550     ,  wlinc   =  0.0       ,
  sza     =  0.0       ,  csza    =  -1.0      ,  solfac  =  1.0       ,
  nf      =  2         ,  iday    =  0         ,  time    =  16.0      ,
  alat    =  -64.7670  ,  alon    =  -64.0670  ,  zpres   =  -1.0      ,
  pbar    =  -1.0      ,  sclh2o  =  -1.0      ,  uw      =  -1.0      ,
  uo3     =  -1.0      ,  o3trp   =  -1.0      ,  ztrp    =  0.0       ,
  xrsc    =  1.0       ,  xn2     =  -1.0      ,  xo2     =  -1.0      ,
  xco2    =  -1.0      ,  xch4    =  -1.0      ,  xn2o    =  -1.0      ,
  xco     =  -1.0      ,  xno2    =  -1.0      ,  xso2    =  -1.0      ,
  xnh3    =  -1.0      ,  xno     =  -1.0      ,  xhno3   =  -1.0      ,
  xo4     =  1.0       ,  isalb   =  0         ,  albcon  =  0.0       ,
  sc      =  1.0,3*0.0 ,  zcloud  =  5*0.0     ,  tcloud  =  5*0.0     ,
  lwp     =  5*0.0     ,  nre     =  5*8.0     ,  rhcld   =  -1.0      ,
  krhclr  =  0         ,  jaer    =  5*0       ,  zaer    =  5*0.0     ,
  taerst  =  5*0.0     ,  iaer    =  0         ,  vis     =  23.0      ,
  rhaer   =  -1.0      ,  wlbaer  =  47*0.0    ,  tbaer   =  47*0.0    ,
  abaer   =  -1.0      ,  wbaer   =  47*0.950  ,  gbaer   =  47*0.70   ,
  pmaer   =  940*0.0   ,  zbaer   =  50*-1.0   ,  dbaer   =  50*-1.0   ,
  nothrm  =  -1        ,  nosct   =  0         ,  kdist   =  3         ,
  zgrid1  =  0.0       ,  zgrid2  =  30.0      ,  ngrid   =  50        ,
  zout    =  0.0,100.0 ,  iout    =  10        ,  deltam  =  t         ,
  lamber  =  t         ,  ibcnd   =  0         ,  phi0    =  0.0       ,
  prnt    =  7*f       ,  ipth    =  1         ,  fisot   =  0.0       ,
  temis   =  0.0       ,  nstr    =  4         ,  nzen    =  0         ,
  uzen    =  20*-1.0   ,  vzen    = 20*90      ,  nphi    =  0         ,
  phi     =  20*-1.0   ,  imomc   =  3         ,  imoma   =  3         ,
  ttemp   =  -1.0      ,  btemp   =  -1.0      ,  spowder =  f         ,
  idb     =  20*0
 /

==========================================================================
```

NOTE: Unfortunately, many fortran compilers produce rather cryptic error messages in response to improper NAMELIST input files. Here are three common NAMELIST error messages and their meaning:

1. ERROR MESSAGE: invalid reference to variable in NAMELIST input

   MEANING: you misspelled one of the NAMELIST variable names

```
      2. ERROR MESSAGE: too many values for NAMELIST variable

         MEANING:       you specified too many values for a variable,
                        most likely because you separated variables by
                        more than one comma.

      2. ERROR MESSAGE: end-of-file during read
                          or
                        namelist block INPUT not found

         MEANING:       There are two possibilities:
                        A) you didn't include a NAMELIST block specifier
                           (INPUT, DINPUT or END) or you misspelled it; or
                        B) You used the wrong character to signify a
                           namelist block name.  FORTRAN90 expects the
                           namelist blocks to start with & and end
                           with /, but most FORTRAN77 compilers used the
                           $, $END convention.


Other input files are sometimes required by SBDART:

atms.dat     -- atmospheric profile          (see input quantity IDATM)
aerosol.dat  -- aerosol information          (see input quantity IAER)
albedo.dat   -- spectral surface albedo      (see input quantity ISALB)
filter.dat   -- sensor filter function       (see input quantity ISAT)
solar.dat    -- solar spectrum               (see input quantity NF)
usrcld.dat   -- cloud vertical profile       (see input quantity TCLOUD)

SBDART usually lists computational results directly to the standard output
device (i.e., the terminal, if run interactively).  However, some
warnings issued by the DISORT radiative transfer module are written to
files named DISORT_WARNING.?? where the question marks indicate a warning
message number.  These files contain a warning message and a copy of the
input file that triggered the warning.


                    General options (NAMELIST &INPUT):
                    ==================================


               WAVELENGTH LIMITS, FILTER FUNCTION SPECIFICATION
               ================================================


  NF:           SOLAR SPECTRUM SELECTOR

               -2 = use TOA solar irradiance read from CKTAU file when
                    kdist=-1.  NF=-2 is not a valid input when kdist.ne.-1

               -1 = read from file solar.dat (user supplied)
                    data file, "solar.dat" is read from the current
                    working directory. This ASCII file is read with the
                    following free format read statements:

                         read(13,*,end=100) (wlsun(i),sun(i),i=1,5000)
                    100  continue

                    where, wlsun   wavelength sample points (microns)

                           sun     direct normal solar irradiance at the
                                   top of the atmosphere (W/m2/micron)

                    The number of wavelength sample points read from
                    solar.dat should be less than or equal to 5000

                    This file format is new for the SBDART_2002 version.
                    Previous versions of SBDART used a different
                    format for spectral input files albedo.dat,
                    filter.dat and solar.dat.  A perl script 'newform'
                    is available from
                    ftp::/ftp.icess.ucsb.edu/pub/esrg/sbdart to
                    convert old data files to the new format.
```

```
                0 = spectrally uniform

                1 = 5s solar spectrum
                    0.005 micron resolution,  .25 to 4 micron

                2 = LOWTRAN_7 solar spectrum  (default)
                    20 cm-1 resolution,       0. to 28780 cm-1
                    10 cm-1 resolution,   28780. to 57490 cm-1

                3 = MODTRAN_3 solar spectrum
                    20 cm-1 resolution,       100 - 49960 cm-1


ISAT:           FILTER FUNCTION TYPES

               -4  Guassian filter, WLINF-2*WLSUP to WLINF+2*WLSUP
               -3  Trianglar filter, WLINF-WLSUP to WLINF+WLSUP
               -2  Flat filter, WLINF-.5*WLSUP to WLINF+.5*WLSUP
               -1  USER DEFINED, read from filter.dat

                0  WLINF TO WLSUP WITH FILTER FUNCTION = 1      (default)

                   NOTE: if ISAT=0 and KDIST=-1, then the values of
                         WLINF and WLSUP only have an effect if they
                         have been changed from their default values
                         such that WLINF .ne. WLSUP.  Otherwise the
                         wavelength sample points are as specified in
                         the CKTAU file.


                1  METEO
                2  GOES(EAST)
                3  GOES(WEST)
                4  AVHRR1(NOAA8)
                5  AVHRR2(NOAA8)
                6  AVHRR1(NOAA9)
                7  AVHRR2(NOAA9)
                8  AVHRR1(NOAA10)
                9  AVHRR2(NOAA10)
               10  AVHRR1(NOAA11)
               11  AVHRR2(NOAA11)
               12  GTR-100 ch1
               13  GTR-100 ch2
               14  GTR-100 410nm channel
               15  GTR-100 936nm channel
               16  MFRSR 415nm channel
               17  MFRSR 500nm channel
               18  MFRSR 610nm channel
               19  MFRSR 665nm channel
               20  MFRSR 862nm channel
               21  MFRSR 940nm channel
               22  AVHRR3 (nominal)
               23  AVHRR4 (nominal)
               24  AVHRR5 (nominal)
               25  Biological action spectra for DNA damage by UVB radiation
               26  AIRS1 380-460nm
               27  AIRS2 520-700nm
               28  AIRS3 670-975nm
               29  AIRS4 415-1110nm


                   NOTE: If ISAT=-1 a user supplied filter data file
                         "filter.dat" is read from the current working
                         directory. This ASCII file is read with the
                         following free format read (numbers may be
                         separated by spaces, commas or carriage returns);

                         read(13,*,end=100) (wlfilt(i),filt(i),i=1,huge(0))
                           100 continue

                         where, wlfilt   wavelength sample points (microns)

                                filt     filter response value (unitless)

                         The number of wavelength sample points read from
                         filter.dat should be less than or equal to 1000
```

```
                    This file format is new. Previous versions of
                    SBDART used a different format for spectral
                    input files albedo.dat, filter.dat and
                    solar.dat.  A perl script 'newform' is
                    available from
                    ftp::/ftp.icess.ucsb.edu/pub/esrg/sbdart to
                    convert old data files to the new format.


WLINF:      lower wavelength limit when ISAT=0    (WLINF > .250 microns)
            central wavelength when ISAT=-2,-3,-4

WLSUP:      upper wavelength limit when ISAT=0    (WLSUP < 100.0  microns)
            equivalent width when ISAT=-2,-3,-4

            NOTE:

            If ISAT eq -2, a rectangular filter (constant with
            wavelength) is used, with central wavelength at WLINF
            and an equivalent width of WLSUP (full width = WLSUP)

            If ISAT eq -3, a triangular filter function is used with
            the central wavelength at WLINF and an equivalent width of
            WLSUP (full width = 2*WLSUP) (filter function is zero at end
            points, and one at WLINF).

            If ISAT eq -4, a gaussian filter function is used with
            the central wavelength at WLINF and an equivalent width of
            WLSUP (full width = 4*WLSUP)

            If output is desired at a single wavelength, set
            WLINF=WLSUP and ISAT=0.  In this case, SBDART will
            set WLINC=1 (the user specified value of WLINC is ignored)
            and the output will be in units of (W/m2/um) for
            irradiance and (W/m2/um/sr) for radiance.


WLINC:      This parameter specifies the spectral resolution of the
            SBDART run.  Though the spectral limits of the calculation
            are always input in terms of wavelength, the spectral
            step size can be specified in terms of constant
            increments of wavelength, log(wavelength) [same as
            constant increment of log(wavenumber)] or wavenumber.
            Which one to choose depends on where in the spectral
            bandpass you want to place the most resolution.

            Since SBDART is based on LOWTRAN7 band models, which
            have a spectral resolution of 20 cm-1, it would be extreme
            overkill to allow spectral step size less than 1 cm-1. On
            the other hand a spectral resolution coarser than 1 um
            is also pretty useless. Therefore the way WLINC is
            interpreted depends on whether it is less than zero,
            between zero and one, or greater than 1.

          * WLINC = 0 (the default) =>  wavelength increment is
            equal to 0.005 um or 1/10 the wavelength range, which
            ever is smaller.  If the WLINF=WLSUP then WLINC=.001

          * WLINC < 0 => wavelength increment is a constant fraction
            of the current wavelength.  WLINC is interpreted as
            a specified value of delta(lambda)/lambda and the
            wavelength steps are adjusted so that wavelength step is
            approximately the product of the current wavelength and
            WLINC.

            Specifying the wavelength increment as a fractional
            step size is useful when the wavelength range extends
            over more than an decade of wavelength.  For example,
            if the wavelength range is 0.5 to 20.0, specifying a
            constant wavelength increment of .01 microns tends to
            under-resolve the low wavelengths and over-resolve the
            long wavelengths.  Setting WLINC = -.01 causes the
            code to use a wavelength increment of about .005
            microns in the visible and about .2 micron in the
            thermal infrared, which is a better compromise of
            resolution and computer time.
```

```
             * 1 >= WLINC > 0    => WLINC is the wavelength step size (um)

               if WLINC > 1 then WLINC is the step size in inverse
               centimeters.  If maximum fidelity is required and
               gaseous absorption is the primary influence on the
               output, then WLINC should be set to 20, which is the
               wavenumber resolution of the LOWTRAN7 band models.

               The total number of wavelength steps, nwl, is given by

                nwl = 1+ln(wlsup/wlinf)/|wlinc|          wlinc < 0

                nwl = 1+(wlsup-wlinf)/wlinc             1 >= wlinc > 0

                nwl = 1+10000*(1/wlinf-1/wlsup)/wlinc    wlinc > 1


               SOLAR GEOMETRY
               ==============

SZA:           solar zenith angle (degrees) (default = 0.)
               Solar input may be turned off by setting sza>90
               SZA is ignored if CSZA is non-negative or IDAY is non-zero.

CSZA:          Cosine of solar zenith angle.  If CSZA > 0, solar
               zenith angle is set to acos(CSZA) (default = -1.)

IDAY:          If IDAY > 0, the solar illumination angles (SZA, PHI0)
               are computed from the specified time and geographic
               coordinates using an internal solar ephemeris
               algorithm (see subroutine zensun). IDAY is the number
               of days into a standard "year", with IDAY=1 and the
               1st of January and IDAY=365 on the 31st of December
               if IDAY > 365, IDAY is replaced internally by
               mod(IDAY-1,365)+1.

               If IDAY < 0, the code writes the values of abs(iday),time,
               alat,alon,sza,azm, and solfac to standard output and exits.

TIME:          UTC time (Grenwich) in decimal hours
ALAT:          latitude of point on earth's surface
ALON:          east longitude of point on earth's surface

               NOTE: TIME, ALAT and ALON are ignored if IDAY .eq. 0

SOLFAC:        solar distance factor.  Use this factor to account
               for seasonal variations of the earth-sun distance.
               If R is the earth-sun distance in Astronomical Units
               then SOLFAC=1./R**2.  SOLFAC is set internally when the
               solar geometry is set through IDAY, TIME, ALAT and ALON.
               In this case SOLFAC is set to,
                                                        -2
               SOLFAC = ( 1.-eps*cos(2*pi*(IDAY-perh)/365) )

               where eps  = orbital eccentricity = 0.01673
               and   perh = day of perihelion    = 2           (jan 2)

               NOTE: seasonal variations in earth-sun distance
                     produce a +/-3.4% perturbation in the TOA solar
                     flux.  This factor should be included when
                     making detailed comparisons to surface
                     measurements.

NOSCT:         aerosol scattering mode used for boundary layer aerosols:
               0    normal scattering and absorption treatment
               1    reduce optical depth by (1-ssa*asym), set ssa=0
               2    set ssa=0
               3    reduce optical depth by (1-ssa), set ssa=0

               where ssa=single scattering albedo, and asym=asymmetry factor

               nosct does not affect the stratospheric aerosol models or
               the iaer=6 boundary layer model.
```

```
                SURFACE REFLECTANCE PROPERTIES
                ==============================

ISALB:          SURFACE ALBEDO FEATURE

               -1 -spectral surface albedo read from "albedo.dat"
                0 -user specified, spectrally uniform albedo set with ALBCON
                1 -snow
                2 -clear water
                3 -lake water
                4 -sea  water
                5 -sand
                6 -vegetation
                7 -sunglint off of ocean, wind speed set with ALBCON
               10 -combination of snow, seawater, sand and vegetation

               NOTE: If ISALB=-1 a user supplied spectral reflectance
                     file "albedo.dat" is read from the current working
                     directory. This ASCII file is read with the
                     following free format read (numbers may be
                     separated by spaces, commas or carriage returns);

                        read(13,*,end=100) (wlalb(i),alb(i),i=1,huge(0))
                    100 continue

                        where, wlalb   wavelength sample points (microns)
                               alb     spectral albedo (unitless)

                        The number of wavelength sample points read from
                        albedo.dat should be less than or equal to 1000


               The user specified reflectance may cover any
               wavelength range and have arbitrarily high resolution.
               This contrasts with the standard reflectance models
               (sand, vegetation, lake water and sea water) which are
               only specified in in the range .25 to 4 um at 5nm
               resolution.

               This file format is new. Previous versions of SBDART used
               a different format for spectral input files albedo.dat,
               filter.dat and solar.dat.  A perl script 'newform' is
               available from ftp::/ftp.icess.ucsb.edu/pub/esrg/sbdart
               to convert old data files to the new format.

ALBCON:        ISALB=0 --  a spectrally uniform, surface albedo
               ISALB=7 --  the wind speed (m/s)

SC:            Composite albedo fractions (applies only when ISALB=10)
               SC(1) = fraction of snow
               SC(2) = fraction of ocean
               SC(3) = fraction of sand
               SC(4) = fraction of vegetation

               NOTE:   SC(1)+SC(2)+SC(3)+SC(4) need not sum to 1.
                       Thus, it is possible to use the SC factor to
                       boost the overall reflectance of a given
                       surface type.  For example, SC=0,0,2,0 yields
                       results for a surface with spectral reflectivity
                       twice that of sand.  Beware, total reflectance
                       greater than one will produce unphysical results.


               MODEL ATMOSPHERES
               =================
                                          default              ozone(atm-cm)
IDATM:         ATMOSPHERIC PROFILE:     water vapor (g/cm2)  total  below_10km
               ------------------       -------------------  -------------
               0 User Specified
               1 TROPICAL                      4.117         0.253   .0216
               2 MID-LATITUDE SUMMER           2.924         0.324   .0325
               3 MID-LATITUDE WINTER           0.854         0.403   .0336
               4 SUB-ARCTIC SUMMER             2.085         0.350   .0346
               5 SUB-ARCTIC WINTER             0.418         0.486   .0340
               6 US62                          1.418         0.349   .0252
              -n List to standard out
```

```
                If IDATM = 0, a user supplied atmospheric profile,
                "atms.dat", is read from the current working
                directory.  This ASCII file is read with the following
                free format read statements (input values may be
                separated by spaces, commas or carriage returns);

                     read(13,*) nn
                     do 10 i=1,nn
                       read(13,*) z(i),p(i),t(i),wh(i),wo(i)
                10   continue

                     where nn is the number atmospheric layers
                              nn should be less or equal to than MXLY,
                              a parameter used in SBDART (see params.f)
                              to set the maximum number of levels in the
                              vertical grids.

                           z  is the layer altitude in km
                              (z must be monotonically decreasing)
                           p  is the pressure in millibars
                           t  is the temperature is Kelvin
                           wh water vapor density g/m3
                           wo ozone density g/m3

                If IDATM is set to a negative number in the range -1
                to -6 SBDART prints the atmospheric model
                corresponding to abs(idatm) to standard out, and then
                quits.

AMIX:           weighting factor,  when positive this factor controls
                how much of the atms.dat atmospheric profile to mix in
                with one of the standard internal profiles selected by
                IDATM.  For example IDATM=1 and AMIX=.7 specifies a
                70% weighting of atms.dat and a 30% weighting of
                profile TROPIC.  No (default=-1)


UW:             integrated water vapor amount (G/CM2)

UO3:            integrated ozone concentration (ATM-CM)
                above the level ZTRP.  The default value of ZTRP=0,
                so UO3 usally specifies the total ozone column.
                (1 atm-cm = 1000 Dobson Units)

          NOTE: Use UW or UO3 to set the integrated amounts of water
                vapor or ozone in the model atmosphere.  Aside from
                multiplicative factors the vertical profile will be
                that of the original model atmosphere set by IDATM.
                The original unmodified density profile is used when
                UW or UO3 is negative.

O3TRP:          integrated ozone concentration (ATM-CM) in troposphere.
               i.e., for z.lt.ZTRP.  The original tropospheric density
                is used when O3TRP is negative. (default=-1.)

ZTRP:           The altitude of the tropopause.  The parameters UO3 and
               O3TRP sets the total column ozone in the stratosphere
               and troposphere, respectively.  Note: since the default
               value of ZTRP is zero, UO3 normally sets the integrated
                ozone amount of the entire atmosphere (default=0).


XN2:            volume mixing ratio of N2   (PPM,  default = 781000.00 )
XO2:            volume mixing ratio of O2   (PPM,  default = 209000.00 )
XCO2:           volume mixing ratio of CO2  (PPM,  default =    360.00 )
XCH4:           volume mixing ratio of CH4  (PPM,  default =      1.74 )
XN2O:           volume mixing ratio of N2O  (PPM,  default =      0.32 )
XCO:            volume mixing ratio of CO   (PPM,  default =      0.15 )
XNH3:           volume mixing ratio of NH3  (PPM,  default =      5.0e-4)
XSO2:           volume mixing ratio of SO2  (PPM,  default =      3.0e-4)
XNO:            volume mixing ratio of NO   (PPM,  default =      3.0e-4)
XHNO3:          volume mixing ratio of HNO3 (PPM,  default =      5.0e-5)
XNO2:           volume mixing ratio of NO2  (PPM,  default =      2.3e-5)

                NOTE: Setting any of these factors to -1 causes that
                      atmospheric component to retain its nominal
```

mixing ratio defined in the US62 atmosphere
(as listed above).

The volume mixing ratio (VMR) of a given species
is adjusted by specifying the surface value of
its VMR in PPM. The entire altitude profile is
multiplied by the ratio of the user specified
VMR and the nominal surface VMR.

There are no further re-normalizations of the
VMR. Thus, the total of all the VMRs may be
greater or less than 10^6. By the way, the
default set of VMRs do not add up to 10^6
because of the exclusion of the noble gases
which do not have any radiative effects.

XRSC: sensitivity factor for Rayleigh scattering (default=1)
This factor varies the strength of Rayleigh scattering
for sensitivity studies.

XO4 sensitivity factor to adjust strength of absorption by
oxygen collisional complexes (default=1, see comments
in subroutine o4cont)

PBAR: surface pressure in millibars.
If PBAR .gt. 0 then each pressure is multiplied by the
factor (PBAR/P0) where P0 is the surface pressure of the
original atmosphere. At each height in the atmosphere
the density of all molecular species (sources of absorption
and Rayleigh scattering) is proportional to pressure.
If PBAR .le. 0, the original pressure profile is used.

ZPRES: Surface altitude in kilometers.
This parameter is just an alternate way of setting the
surface pressure, and should not be set when PBAR is
specified. When ZPRES is set PBAR is obtained by
logarithmic interpolation on the current model's
atmosphere pressure and altitude arrays. Changing
ZPRES does not alter other parameters in the
atmospheric model in any way. Note that setting a
large value of ZPRES may push the tropopause (where
dT/dz=0) to an unrealistically high altitude.

SCLH2O: Water vapor scale height in km.

If SCLH2O .gt. 0, then water vapor is vertically
distributed as exp(-z/SCLH2O)

If SCLH2O .le. 0, then the original vertical profile
is used. Changing SCLH2O has no effect on the total
water vapor amount.

CLOUD PARAMETERS
================

ZCLOUD: Altitude of cloud layers (km) (up to 5 values), Cloud
layers may be specified in two ways. To specify
separate cloud layers, set ZCLOUD to a sequence of
monotonically increasing altitudes. Each value of
ZCLOUD will set the altitude (above the surface) of the
corresponding optical depth in the TCLOUD array.

To specify a range of altitudes which will be filled
by cloud, tag the second element of the range with a
minus sign. Consider,

zcloud=1,-3,10,-15
tcloud=4,0,8,0,0
nre=6,6,8,9,10

In this example two continuous cloud layers are
defined, the lower one extends from 1 to 3 km and has
a total optical depth of 4 and an effective radius of
6um. The upper cloud layer extends from 10 to 15 km,
has a total optical thickness of 8 and a sliding value
of effective radius which starts 8um at the bottom of

```
the cloud and ramps up to 9um at 15km. However, beware
that the actual location of the cloud layers is
determined by the resolution and placement of vertical
grid points in SBDART, as explained below.

SBDART puts the i'th cloud layer at the highest vertical
grid point, k, such that

   z(k) .le. abs(ZCLOUD(i)+.001)

NOTE: A cloud with a nominal altitude equal to that of
one of the computational layer altitudes, Z(K),
actually extends from Z(k) to the next higher grid
point.  For example, a cloud layer at Z(k) will not
affect the direct beam flux at Z(k-1) (one layer
above) but will strongly affect it at Z(k).  (You can
check this out your self by setting IOUT=10 and
ZCLOUD=1 and messing around with ZOUT to get outputs
just above or below the cloud).

Suppose the bottom of your computational grid looks like

 k         z(k)
 ...       ...
 30        2.5
 31        2.0
 32        1.5
 33        1.0
 34        0.5
 35        0.0

If you want a cloud to extend from 0.5 to 1.5 km, then
set ZCLOUD= .5, -1.5.  Actually the same result would
be obtained by setting the second element of ZCLOUD to
anything between -(1.0+epsilon) and -1.5.

Consider,

         TCLOUD= 6.,  0.0,  12.,  0.0, 0
         ZCLOUD=1.0, -6.0,  4.0, -9.0, 0

Here two overlapping cloud decks are specified, one
extending from 1 to 6 km with a total optical
thickness of 6, and the other from 4 to 9 km with a
total thickness of 10.  Since the total optical
thickness is spread over the total altitude range we
would have 1 optical depth per km for the lower cloud
deck and 2 optical depths per km for the second.  The
code adds the effects of both cloud decks in the
region of overlap.  So the above specification would
yield 1 optical depth per km between 1 and 4 km, 3
optical depths per km between 4 and 6 km and 2 optical
depths per km between 4 and 9 km for a total optical
depth of 18.

NOTE: When a ZCLOUD range is being specified (i.e., a
negative value is used to set the upper end of a
range), the opacity appears only between the low and
high altitudes in the range.  When individual cloud
layers are set, the opacity extends from the named
altitude to the altitude of the next higher
computational level.  Thus, if the computational
levels are at 1km intervals starting from zero, the
following inputs do exactly the same thing:

   TCLOUD=2,0
   ZCLOUD=1,-3


   TCLOUD=1,1
   ZCLOUD=1,2


If you have any doubt about where the code is putting
the cloud, set IDB(8)=1 (see below) to get a diagnostic
print out of cloud optical depth.
```

```
                NOTE: do not try to put an ice cloud (NRE < 0) in a
                      cloud layer range which includes water cloud
                      (2 le NRE le 128).  In other words this specification
                      won't work:

                        ZCLOUD = 1,-4
                        TCLOUD = 1, 0
                        NRE    = 8,-1

TCLOUD:         Optical thickness of cloud layer, (up to 5 values)

                TCLOUD specifies the cloud optical depth at a
                wavelength of 0.55um.    The cloud optical depth at
                other wavelengths is computed using the relation,

                tau = TCLOUD*Q(wl)/Q(0.55um),

                where Q is the extinction efficiency, which is a
                function of effective radius and wavelength (see
                discussion of LWP for a definition of Q).  The code
                contains a look-up table of Q that covers effective
                radii in the range 2 to 128um for water clouds and for
                a single effective radius of 106um for ice clouds.
                The wavelengths range is 0.29 to 333.33 um for water
                clouds and .29 to 20 um for ice clouds.

                When specifying an optical depth for a range of grid
                levels, the second TCLOUD entry corresponding to the
                cloud top altitude is usually set to zero.  This
                produces a uniform distribution of opacity over the
                altitude range.

                For example,

                ZCLOUD= 1 ,-5,0,0,0    # uniformly distributed opacity
                TCLOUD= 10, 0,0,0,0    # for a cloud of extent 4 km
                NRE=    10,20          # 2.5 optical depths per km
                                       # effective radius ramps from
                                       # 10 to 20 between 1 and 5km

                A linearly varying opacity distribution can be
                obtained by setting the second TCLOUD entry to a
                factor which represents the ratio of the opacity in
                the highest layer to that in the lowest layer

                For example,

                ZCLOUD= 1 ,-5,0,0,0    # linearly distributed opacity
                TCLOUD= 10, 4,0,0,0    # for a cloud of extent 4 km
                                       # between tau(1-2km)=1
                                       # between tau(2-3km)=2
                                       # between tau(3-4km)=3
                                       # between tau(4-5km)=4
                                       #
                                       # tau(total)=10
                                       # tau(4-5km)/tau(1-2km)=4

                NOTE: if r is the ratio of the top to bottom and t
                      is the average opacity per level then,

                           tau(top_level)=t*2r/(1+r)

                           tau(bot_level)=t*2/(1+r)

                NOTE: a linear increase in opacity, starting from zero
                      at the cloud bottom, is obtained by setting,

                           r=1 + 2*zdiff/dz

                      where dz is the grid spacing and zdiff is the total
                      altitude range over which the cloud extends.  This
                      formula assumes constant grid spacing over the
                      cloud altitude range.  Thus, if dz=1 then
                      ZCLOUD= 1,-5  and TCLOUD= 10,7  yeilds a linear
                      increase from zero.
```

```
NRE:          Cloud drop effective radius (microns).    (up to 5 values)
              Default value of NRE=8.

              The absolute value of NRE should be a floating point
              number in the range 2.0 to 128.0.
              NRE < 0 selects mie scattering parameters for ice particles
              NRE > 0 selects mie scattering parameters for water droplets

              The drop size distribution is assumed to follow a
              gamma distribution:

                              (p-1)  (-r/Ro)
              N(r) = C * (r/Ro)      e

              where C is a normalization constant [C=1./(Ro*gamma(p))],
              p=7, and Ro=NRE/(p+2)

              The factor (p+2) relating Ro to NRE follows
              from the defining equation of NRE:

                       3             2
              NRE = < r  N(r) > / < r  N(r) >,

              where the angle brackets indicate integration over all
              drop radii.

              Another frequently used parameter to describe the size
              distribution is the mode radius, Rm, which is defined
              as the radius at which N(r) is maximized.  For our
              drop size distribution Rm=(p-1)*Ro.  Using the relation
              between Ro and NRE we find that, Rm=(p-1)*NRE/(p+2)

              NOTE: If the first element of NRE is zero, the
              values of TCLOUD, ZCLOUD, LWP and NRE are ignored and
              cloud specification records are read from file usrcld.dat.
              The first record in this file corresponds to the
              lowest layer in the atmosphere, that is between the
              surface and the lowest cell boundary altitude. Each
              following record sets values for the next higher
              atmospheric layer in the model atmosphere.  usrcld.dat
              is read with the following fortran statements:

              do i=1,nz-1
                read(13,*,end=100) lwp(i),re(i),fwp(i),rei(i),cldfrac(i)
              enddo
            100 continue

              where
               lwp      liquid water path in layer i.          (g/m2)
                        (default=0)

               re       effective radius of liquid water       (um)
                        in layer i. (default=8um)

               fwp      frozen water path in layer i.          (g/m2)
                        if fwp < 0 then scattering parameters
                        are obtained from ccm3 cirrus model
                        (see subroutine icepar), if fwp > 0
                        then scattering parameters are obtained
                        from an internal mie scattering database
                        covering ice spheres with effective radii
                        between 2 and 128 um. (default=0)

               rei      effective radius of frozen water       (um)
                        in layer i.  only active when fwp
                        is non-zero.  if fwp.lt.0 and rei.le.0
                        then effective radius of ice is taken
                        from ccm3 cirrus model (see subroutine
                        icepar) (default=-1)

               cldfrac  cloud fraction in layer. this parameter
                        reduces cloud optical depth by factor
                        cldfrac**1.5 (default=1)

              It is not necessary to provide input records for
              layers above the highest cloud.  In addition, a
              forward slash terminates interpretation of data values
```

```
                in a record.  For example, the following records in
                usrcld.dat specify a cloud that extends from 2 to 4 km
                (assuming idatm>0 and no regridding):

                /                # no cloud between 0-1 km
                /                # no cloud between 1-2 km
                30 /             # lwp=30, re=10 between 2-3 km
                60 20 5 30 .2 /  # lwp=60, re=20 fwp=5 reice=30 cldfrac=.2
                                 #        between 3-4 km

                Any input quantities that are left unspecified will
                retain their default values of lwp=0. reff=8,
                fwp=0, reice=-1, and cldfrac=1.  The radiative
                properties of ice are computed from a CCM3 model (see
                subroutine ICEPAR).


IMOMC:          Controls the phase function model used in cloud layers:
                1  isotropic scattering
                2  rayleigh scattering phase function
                3  henyey_greenstein [a function of asymmetry factor, g(re)]
                4  haze L as specified by garcia/siewert
                5  cloud c.1 as specified by garcia/siewert

                (default=3)

LWP:            The liquid water path (or frozen water path if nre.le.0)
                of a cloud is specified in units of g/m2.  This is another
                way to specify cloud optical depth.

                A linearly varying opacity distribution can be
                obtained by setting the second LWP entry to a
                factor which represents the ratio of the opacity in
                the highest layer to that in the lowest layer
                For more details see the discussion of TCLOUD.

                NOTE: a 1 mm column of liquid water = 1000 g/m2,

                NOTE: LWP and TCLOUD cannot be used at the same time

                NOTE: The cloud optical depth is related to LWP by

                             3 Q(wl) * LWP
                      tau =  -------------
                             4  RHO * NRE

                      where Q is the scattering efficiency and RHO is
                      the density of liquid water (1 g/cm3).  The
                      value of Q that applies to a distribution of
                      cloud droplets can be expressed in terms of the
                      extinction cross-section at a given wavelength and
                      liquid drop radius.


                      Let      sigma = extinction cross-section at a given
                                       wavelength and drop radius

                                   q = sigma/(pi*r^2) (dimensionless)

                                       where (pi*r^2) is the geometrical
                                       cross-section of the cloud drop

                      then Q is a weighted average over drop radius,
                      given by:

                             2              2
                      Q = < r q N(r) > / < r  N(r) >

                      for visible light Q is typically about 2
                      (dimensionless).

                      For example:
                      NRE = 10um and LWP= 200g/m2 = 0.2mm =>   tau = 30


RHCLD:          The relative humidity within a cloud layer (a floating point
                value between 0.0 and 1.0).  RHCLD<0 disables the adjustment
```

```
                of relative humidity, in which case the relative humidity in
                the cloud layer follows solely from the temperature and
                water vapor density of the initial model atmosphere.  This
                parameter has no effect when KDIST<0

KRHCLR:         If zero, water vapor mixing ratio in clear layers is
                proportionately reduced to maintain the water vapor
                path specified by WH. This option has no effect if
                RHCLD is negative or TCLOUD is zero. (default)

                if 1, the relative humidity in clear layers is unchanged.

                NOTE: if KRHCLR=1 and clouds are present, the actual
                water vapor path will differ from that specified by WH.
                On the other hand, if KRHCLR=0, the normalization procedure
                may drive the water vapor in clear layers to zero and still
                be unable produce a given WVP.  This parameter has no
                effect when KDIST<0



                STRATOSPHERIC AEROSOLS (LOWTRAN 7 model)
                ========================================

JAER:           5 element array of stratospheric aerosol types

                0-no aerosol
                1-background stratospheric
                2-aged volcanic
                3-fresh volcanic
                4-meteor dust

ZAER:           altitudes (above the surface) of stratospheric aerosol
                layers (km) Up to 5 layer altitudes may be specified.
                NOTE: even though these models are for stratospheric
                aerosols, the scattering layer may be placed anywhere
                within the numerical grid. See ZCLOUD for a discussion
                of how aerosol (cloud) layers are positioned within
                SBDART's computational grid.

TAERST:         optical depth (at 0.55 microns) of each stratospheric
                aerosol layer.  Up to 5 layer optical depths may be
                specified.


                BOUNDARY LAYER AEROSOLS (BLA)
                =============================

IAER:           Boundary layer aerosol type selector

               -1-read aerosol optical depth and scattering parameters
                  from aerosol.dat. See subroutine AEREAD.

                  the file format is readable by the following Fortran code:

                  read(11,*) nn, moma
                  do k=1,huge(0)
                    read(11,*,end=100) wl(k)
                    do i=nz-nn+1,nz
                      read(11,*) dtau(i,k),waer(i,k),pmom(1:moma,i,k)
                    enddo
                  enddo
              100 continue

                  where
                    nn          is the number of atmospheric levels for
                                which aerosol information is specified.
                                layers 1 through nz-nn are unchanged.
                                (nn should be less than MXLY)

                    moma        number of phase function moments

                    wl(k)       is the wavelength [ wl(k) < wl(k+1) ]

                    dtau(i,k)   is the optical depth increment within
                                level i at wavelength k, information is
                                specified in top-down order.
```

```
    waer(i,k)   is the single scattering albedo

    pmom(m,i,k) are legendre moments of the phase function.
                Note that zeroeth moment is not read, it
                is assumed to be 1.


0-no boundary layer aerosols (all BLA parameters ignored)

1-rural

2-urban

3-oceanic

4-tropospheric

5-user defined spectral dependence of BLA

  The wavelength dependence of the aerosol scattering
  parameters are replaced by those read in from input
  parameters wlbaer, tbaer, wbaer and gbaer.  Between
  1 and 47 spectral values may be specified.


NOTE: the spectral dependence of the boundary layer aerosol
      models (IAER=1,2,3,4) vary with relative humidity.
      See subroutine AEROSOL for details.

NOTE: Don't be mislead by the term "boundary layer
      aerosol". The BLA models, IAER=1,2,3,4
      were originally developed to describe aerosols
      in the lower atmosphere.  However in SBDART, the
      default vertical density of BLA falls off
      exponentially, and affects regions above the
      normal extent of the boundary layer.  The
      vertical influence of these aerosols may be
      confined to a specified boundary layer altitude
      with the optional parameters ZBAER and DBAER.

RHAER:      The spectral dependence of the boundary layer aerosol
            scattering parameters are sensitive to relative humidity.
            Use input parameter RHAER to set the relative humidity
            used in the boundary layer aerosol model. Set RHAER=-1
            (the default value) to use the ambient surface relative
            humidity.  RHAER has no effect when IAER = 5.

VIS:        (Horizontal Path) Visibility (km) at 0.55 microns
            due to boundary layer aerosols.  This parameter does
            not set the optical depth for the user defined aerosol
            model (IAER=5), but does affect that model through
            the vertical structure (see below).


NOTE: unlike the stratospheric aerosols, the boundary
      layer aerosols have predefined vertical density
      distributions.  These vertical structure models
      vary with visibility.  (see discussion of ZBAER
      and DBAER)

NOTE: The boundary layer aerosol optical depth
      (absorption + scattering) at 0.55 microns
      is given by

      tauaero(0.55um) = 3.912 * integral ( n(z)/n(0) dz) / VIS

      where n(z) is the vertical profile of aerosol
      density.  For the 5 and 23 km visibility models
      the indicated integral is 1.05 and 1.51 km,
      respectively. So,

      tauaero(0.55um) = 3.912*(1.05*w+1.51*(1-w))/vis

      where w is a weighting factor between the two
      extremes and is given by
```

$$w = \frac{(1/vis - 1/23)}{(1/5 - 1/23)} \quad , \quad 5 < vis < 23$$

$$w = 1 \quad , \quad vis < 5$$

$$w = 0 \quad , \quad vis > 23$$

NOTE: Visibility is defined as the horizontal distance in km at which a beam of light at 0.55um is attenuated by a factor of 0.02.

$n(0)*sigma*VIS = -ln(.02)$, or

$VIS = 3.912/(n(0)*sigma)$

where sigma is the aerosol absorption+scattering cross-section at 0.55 microns. See Glossary of Meteorology, American Meteorology Society, 1959

ZBAER: Altitude grid for custom aerosol vertical profile (km) Up to MXLY altitude points may be specified. ZBAER is active for all positive values of IAER.

DBAER: Aerosol density at ZBAER altitude grid points, active for all positive values of IAER. Up to MXLY density values may be specified. The number of density values must match the number of ZBAER. The units used to specify aerosol density is arbitrary, since the overall profile is scaled by the user specified total vertical optical depth. The aerosol density at all computational grid points is found through logarithmic interpolation on the ZBAER and DBAER values. The normal vertical profile from 5s is used when DBAER is unset.

For example

```
ZBAER=0,1,100
DBAER=1000,500,1
```

specifies a aerosol density profile that drops by a factor 2 (exponential fall off) between 0 and 1km altitude and then by a factor of 500 between 1 and 100 km.

If DBAER is set but ZBAER is not set, then the elements of DBAER are used to set the aerosol density for each computational layer, starting from the bottom layer.

For example,

```
DBAER=10,0,1,0
```

puts aerosol in the first and third layer.

If neither ZBAER or DBAER are set, the boundary layer aerosols are assumed to follow a pre-defined vertical distribution which drops off exponentially with a scale height between 1.05 and 1.51 km depending visibility (see VIS). Thus, even if visibility is not used to set the vertical optical depth it can affect the result through the vertical profile. Note that ZBAER and DBAER do not affect the total vertical optical depth of aerosols. (See discussion for VIS).

TBAER: Vertical optical depth of boundary layer aerosols at 0.55 um. TBAER input is significant for all values of IAER. When IAER=1,2,3,4 the specified value of TBAER supersedes the aerosol optical depth derived from input parameter VIS (but VIS still controls vertical structure model unless DBAER and ZBAER are set).

QBAER: QBAER is the extinction efficiency. QBAER is only active when IAER=5. When TBAER is set, QBAER sets the spectral dependence of the extinction optical depth as,

tau= tbaer * Qext(wave_length)/Qext(0.55um)

where Qext(wave_length) = QBAER interpolated to wave_length

If TBAER is not set, then the values of QBAER are interpreted as extinction optical depths at each wavelength WLBAER.

For example, the Multi Filter Rotating Shadowband Radiometer (MFRSR) installed at the Southern Great Plains ARM site is able to retrieve aerosol optical depth in 6 SW spectral channels. This information may be supplied to SBDART by setting,

```
wlbaer= .414,  .499,  .609,  .665,  .860,  .938
qbaer= 0.109, 0.083, 0.062, 0.053, 0.044, 0.041
wbaer=6*.9
gbaer=6*0.8
```

This spectral information is iterpolated or extrapolated to all wavelengths using logarithmic fitting on QBAER and linear fitting on WBAER and GBAER. Many aerosol types display a power law dependence of extinction efficiency on wavelength. The logarithmic interpolation/extrapolation on QBAER will reproduce this behavior if it exists in the input data.

WLBAER: Wavelengths points (um) for user defined aerosol spectral dependence. Only used when IAER=5. WLBAER (and QBAER) need not be specified if a single spectral point is set. In this case the aerosol optical depth is extrapolated to other wavelengths using a power law (see ABAER)

WBAER: Single scattering albedo used with IAER=5.

WBAER represents the single scattering albedo of boundary layer aerosols at wavelengths WLBAER.

GBAER: Asymmetry factor used with IAER=5

GBAER represents the asymmetry factor of boundary layer aerosols at wavelengths WLBAER. Number of values must match the number of WLBAER.

GBAER is ignored when parameter PMAER is set or when IMOMA .ne. 3

PMAER: Legendre moments of the scattering phase function of boundary layer aerosols, only active for IAER=5. The Legendre moments of the phase function are defined as the following integral over the scattering phase function, f:

$$\text{pmaer}(i) = \frac{\int f(\mu)\, P(i,\mu)\, d\mu}{\int f(\mu)\, d\mu}$$

where P(i,mu) is the Legendre polynomial, mu is the cosine of the scattering angle, and the range of the integrals are from -1 to 1. The Legendre moment for i=0 is always one. Hence, the zero'th moment is assumed by SBDART and should not be specified.

Unlike the previous boundary layer aerosol parameters, you need to specify at least NSTR values for each wavelength point, for a total of NSTR*NAER values, where NAER is the number of wavelength points supplied. The order of specification should be such that wavelength variation is most rapid. For example, here is a case with 4 wavelengths and 6 streams:

```
nstr=6
wlbaer=.400,.500,.600,.700
pmaer= 0.80,0.70,0.60,0.50,
       0.64,0.49,0.36,0.25,
       0.51,0.34,0.22,0.12,
```

```
                        0.41,0.24,0.13,0.06,
                        0.33,0.17,0.08,0.03,
                        0.26,0.12,0.05,0.02

ABAER:          Wavelength (Angstrom model) exponent used to extrapolate
                BLA extinction efficiency to wavelengths outside the
                range of WLBAER [Qext ~ (lambda)^(-abaer)].  This parameter
                is only operative when IAER=5.

                If ABAER is set to a positive number, then that value
                is used as a power-law wavelength dependence to
                extrapolate the extinction efficiency for wavelengths
                less than WLBAER(1) or greater than WLBAER(nn) (where
                nn is the number of specified values).  If ABAER is
                not set, the wavelength extrapolation is based on the
                last two specified points (wlbaer(1),wlbaer(2) or
                wlbaer(nn-1),wlbaer(nn)).  If ABAER is not set and a
                single wavelength is set, then a spectrally constant
                extinction efficiency is used.

IMOMA:          Controls phase function used for boundary layer aerosol
                The value of IMOMA is ignored when IAER=5 and PMAER is
                specified.  Note that an asymmetry factor must be
                specified when IMOMA=3 (default=3).

                1  isotropic scattering
                2  rayleigh scattering
                3  henyey_greenstein(g(re))                <-- default
                4  haze L as specified by garcia/siewert
                5  cloud c.1 as specified by garcia/siewert

SPOWDER:        Setting SPOWDER to true causes an extra sub-surface layer
                extending between -1 and 0 km to be added to the bottom of
                the atmospheric grid.  This layer may be used to model the
                effects of surface reflection and thermal emission caused
                by a granular surface material (e.g., snow or sand).  The
                scattering properties of the surface layer may be specified
                either with the cloud or aerosol inputs.  For example, a
                thin water cloud over a snow surface composed of 100um
                snow grains may be modeled with the following input file:

                 &INPUT
                   sza=30,     idatm=4,   spowder=t
                   wlinf=.3,   wlsup=2,   iout=1
                   tcloud= 10000,  10
                   zcloud=    -1,   2
                   nre=     -100,  10
                 /

                Similarly, the scattering properties of the surface and
                atmosphere may be read from aerosol.dat with IAER=-1.
                To model a semi-infinite granular surface layer the optical
                depth of the bottom layer should be made very large, e.g.,
                10000, as indicated in the example. However, a smaller
                optical depth may also be specified in conjuntion with a
                given value of sub-surface albedo selected with ISALB.
                Thus, in the previous example the effect of a thin snow
                layer covering a grass field may be modeled by setting,
                tcloud=100,10 and isalb=6

                NOTE: At present there is no way to set the surface
                (skin) temperature to something other than the atmospheric
                temperature of the bottom level, i.e. the parameter BTEMP
                is ineffective when SPOWDER is true.

                ================================================================


NOTHRM:          nothrm=-1 => Thermal emission turned on only for wavelengths
                              greater than 2.0 um (default)

                              (Note: During daylight hours solar
                              radiation is a factor of about 1.e5
                              greater than thermal radiation at 2.0um)

                 nothrm=0  => Thermal emission turned on for all wavelengths
```

```
                nothrm=1  => No thermal emission

                NOTE: If thermal emission is desired, be sure that the
                      temperature steps in the atmospheric model are small
                      enough to resolve changes in the Planck function.  The
                      original version of the DISORT radiative transfer
                      module issued a warning message if the temperature
                      difference between successive levels in the atmosphere
                      exceeded 20 K.  All the standard atmospheres violate
                      this condition for at least 1 stratospheric layer.
                      This warning message has been disabled to avoid
                      clutter in SBDART's standard output. If near-IR thermal
                      emission from the stratosphere is important to your
                      application, you should supply SBDART with a new model
                      atmosphere with higher resolution in the stratosphere.
                      (see ZGRID1, ZGRID2, an NGRID)


KDIST:          KDIST=-1 causes correlated-k optical depths and weighting
                factors to be read from files CKATM and CKTAU.

                NOTE: KDIST=-1 disables the effect of all input
                parameters that control aspects of the gaseous
                atmospheric profile.  Thus, KDIST=-1 disables input
                parameters AMIX, SCLH2O, UW, UO3, O3TRP, ZTRP, XN2,
                XO2, XCO2, XCH4, XN2O, XCO, XNO2, XSO2, XNH3, XNO,
                XHNO3, XO4, RHCLD, KRHCLR, NGRID, ZGRID1, ZGRID2,
                PBAR, and ZPRES.

                KDIST=0 causes the optical depth due to molecular
                absorption to be set to the negative log of the LOWTRAN
                transmission function.  This approximation is not
                appropriate for cases in which multiple scattering is
                important, but is not very wrong when the molecular
                absorption is weak or the scattering optical depth is
                small.

                KDIST=1 causes SBDART to use the LOWTRAN7
                k-distribution model of absorption by atmospheric
                gases.  Since a three term exponential fit is used,
                SBDART execution times are up to 3 times longer with
                KDIST > 0 compared to KDIST=0.

                KDIST=2 causes the k-fit transmissions to exactly
                match the LOWTRAN transmission along the solar beam
                direction.  This option may be useful when computing
                surface irradiance under clouds of optical thickness
                less than about 10.  This is because in this thin
                cloud case much of the radiation which reaches the
                surface propagates along the direct beam direction.

                KDIST=3 causes the k-fit transmissions to exactly
                match the LOWTRAN transmission along the solar beam
                direction for parts of the atmosphere above a scattering
                layer.  As the scattering optical depth increases above
                1 the k-fit factors are ramped back to there original
                LOWTRAN values.  The effect of the slant path correction
                is ramped down to zero for wavelengths greater than 4um,
                where solar energy input is less important.  KDIST=3 is
                the default.

ZGRID1:         These three parameters can be used to change the grid
ZGRID2:         resolution of the model atmosphere.  ZGRID1 controls
NGRID:          the resolution near the bottom of the grid while
                ZGRID2 sets the maximum permissible step size (at the
                top of the grid).  NGRID sets the number of grid
                points.  For example ZGRID1=.5, ZGRID2=30, NGRID=45
                specifies a 45 element grid with a resolution of .5 km
                throughout the lower part of the grid and a largest step
                of 30 km.

                The regridding is performed after the call to
                subroutine ATMS.  This allows regridding of the
                standard internal atmospheres as well as user
                specified atmospheres (read with IDATM=0).  No matter
                how many grid points were used to specify the original
```

```
                atmosphere, the new regridded atmosphere will contain
                NGRID vertical array elements.  The default value of
                ZGRID1 and ZGRID2 are set to 1 and 30km, respectively.
                The default value of NGRID=0, causes the initial un-modified
                atmsopheric model to be used.  The internal parameter,
                MXLY, sets the maximum number of levels allowed.  Setting
                NGRID>MXLY causes NGRID to be set to MXLY.

                If NGRID is negative SBDART terminates execution
                after printing out the regridded values of Z,P,T,WH,WO
                to standard out.  This option can be used to preview the
                effect of a given set of ZGRID1,ZGRID2 and abs(NGRID)
                values.

                OUTPUT OPTIONS
                ==============

IDB:            DIAGNOSTIC OUTPUT SELECTOR (integer array)

                The IDB print flag is used to select print diagnostics
                for a variety of computational parameters.  Setting
                IDB(n)=m where m is any non-zero integer, will produce
                the diagnostics associated with array index n to be listed.
                For some values of n, increasing the value of m (e.g.,
                IDB(8)=2) will produce more detailed diagnostics.  (default=0)

                array
                index
                  1  print an explanation of quantities in IOUT output group

                  2  cloud liquid water profiles

                  3  for kdist>-1: atmospheric profile and gas absorption
                     integrals
                     for kdist=-1: atmospheric profile read from CKATM
                     also lists wavenumbers specified in CKTAU.
                     idb(3)=2 causes sbdart to exit after diagnostics.

                  4  solar ephemeris, grid parameters, atmospheric profiles

                  5  cloud parameters, extinction efficiency, asymmetry
                     factor and single scatter albedo

                  6  aerosol single scattering albedo assymetry factor
                     optical depth increments of total (taua) and boundary
                     layer aerosols (tauab).  The total optical depth
                     is displayed on the final line of output.

                  7  gaseous absorption integrals and optical depth

                  8  Optical depth due to Rayleigh, aerosols, cloud
                     molecular continuum and line, single scattering albedo
                     and asymmetry factor. Additional printouts for each
                     term in the k-fit are produced if KDIST=1.

ZOUT:           2 element array specifying BOT and TOP altitude
                points (km) for IOUT output.  For example ZOUT=0,50
                specifies output information for 0 and 50 km. The
                surface is always set at zero.  Note that the actual
                layers for which output is generated is determined by
                finding the atmospheric layers nearest the chosen
                value of ZOUT(1) and ZOUT(2).  (default = 0,100)


IOUT:           STANDARD OUTPUT SELECTOR
     value
     -----
       0.       no standard output is produced, DISORT subroutine
                is not called, but diagnostics selected by idb in gas
                absorption or aerosol subroutines are active.

       1.       one output record for each wavelength,
                output quantities are,

                   WL,FFV,TOPDN,TOPUP,TOPDIR,BOTDN,BOTUP,BOTDIR

                   WL    = wavelength                          (microns)
```

```
            FFV   = filter function value
            TOPDN = total downward flux at ZOUT(2) km  (w/m2/micron)
            TOPUP = total upward flux at ZOUT(2) km    (w/m2/micron)
            TOPDIR= direct downward flux at ZOUT(2) km (w/m2/micron)
            BOTDN = total downward flux at ZOUT(1) km  (w/m2/micron)
            BOTUP = total upward flux at  ZOUT(1) km   (w/m2/micron)
            BOTDIR= direct downward flux at ZOUT(1) km (w/m2/micron)

            NOTE: When ISAT ne 1 these radiometric quantities
                  are each multiplied by the filter function,
                  To get the actual specific irradiance divide
                  by FFV(WL).

      2.    one output record per wavelength, not available for kdist=-1
            output quantities are,

          WL,TXH2O,TXCO2,TXO3,TXN2O,TXCO,TXCH4,TXO2N2,TXTRC,TXTOT,TXMOL

            WL     = wavelength
            TXH2O  = -log transmission due to water vapor
            TXCO2  = -log transmission due to co2
            TXO3   = -log transmission due to ozone
            TXN2O  = -log transmission due to n2o
            TXCO   = -log transmission due to co
            TXCH4  = -log transmission due to ch4
            TXO2N2 = -log transmission due to o2 and n2
            TXTRC  = -log transmission due to trace gases
            TXTOT  = -log transmission due to all gases
            TXMOL  = optical depth due to rayleigh scattering

            NOTE: if you define the optical depth as
                  transmission = exp(-tau) then
                  -log transmission = tau

      3.    Averaged gas absorption over solar spectrum and
            filter function. Not available when kdist=-1.
            Output format:

            write(*,'(5x,11a13)') 'z','airmass','h2o','co2','o3',
           &            'n2o','co','ch4','o2+n2','trace','total'
            do j=nz,1,-1
               write(*,'(i5,1p11e13.5)') j,z(j),airmass(j),
           &    (-log(eps+trnsgas(i,j)/phidw),i=1,nta)

            where j is the level index
                  z is the level height (km)
                  airmass = g * integral(rho dz/mu) / Pzero
                      where g=9.8m/s2, pzero 1013.25mb
                      rho is the mass desity of air, and
                      mu is the cosine of the solar zenith angle (SZA)
                  trnsgas is the transmission due to the species
                      listed in the title line

            the output quantity is the negative log of the transmission
            which, aside from non-Beer's law behaviour, is like
            optical depth.  If the input quantity NF is non-zero
            then the transmission is averaged over the solar spectrum.
            If NF=0 the average is over the filter function.
            Remember to set NF=0 and SZA=0 when dealing with LW radiation.



      5.    nzen+3) records for each wavelength.  Output format:
                 write(*,*) '"tbf'     ; Block id (used in postprocessors)

                 do m=1,nw
                   write(*,*)
                &    wl,ffv,topdn,topup,topdir,botdn,botup,botdir
                   write(*,*)  nphi,nzen
                   write(*,*) (phi(j),j=1,nphi)
                   write(*,*) (uzen(j),j=1,nzen)
                   do i=nzen,1,-1
                     write(*,*) (uurs(i,k),k=1,nphi)
                   enddo
                 enddo

            where,
```

```
   WL    = wavelength                          (microns)
   FFV   = filter function value
   TOPDN = total downward flux at ZOUT(2) km  (w/m2/micron)
   TOPUP = total upward flux at ZOUT(2) km    (w/m2/micron)
   TOPDIR= direct downward flux at ZOUT(2) km (w/m2/micron)
   BOTDN = total downward flux at ZOUT(1) km  (w/m2/micron)
   BOTUP = total upward flux at  ZOUT(1) km   (w/m2/micron)
   BOTDIR= direct downward flux at ZOUT(1) km (w/m2/micron)
   NPHI  = number of user azimuth angles
   NZEN  = number of user zenith angles
   PHI   = user specified azimuth angles      (degrees)
   UZEN  = user specified zenith angles       (degrees)
   VZEN  = user specified nadir angles        (degrees)
   UURS  = radiance at user angles at         (w/m2/um/str)
           altitude ZOUT(2) (top)


NOTE: The radiance output from SBDART represents
      scattered radiation. It does not include the
      solar direct beam.  Also, keep in mind that UURS
      represents the radiance at the user specified
      sample directions.  Hence, computing the
      irradiance by an angular integration of UURS
      will not yield BOTDN because of the neglect of
      the direct beam, and it will probably not yield
      (BOTDN-BOTDIR) because of under-sampling.


   NOTE: if IDAY is set, then PHI is the actual compass
         direction in which the radiation in propagating
```

6. same as IOUT=5 except radiance is for ZOUT(1) altitude (bottom)

7. radiative flux at each layer for each wavelength. This output option can produce a huge amount of output if many wavelength sample points are used

```
   write(*,*) '"fzw'       ; block id (used in postprocessors)
   write(*,*) nz           ; number of z levels
   write(*,*) nw           ; number of wavelengths

   do j=1,nw
     write(*,*) wl
     write(*,*)
  &    (Z(i),i=nz,1,-1),   ; altitude              (km)
  &    (fdird(i),i=1,nz),  ; downward direct flux  (w/m2/um)
  &    (fdifd(i),i=1,nz),  ; downward diffuse flux (w/m2/um)
  &    (flxdn(i),i=1,nz),  ; total downward flux   (w/m2/um)
  &    (flxup(i),i=1,nz)   ; total upward flux     (w/m2/um)
   enddo
```

10. one output record per run, integrated over wavelength. output quantities are, (integrations by trapezoid rule)

```
   WLINF,WLSUP,FFEW,TOPDN,TOPUP,TOPDIR,BOTDN,BOTUP,BOTDIR

   WLINF = lower wavelength limit             (microns)
   WLSUP = upper wavelength limit             (microns)
   FFEW  = filter function equivalent width   (microns)
   TOPDN = total downward flux at ZOUT(2) km  (w/m2)
   TOPUP = total upward flux at ZOUT(2) km    (w/m2)
   TOPDIR= direct downward flux at ZOUT(2) km (w/m2)
   BOTDN = total downward flux at ZOUT(1) km  (w/m2)
   BOTUP = total upward flux at  ZOUT(1) km   (w/m2)
   BOTDIR= direct downward flux at ZOUT(1) km (w/m2)
```

11. radiant fluxes at each atmospheric layer integrated over wavelength. Output format:

```
   write(*,*) nz,phidw
   do i=1,nz
     write(*,*) zz,pp,fxdn(i),fxup(i),fxdir(i),dfdz,heat
   enddo
```

```
 where,  nz    = number of atmospheric layers
         ffew  = filter function equivalent width(um)
         zz    = level altitudes                 (km)
         pp    = level pressure                  (mb)
         fxdn  = downward flux (direct+diffuse)  (W/m2)
         fxup  = upward flux                     (W/m2)
         fxdir = downward flux, direct beam only (W/m2)
         dfdz  = radiant energy flux divergence  (mW/m3)
         heat  = heating rate                    (K/day)

NOTE:  dfdz(i) and heat(i) are defined at the layer
centers, i.e., halfway between level i-1 and level i.
```

20. radiance output at ZOUT(2) km.

Output format:

```
   write(*,*)  wlinf,wlsup,ffew,topdn,topup,topdir,
  &            botdn,botup,botdir
   write(*,*)  nphi,nzen
   write(*,*)  (phi(i),i=1,nphi)
   write(*,*)  (uzen(j),j=1,nzen)
   write(*,*)  ((r(i,j),i=1,nphi),j=1,nzen)
```

The first record of output is the same as format IOUT=10 (WLINF,WLSUP,FFEW,TOPDN,TOPUP,TOPDIR,BOTDN,BOTUP,BOTDIR) addition records contain:

```
   NPHI  = number of user azimuth angles
   NZEN  = number of user zenith angles
   PHI   = user relative azimuth angles (nphi values)
   UZEN  = user zenith angles (nzen values)
   R     = radiance array (nphi,nzen) (W/m2/sr)

   NOTE: if IDAY is set, then PHI is the actual compass
         direction in which the radiation in propagating
```

21. same as IOUT=20 except radiance output at ZOUT(1) km.

22. radiance and flux at each atmospheric layer integrated over wavelength.

Output format:

```
   write(*,*) nphi,nzen,nz,ffew
   write(*,*) (phi(i),i=1,nphi)
   write(*,*) (uzen(j),j=1,nzen)
   write(*,*) (z(k),k=nz,1,-1)
   write(*,*) (fxdn(k),k=1,nz)
   write(*,*) (fxup(k),k=1,nz)
   write(*,*) (fxdir(k),k=1,nz)
   write(*,*) (((uurl(i,j,k),i=1,nphi),j=1,nzen),k=1,nz)

 where,  nphi  = number of user specified azimuth angles
         nzen  = number of user specified zenith angles
         nz    = number of atmospheric levels
         ffew  = filter function equivalent width (um)
         phi   = user specified anizmuth angles   (degrees)
         uzen  = user specified zenith angles     (degrees)
         z     = altitudes of atmospheric layers  (km)
         fxdn  = downward flux (direct+diffuse)   (W/m2)
         fxup  = upward flux                      (W/m2)
         fxdir = downward flux, direct beam only  (W/m2)
         uurl  = radiance at each layer           (W/m2/str)

   NOTE: if IDAY is set, then PHI is the actual compass
         direction in which the radiation in propagating
```

23. same as IOUT=20 except
lower hemisphere radiance output corresponds to ZOUT(1)
upper hemisphere radiance output corresponds to ZOUT(2)
Use this output format to determine radiance above and
and below a scattering layer. For example, if ZCLOUD=1
and TCLOUD=10, you can get the scattered radiation field
above and below the cloud with, IOUT=23, ZOUT=1,2.

```
                NOTE: if IDAY is set, then PHI is the actual compass
                      direction in which the radiation in propagating

=========================================================================


                    DISORT options
                    ==============

 DELTAM:       if set to true, use delta-m method (see Wiscombe, 1977).
               This method is essentially a delta-Eddington
               approximation applied to multiple radiation streams.

               In general, for a given number of streams, intensities
               and fluxes will be more accurate for phase functions
               with a large forward peak if 'DELTAM' is set
               TRUE. Intensities within 10 degrees or so of the
               forward scattering direction will often be less
               accurate, however, so when primary interest centers in
               this so-called 'aureole region', DELTAM should be set
               FALSE.(default=true)

 NSTR:         number of computational zenith angles used.  NSTR must
               be divisible by 2.  Using NSTR=4 reduces the time
               required for flux calculations by about a factor of 5
               compared to NSTR=16, with very little penalty in
               accuracy (about 0.5% difference when DELTAM is set
               true).

 CORINT:       When set TRUE, correct intensities for
               delta-M scaling effects (see Nakajima and Tanaka,
               1988). When FALSE, intensities are not corrected.
               In general, CORINT should be set true when beam
               source is present (FBEAM is not zero) and DELTAM
               is TRUE in a problem including scattering.
               However, execution is faster when CORINT is FALSE,
               and intensities outside the aureole may still be
               accurate enough.  When CORINT is TRUE, it is
               important to have a sufficiently high order of
               Legendre approximation of the phase function. This
               is because the intensities are corrected by
               calculating the single-scattered radiation, for
               which an adequate representation of the phase
               function is crucial.  In case of a low order
               Legendre approximation of an otherwise highly
               anisotropic phase function, the intensities might
               actually be more accurate when CORINT is FALSE.

               Default value=.false.

               The input value of CORINT is ignored for:
               1) in irradiance mode, i.e., iout ne (5,6,20,21,22)
               2) there is no beam source (FBEAM=0.0), or
               3) there is no scattering (SSALB=0.0 for all layers)

                      Radiance output
                      ===============


 NZEN:         Number of user zenith angles.  If this parameter is
               specified SBDART will output radiance values at NZEN
               zenith angles, evenly spaced between the first two
               values of input array UZEN.  For example,

                nzen=9,
                uzen=0,80

               will cause output at zenith angles 0,10,20,30,40,50,60,70,80.

 UZEN:         User zenith angles.  If NZEN is specified then UZEN is
               interpreted as the limits of the zenith angle range,
               and only the first two elements are required.  If NZEN
               is not specified then up to NSTR values of UZEN may be
               specified.  If neither NZEN nor UZEN is specified and
```

```
                a radiance calculation is requested
                (IOUT=5,6,20,21,22,23) a default set of zenith angles
                is used, which depends on the value IOUT as follows:
                      * IOUT=5 or 20:  NZEN=18, UZEN=0,85
                      * IOUT=6 or 21:  NZEN=18, UZEN=95,180
                      * IOUT=22 or 23: NZEN=18, UZEN=0,180

                NOTE: UZEN specifies the zenith angle of at which the
                radiation is propagating:

                UZEN = 0    => radiation propagates directly up
                UZEN < 90   => radiation in upper hemisphere
                UZEN > 90   => radiation in lower hemisphere
                UZEN = 180  => radiation propagates directly down

VZEN:           user nadir angles.  This is just an alternate way to
                specify the direction of user radiance angles, whereby
                uzen=180-vzen

NPHI:           Number of user azimuth angles.  If this parameter is
                specified SBDART will output radiance values at NPHI
                azimuth angles, evenly spaced between the first two
                values of input array PHI.  For example, nphi=7,
                phi=0,180 will cause output at zenith angles
                0,30,60,90,120,150,180

PHI:            User relative azimuth angles.  If NPHI is specified then
                PHI is interpreted as the limits of the azimuth angle range,
                and only the first two elements are required.  If NPHI
                is not specified then up to NSTR values of PHI may be
                specified.  If neither NPHI nor PHI is specified and a
                radiance calculation is requested (IOUT=5,6,20,21,22,23)
                a default set of azimuth angles is used, equivalent to
                the case NPHI=19, PHI=0,180.

                NOTE: Azimuth increases clockwise looking down on the
                Earth's surface. PHI is the relative azimuth angle from
                the forward scattering direction.

                * PHI < 90   => forward scattered radiation
                * PHI > 90   => backward scattered radiation

                For example, if the sun is setting in the West,
                radiation propagating to the South-East has a relative
                azimuth of 45 degrees.

                NOTE: SBDART is currently configured to model radiation
                      with at most 40 computational zenith angles and
                      40 azimuthal modes.  While these limits may be
                      expanded, be aware that running SBDART with a
                      much larger number will significantly increase
                      running time and memory requirements.  In tests
                      performed on a DEC Alpha, the execution time
                      scaled roughly with NSTR^2, for NSTR less than
                      40.  The code's memory usage also scales roughly
                      as NSTR^2.

PHI0:           azimuth angle of incident beam.  Use this parameter
                to relate the radiance output to fixed navigational
                headings.  For example if the sun is positioned at
                zenith=10 and azimuth=110 degrees (measured clockwise
                from due north) then setting PHI0=-70 degrees will
                cause PHI to be interpreted as a compass direction.  In
                this example the forward scattering peak will be found
                at uzen=170, phi=-70.  Otherwise if PHI0 is zero (the
                default value), PHI is interpreted as a relative
                azimuth angle (i.e., relative to the forward scattering
                direction).

                NOTE: When IDAY is set, PHI0 is automatically set
                to the correct solar azimuth, and the input value of
                PHI0 is ignored.


                     radiation boundary conditions
                     =============================
```

```
 IBCND:
       = 0 : general case: boundary conditions any combination of:
              * beam illumination from the top      ( see FBEAM)
              * isotropic illumination from the top ( see FISOT)
              * thermal emission from the top       ( see TEMIS,TTEMP)
              * internal thermal emission sources   ( see TEMPER)
              * reflection at the bottom            ( see LAMBER,ALBEDO,HL)
              * thermal emission from the bottom    ( see BTEMP)
       = 1 : isotropic illumination from top and bottom, in order to get
             ALBEDO and transmissivity of the entire medium vs. incident
             beam angle;
             The only input variables considered in this case are
             NLYR,DTAUC,SSALB,PMOM,NSTR,USRANG,NUMU,UMU,ALBEDO,DELTAM,
             PRNT,HEADER, and the array dimensions.
             NOPLNK,LAMBER are assumed TRUE, the bottom boundary can have
             any ALBEDO.  the sole output is ALBMED,TRNMED.  UMU is
             interpreted as the array of beam angles in this case.
             If USRANG = TRUE they must be positive and in increasing
             order, and will be returned this way; internally, however, the
             negatives of the UMU's are added, so MAXUMU must be at least
             2*NUMU.
             If USRANG = FALSE, UMU is returned as the NSTR/2 positive
             quadrature angle cosines, in increasing order.

 FISOT:     intensity of top-boundary isotropic illumination.  (units
            w/sq m if thermal sources active, otherwise arbitrary
            units).  corresponding incident flux is pi (3.14159...)
            times 'FISOT'.

 TEMIS:     emissivity of top layer, not used if NOTHRM=1

 BTEMP:     Surface (skin) temperature in Kelvin.  If not set, the
            surface temperature is set to the temperature of the
            bottom layer of the model atmosphere.  The surface
            emissivity is calculated from the albedo (see ISALB and
            ALBCON).  The input value of BTEMP is ignored when
            NOTHRM=1 or SPOWDER=.true.


          DISORT specific output options:
          ===============================
PRNT(k):   k
           1        input variables (except PMOM)
           2        fluxes
           3        intensities at user levels and angles
           4        planar transmissivity and planar albedo
                    as a function solar zenith angle ( IBCND = 1 )
           5        phase function moments PMOM for each layer
                    ( only if PRNT(1) = TRUE, and only for layers
                    with scattering )

=========================================================================
```