RADS User Manual

Remko Scharroo

Version 4.2.4
14 June 2016

RADAR ALTIMETER DATABASE SYSTEM

TU Delft

EUMETSAT

NOAA
Laboratory for Satellite Altimetry
NOAA-NESDIS-STAR

This document was typeset with LaTeX 2$_\varepsilon$.

Contents

Chapter 1

Introduction

This document describes the layout and use of the Radar Altimeter Database System (RADS), Version 4. RADS was first developed at Delft University of Technology's Department of Aerospace Engineering, and remains a joint development with NOAA Laboratory for Satellite Altimetry and EUMETSAT.

The Radar Altimeter Database System is composed of three elements:

- A few hundred gigabytes of altimeter data files from missions stretching from Geosat to whatever altimeter data was made available in the last few days;
- A set of software tools (object library and executables);
- Configuration files.

So apart from the actual altimeter data, RADS provides a suite of applications and subroutines that simplify the reading, editing, handling and analysing of data from numerous radar altimeters. Although the actual content and layout of the underlying data products do not have to be identical for all altimeters, the user interface is. Also, the data base is easily expandable with additional data and/or additional corrections without impact to the user interface, or to the software in general. In fact, only in very few cases the core software will need to be adjusted and recompiled, in even fewer cases adjustments to the actual tools will be required. Most changes can be covered by changes in the configuration file.

The data base consists of netCDF files, one for each satellite pass (half a revolution starting and ending close to the poles). Ascending passes have odd numbers, descending passes even numbers. The pass numbering increases consecutively within a repeat cycle.

In case of exact repeat missions the satellite returns to the same ground track every repeat cycle. For Jason-2, for example, this is after 254 passes, when the pass number starts over at 1. Which pass is number 1 is based on the longitude of the equator crossing (ascending node). Thus all passes with the same pass number are collinear.

For non-repeat missions or those with very long repeat cycles (like CryoSat-2 or the Jason-1 Extended Mission), we created "sub-cycles" of a manageable length. There too passes with the same pass number are nearly collinear. Note that the length of the "sub-cycle" may change for cycle to cycle in a kind of dance-step manner.

Each netCDF data file contains the actual (binary) data as well as the meta data that describe the contents (data type, units, creation history, etc.) The naming convention for the files is `SSpPPPPcCCC.nc`, where `SS` is an abbreviation for the satellite (altimeter), `PPPP` is the pass

number, `CCC` is the cycle number, and `nc` is the extension, a standard convention for netCDF data files.

The data files are grouped in one directory for each cycle, named `cCCC`. These cycle directories are then grouped into one directory for each mission phase, which are finally part of one directory per satellite. For example, the data file for pass 801 of cycle 150 in ERS-1's tandem mission is `$RADSDATAROOT/e1/g/c150/e1p0801c150.nc`, where `$RADSDATAROOT` is the root directory of the RADS data base.

To read and manipulate the data, you can use standard netCDF tools, like `ncdump` (that comes with the netCDF package), GMT (Generic Mapping Tools), nco (NetCDF Operators). But more suitable is the use of the RADS subroutine library and programs. The library is the basis for all data utilities provided with RADS and can also be used to create other programs to the user's convenience. For a description on each of the subroutines in the library and on how to create your own program see Appendix 6. In addition, a number of handy utilities are provided to do some of the most essential jobs (Chapter 5).

Whether you are using the routines, or the provided utilities, you will have to know how the data handling system of RADS works. It is not essential that you understand the intrinsics of the data files, but it is highly recommended that you familiarise yourself with the way the data can be manipulated, selected and edited *on the fly* by the RADS routines. Basically, the RADS routines can take you a lot of work out of your hands, provided you have read Chapter 4.

Before going into the details of RADS, the software and the data have to be installed on your computer. Chapter 2 guides you through the process of software installation, and Chapter 3 tells you how to keep the database up to date.

Chapter 2

RADS software installation

In order to work efficiently with the RADS data base you are required to install the software (subroutine library, utilities, scripts, and configuration files). This we will tackle in this Chapter. Once you are done with that at least part of the data base needs to be copied onto your hard disk (or another mounted device), which will be described in the next Chapter.

2.1 Prerequisites

In order to install and run the RADS software you need a few things installed on your system:

- A unix platform (for example Linux or Mac OS X).
- The make command.
- A Fortran 90 compiler. RADS is known to compile with gfortran, f90, f95, xlf90, xlf95, ifort.
- The netCDF library (version 4) and module file compiled with the Fortran 90 interface. Of course, netCDF comes with its own dependencies (like HDF5 and szip). Please figure out where to find the netCDF module file netcdf.mod and the netCDF C library libnetcdf and Fortran library libnetcdff before you continue.
- Optionally, the git program.
- For downloading and synchronising the data base, the rsync program.

2.2 Download the source code

The source code can be downloaded as a bundle (zip or tarball) from GitHub or can be synchronised directly with the github server with the git program. The two methods are described below in Sections 2.2.1 and 2.2.2.

You can put the source code anywhere you like. We will later configure where things will be installed. After downloading the software, continue with the configuration, compilation, and installation steps in Sections 2.3 through 2.5.

It is recommended to regularly check for updates of the RADS source code and recompile if necessary.

2.2.1 Download the bundle from GitHub

To download the latest bundle of the source code, simple go to https://github.com/remkos/rads/releases/latest. There you will find the latest release notes, and links for the downloading of the bundle, either as a zip file, or as a compressed tarball.

You can extract the software in place, or anywhere you want by running:

```
$ tar -xvzf rads-v4.2.4.tar.gz
```

or

```
$ unzip rads-v4.2.4.zip
```

This will create a directory called `rads-v4.2.4`.

2.2.2 Software synchronisation with git

The version control system git helps to administrate software development projects on distributed systems (or at least by distributed users), avoiding problems of accidentally wiping out each others changes. Also, it is a very practical tool for distributing trees of software to others, who then can make their own changes without running the risk of accidentally overwriting them when a new update is provided. The git command can merge those changes, and alerts you of that happening.

You need to have at least the executable git installed on your system to connect to the GitHub repository. This program comes installed by default on Mac OS X and most Linux and Unix systems.

First you need to 'clone' the code from the GitHub server onto your machine:

```
$ git clone https://github.com/remkos/rads
```

This downloads all the code and puts it into a directory called `rads`. This needs to be done only once.

Later on you can bring the source on your machine up to date by going into the `rads` directory and executing:

```
$ git pull -t origin master
```

although it is much simpler to just use:

```
$ make update
```

2.3 Software configuration

Now we are going to determine where the software executables, library, and data is going to be stored. For this we run the configure in the source directory (`rads-v4.2.4` if you downloaded the tarball, or `rads` if you used git). The program configure will allow you to specify where you want things installed and also determines which Fortran compiler you have and what special options are needed for your platform.

By default, configure will install everything under the directory were it resides itself. It will create directories:

bin for the executables (both binaries and scripts)

include for the Fortran 90 module files to be used with the RADS library

lib for the RADS library

share for the system independent data: the satellite data and configuration files. This one particularly, you might want to put somewhere else, on a dedicated disk, for example.

Normally, you would need to tell configure only where you want to install the aforementioned directories. The rest, like where to find your Fortran compiler and the netCDF library, are things that configure should be able to figure out by itself, using the nf-config script, for example. Therefore, you will only have to specify the root directory for the installation (prefix) and likely the place where you want the RADS altimeter data to reside or where they are already residing (datadir). Run, for example:

```
$ configure --prefix=/usr/local --datadir=/rads/data
```

The first argument to configure specifies that the bin, include and lib directory are to be put under /usr/local. The second argument specifies the directory for the data and configuration file (which could be on a server for more systems to use). Still a directory share is created under /usr/local to contain the manuals.

If, for whatever reason, configure cannot find a Fortran compiler or the netCDF libraries on its own, you need to specify the location of the Fortran compiler and the netCDF library and include files. Here is an example:

```
$ configure FC=/sw/bin/gfortran \
  --with-netcdf-inc=/sw/lib/netcdf-gfortran/include \
  --with-netcdf-lib=/sw/lib:/sw/lib/netcdf-gfortran/lib \
  --prefix=/usr/local --datadir=/rads/data
```

The first argument to configure specifies the location of the fortran compiler, while the second identifies the directory where we can find netcdf.mod. The third argument specifies the two directories that contain the netCDF C library (libnetcdf) and netCDF Fortran library (libnetcdff), separated by a colon. If these two are merged, or in one directory, you can just use one directory name.

The configure program also tests if your Fortran compiler is ready for Fortran 90 and can compile with the netCDF library. If you have problems, you may need to review the options you gave to configure, and make sure that configure picked the same compiler that was used to compile the netCDF library. Finding the nf-config command on your system may be pivotal.

Run `configure --help` to get more info.

2.4 Software compilation

Now that your system is configured, it should be easy to compile the software. Just run in the source directory (where you also ran configure):

```
$ make
```

It will compile and link the programs in the subdirectory src, but not those in devel. The latter are only provided to you to get a feel of how the RADS altimeter database was created. You will not be able to compile or link those programs, as essential routines have been left out.

If you have problems compiling, you may need to tweak one of the makefiles, config.mk. Please let us know about it, so that we can change the configure program accordingly. You can do this at the issue tracker on the RADS GitHub page: https://github.com/remkos/rads/issues.

2.5 Installation

To install the software, configuration file, and manuals in the places discussed in Section 2.3, run the following command in the directory where configure resides:

```
$ make install
```

Now you can continue with the mirroring of the data files.

Chapter 3

RADS data mirroring

RADS now exceeds 400 GBytes of data. It virtually impossible to copy all of it in one go, or copy all of it every time that updates have been made. To facilitate the updating, it is recommended to use the rsync program. This program will determine by it self which files are updated and will update only those. In fact, it will transfer only those parts of the files that are actually changed. This provides a significant speed benefit when, for example, an extra data field is added.

You need to have at least the executable rsync installed on your system to use rsync. In case of Linux machines, simply install the rsync package available on most distributions. The program rsync comes standard with Mac OS X, or can be obtained from http://rsync.samba.org.

The rsync command will download the data from the rsync server at the Delft University of Technology in The Netherlands. This server is setup such that it will allow you to access only the RADS data and software. It will not allow you to log in to the server as a common user. Thus, setting up ssh key pairs is not possible.

Let us start, for example, to synchronise the Jason-2 data. The subdirectory for the Jason-2 data is `j2` (See Table 3.1 for all 2-character abbreviations of the altimeter missions). To get all the Jason-2 data, you will type the following commands (still assuming you have your data in /rads/data):

```
$ cd /rads/data
$ rsync -avz --del radsuser@rads.tudelft.nl::rads/data/j2 .
```

At the beginning rsync will ask you to enter the password for `radsuser`. It will have been provided to you when you registered as a user.

Apart from the satellite specific directories, there is a directory that contains configuration files, that help RADS to read the data files. These files are also installed in the same place, when installing the software, but you can download them from the rsync server as well. Be sure to keep these feels up to date.

```
$ cd /rads/data
$ rsync -avz --del radsuser@rads.tudelft.nl::rads/data/conf .
```

If you are patient, and want to get all of the data at once, you can perform the following commands:

```
$ cd /rads
$ rsync -avz --del radsuser@rads.tudelft.nl::rads/data .
```

Altimeter	Abbr.	Nr	Alternatives	References
GEOS 3	g3	1	ge3 geos-3 geos3	(not included in RADS)
Seasat	ss	2	sea seasat-a	(not included in RADS)
Geosat	gs	3	geo geosat	
ERS-1	e1	4	er1 ers-1 ers1	[*Francis*, 1990; *Francis et al.*, 1991]
TOPEX	tx	5	top topex	[*Fu et al.*, 1994]
Poseidon	pn	6	pos poseidon	
ERS-2	e2	7	er2 ers-2 ers2	[*Francis et al.*, 1995]
GFO	g1	8	gfo gfo-1 gfo1	
Jason-1	j1	9	ja1 jason-1 jason1	[*Ménard et al.*, 2003]
Envisat	n1	10	en1 envisat	
Jason-2	j2	11	ja2 jason-2 jason2	[*Lambin et al.*, 2010]
CryoSat-2	c2	12	cs2 cryosat-2 cryosat2	[*Wingham et al.*, 2006]
SARAL	sa	13	sa srl saral altika	
Jason-3	j3	14	ja3 jason-3 jason3	(limited access in RADS)
HY-2A	2a	15	h2a hy-2a hy2a	(not included in RADS)
Sentinel-3A	3a	16	s3a sentinel-3a sentinel3a sntnl-3a	(in RADS summer 2016)
Sentinel-3B	3b	17	s3b sentinel-3b sentinel3b sntnl-3b	(to be launched end 2017)

Table 3.1 Abbreviation and numbers used for the various altimeter missions.

If, for whatever reason, the mirroring is interrupted, you can simply start it again, and it will continue where it left off. If you have a recent version of the rsync program, we recommend that you use the option `--del` instead of `--delete`, as it speeds up the process significantly.

There are one more directory that may be of interest, but is not essential. The `/rads/tables` directory contains a number of lists: lists of the time intervals of passes and cycles, and lists of the data available for each satellite.

To mirror this directories use rsync:

```
$ cd /rads
$ rsync -avz --del radsuser@rads.tudelft.nl::rads/tables .
```

If you need to use rsync regularly to synchronise the RADS data base and you do not want to enter the password every time, you can set up the environment variable `RSYNC_PASSWD` by one of the following methods (depending on the shell):

```
export RSYNC_PASSWD=radspasswd    # under sh or bash
setenv RSYNC_PASSWD radspasswd    # under csh or tcsh
```

If you are using the bash or sh shell, you can do it all in one line, for example:

```
RSYNC_PASSWD=radspasswd \
   rsync -avz --del radsuser@rads.tudelft.nl::rads/data .
```

Obviously, the password is **not** simply `radspasswd`.

Chapter 4

RADS data management

This Chapter describes the basic functionalities of the data management system of RADS. These functionalities are part of the RADS utilities as well as the RADS subroutine library on which the utilities are based.

4.1 Preparations before using the data base

The use of RADS starts with the definition of the environment variable `RADSDATAROOT`, such that it points to the root of the RADS data base. For example (for bash users):

```
$ export RADSDATAROOT=/rads/data
```

If you already specified this directory in the configure step, then you do not have to set the `RADSDATAROOT` directory.

It is also practical to include /usr/local/bin (or where ever you installed the binaries) in your executable search path, so that existing executables can be used. However, this is not essential, just practical.

4.2 Common functionalities

A subroutine library is created to facilitate the data reading, conversion to SI units, editing and the construction of sea level anomalies. Based on these routines, several programs (utilities) are created to list or manipulate the contents of the data base. Since these programs share the same routines, much of their functionalities are the same, as well as their user interface. In addition, a number of developer tools are available to create the data base. These programs will be of less concern to the users, and are not yet explained in this manual. Their code is only provided for reference, and are not intended to be compiled on your system.

Common to all RADS utilities is the internal data selection, editing, and the ability to construct the sea level anomaly (or any other fields) *on the fly*. The construction of the sea level anomaly includes a number of arithmetic expressions (adding and subtracting), plus applying a number of selection criteria that can flag the sea level anomaly as invalid. Both the ability to do arithmetic (even beyond mere adding and subtracting) and editing are built into the software library, so that they can be shared by the various RADS utilities.

Even when writing their own program, the user does not have to generate code to construct a sea level anomaly or to do editing. At the same time that the sea level anomaly is constructed,

the data is edited based on system-wide, user, or local preferences. These preferences can be specified at several levels, either in XML configuration files, or by command line options. The order in which these preferences are processed is the following:

- System-wide general preferences, found in `$RADSDATAROOT/conf/rads.xml`.
- General user preferences, found in a file `~/.rads/rads.xml`, if available.
- General local preferences, found in a file `rads.xml` in the current working directory.
- Preference files indicated by the `-X` or `--xml` option on the command line.
- Other command line arguments, such as `-L` or `--limits`.

So the command line arguments overrule the local preferences and which overrule the user preferences and eventually the system-wide preferences.

The XML configuration file is "human readable" and is described in the next section.

4.3 RADS4 configuration file

The RADS4 configuration file rads.xml controls:

- Information about the various altimeter missions, such as repeat period, number of passes per cycle, etc.
- Information about all the variables available for each mission, including editing criteria, format for writing to ascii, and the complete information needed to store the variable upon creation of the database.
- Optionally, description of variables that can be created *on the fly* from other variables.

The configuration file is a structured XML file, comprised of a number items and blocks, of the following shape:

```
<block>
    <item option="value">....</item>
    <tag>....</tag>
</block>
```

As usual in an XML file, `<tag>` starts an item with name `tag` and `</tag>` ends it. Any content in between is the value, or are the values associated with this tag. Generally, the line can be broken by carriage returns and leading spaces are ignored.

4.3.1 Tags in the configuration file

The various names of the tags used in the configuration file are described below.

if is followed by a condition such as `sat="j1 j2"`, which means that the contents of the block following this `<if>` applies only to the missions `j1` and `j2`. The selection can be negated by starting it with an exclamation point, i.e. `sat="!j1 j2"` means that the following block applies to all missions *except* `j1` *and* `j2`.

elseif, else can follow an if-block. For example:

```
<if sat="j1 j2">
    ... for j1 and j2 ...
```

```
</if>
<elseif sat="e1 e2">
    ... for e1 and e2 ...
</elseif>
<else>
    ... all other missions ...
</else>
```

global_attributes specifies the global attributes that are always to be written to the output file, line by line. The first word on each line is the name of the global attribute, the rest is its value.

satellites specifies, line by line, the 2- and 3-character abbreviations for the altimeter missions. The rest of each line is used to specify alternative names which can be searched for substrings.

var identifies a block of tags that describe a single variable. The tag needs to include an option such as `name="alt_gdre"` to specify the variable name, and optionally (for example) `field=425` which specify the field numbers used in RADS3. A condition like `sat="j1 j2"` can also be added as an option in this tag.

alias specifies an alias (i.e. an alternative name) for a previously specified variable. The tag should at least include an option such as `name="alt"` to specify the name of the alias, and optionally the field number used in RADS3 (e.g. `field=4`). Again, a condition like `sat="j1"` can be added. If more than one value is given, then it means that the alias will point to the first variable, unless it is not available, then it points to the second. For example:

```
<alias name="alt" field="4" sat="j2">alt_gdre alt_gdrd</alias>
```

means that if one uses the variable `alt` for Jason-2, then the variable `alt_gdre` is processed. If determining this variable was unsuccessful, then `alt_gdrd` is used instead.

4.3.2 Tags within a var-block

The following tags can only occur within a var-block (with one exception, see below):

long_name specifies the description of the variable, as well as the `long_name` attribute to be written to the netCDF data files.

standard_name, source, comment specify the `standard_name`, `source`, and `comment` attributes to be written to the netCDF data files.

units specifies the units of the variable as used in any output, as well as the `units` attribute in the netCDF data files.

flag_values specifies the meanings of a flag, counting up from 0. For example `yes no` means that 0 is the be interpreted as ”yes”, 1 as ”no”.

flag_mask specifies the meanings of a flag word made up of bits. For example `left front up` means that when the LSB is raised ”left” is true, and the next bits indicate if ”front” and ”up” are true.

limits specifies the range of ”good” values. Any value less than the lower limit or greater than the upper limit result in the variable to be set to NaN. In case the variable is a flag_mask, then the first value indicates the mask of bits that should not be set, the second value the bits that should be set, otherwise the result will yield NaN.

plot_range specifies a suggested range for plotting the variable. It is not used for any processing, except for the radsvar program.

parameters is used for the RADS4 database creation and has no impact on any user programs.

data specifies the variable name as used in the input RADS netCDF data files (normally the same as the variable name), but could also be used to specify a mathematical statement that derives the variable from others, or to interpolate values from a grid. More about this in Section 4.3.3.

quality_flag indicates which variables are checked to determine the quality of the current variable. If any of the specified variables lead to NaN (not-a-number), the corresponding value of the current variable is set to NaN as well.

dimension is the dimension of the variable. Default is 1-dimensional. Some variables can be 2-dimensional.

format is the Fortran format specification for ASCII output of the variable.

compress specifies the data type (i.e., `int1`, `int2`, `int4`, `real`, or `dble`) and optional `scale_factor` and `add_offset` to be used in the output netCDF files. This applied only the output files. Input netCDF files may be different.

Because you may want to overrule just one line in a var-block, one can stick the option `var="varname"` in the above tags. For example,

```
<var name="alt_gdre">
    <compress sat="j1 j2 j3 tx">int4 1e-4 1300e3</compress>
</var>
```

is equivalent to

```
<compress var="alt_gdre" sat="j1 j2 j3 tx">int4 1e-4 1300e3</compress>
```

4.3.3 The data-tag

The data-tag can be used in a variety of ways, depending on the text (value) between the `<data>` and `</data>` strings, or the optional `source=`*source* option.

netcdf: The simplest form, explained above, is that the value indicates the name of the netCDF variable in the RADS data files. For example: `<data>alt_gdre</data>`. RADS automatically assumes this type of input if only one value is given. Alternatively use `<data source="nc">` or `<data source="netcdf">`.

It is also possible to read global or variable attributes, which will then be assigned to the variable for all of the given pass. For example, if such attributes exist, `range_ku:add_offset` (the attribute `add_offset` of the variable `range_ku`) or `:range_bias_ku` (the global attribute `range_bias_ku`) could be used to retrieve a possible bias added to the Ku-band range.

math: If more than one value is given (i.e. it contains at least one space), then it will be recognised as a mathematical statement, in "Reverse Polish" notation. A number of mathematical operators are available, as described in Table 4.1. This can be very practical to determine on the fly the difference between two variables, or a combination of many (such as the sea level anomaly). For example `<data>wet_tropo_rad wet_tropo_ecmwf SUB</data>` computes the difference between the radiometer wet tropospheric correction and the

ECWMF wet tropospheric corrections by subtracting the latter from the former. To exercise this mathematical machinery, one can also use `<data source="math">`.

grid: RADS can interpolate in 2-D netCDF grids *on the fly*. This is practical if you have a static model to be interpolated at a certain location (longitude and latitude, or any other two variables). Depending on the argument of the `source=` option, one can select to do bilinear interpolation (`"grid"` or `"grid_l"`), cubic spline interpolation (`"grid_c"` or `"grid_s"`), or nearest neighbour selection (`"grid_q"` or `"grid_n"`). Without any of these options, a file name ending on ".nc" will automatically be recognised as needing linear interpolation. In addition one can specify the variables to be used as x- and y-coordinated in the interpolation. Default is `x="lon" y="lat"`. Example for interpolating an SSB model:

```
<data source="grid" x="wind_speed_alt" y="swh">my_ssb_model.nc</data>
```

constant: To assign a single constant value use, for example,
`<data source="constant">0</data>`.

branch: An additional option allows you to read a variable from a different directory branch. For example, if you have an additional set of data stored in the directory `c2.mydata` next to the normal `c2` branch, then you can add the option `branch=".mydata"` which will indicate that this extension is used to the 2-character mission abbreviation to find files containing the current variable within the var block. Currently up to 5 branches (including the standard one) can be used.

4.3.4 Tags for mission definitions

A number of tags are employed to give general information about the various altimeter missions. You find them at the end of the global configuration file rads.xml. These tags are:

satellite Name of the satellite (maximum 8 characters).

satid Satellite ID used in the crossover program radsxogen.

dt1hz Time step of 1-Hz data.

inclination Orbital inclination in degrees.

frequency Altimeter frequency or frequencies.

xover_params Parameters used in radsxogen.

phase A block of tags that apply to a given mission phase.

mission Name of the mission phase.

cycles First and last cycle number in this phase.

repeat Length of the repeat cycle (not subcycle) in days and passes (half revolutions) in this phase.

ref_pass Time of the equator crossing, longitude of the equator crossing, cycle number (not subcycle number) and pass number of reference pass in this phase.

start_time Start time of the mission phase.

In the above, time epochs can be written as *yyyy-mm-dd*T*hh*:*mm*:*ss*, where the part from 'T' onward can be omitted in whole or in part.

Operator	Description
x y SUB a	a = x - y
x y ADD a	a = x + y
x y MUL a	a = x * y
PI a	a = pi
E a	a = exp(1)
x POP	remove last item from stack
x NEG a	a = −x
x ABS a	a = |x|
x INV a	a = 1/x
x SQRT a	a = sqrt(x)
x SQR a	a = x*x
x EXP a	a = exp(x)
x LOG a	a = ln(x)
x LOG10 a	a = log10(x)
x SIN a	a = sin(x)
x COS a	a = cos(x)
x TAN a	a = tan(x)
x SIND a	a = sin(x) [x in degrees]
x COSD a	a = cos(x) [x in degrees]
x TAND a	a = tan(x) [x in degrees]
x SINH a	a = sinh(x)
x COSH a	a = cosh(x)
x TANH a	a = tanh(x)
x ASIN a	a = arcsin(x)
x ACOS a	a = arccos(x)
x ATAN a	a = arctan(x)
x ASIND a	a = arcsin(x) [a in degrees]
x ACOSD a	a = arccos(x) [a in degrees]
x ATAND a	a = arctan(x) [a in degrees]
x ASINH a	a = arcsinh(x)
x ACOSH a	a = arccosh(x)
x ATANH a	a = arctanh(x)
x ISNAN a	a = 1 if x is NaN; a = 0 otherwise
x ISAN a	a = 0 if x is NaN; a = 1 otherwise
x RINT a	a is nearest integer to x
x NINT a	a is nearest integer to x
x CEIL a	a is nearest integer greater or equal to x
x CEILING a	a is nearest integer greater or equal to x
x FLOOR a	a is nearest integer less or equal to x
x D2R a	convert x from degrees to radian
x R2D a	convert x from radian to degrees
x YMDHMS a	convert seconds of 1985 to format YYYYMMDDHHMMSS
x SUM a	a(i) = x(1) + ... + x(i) while skipping all NaN
x DIF a	a(i) = x(i)-x(i-1); a(1) = NaN
x DUP a b	duplicate the last item on the stack
x y DIV a	a = x / y
x y POW a	a = x^y
x y FMOD a	a = x modulo y
x y MIN a	a = the lesser of x and y
x y MAX a	a = the greater of x and y
x y ATAN2 a	a = arctan(x) taking into account the quadrant as determined by y
x y HYPOT a	a = sqrt(x*x+y*y)
x y R2 a	a = x*x + y*y
x y EQ a	a = 1 if x == y; a = 0 otherwise
x y NE a	a = 0 if x == y; a = 1 otherwise
x y LT a	a = 1 if x < y; a = 0 otherwise
x y LE a	a = 1 if x ≤ y; a = 0 otherwise
x y GT a	a = 1 if x > y; a = 0 otherwise
x y GE a	a = 1 if x ≥ y; a = 0 otherwise
x y NAN a	a = NaN if x == y; a = x otherwise
x y AND a	a = y if x is NaN; a = x otherwise
x y OR a	a = NaN if y is NaN; a = x otherwise
x y IAND a	a = bitwise AND of x and y
x y IOR a	a = bitwise OR of x and y
x y BTEST a	a = 1 if bit y of x is set; a = 0 otherwise
x y AVG a	a = 0.5*(x+y) [when x or y is NaN a returns the other value]
x y DXDY a	a(i) = (x(i+1)-x(i-1))/(y(i+1)-y(i-1)); a(1) = a(n) = NaN
x y EXCH a b	exchange the last two items on the stack (NaNs have no influence)
x y z INRANGE a	a = 1 if x is between y and z; a = 0 otherwise (also in case of any NaN)
x y z BOXCAR a	a = filter x along monotonic dimension y with boxcar of length z (NaNs are skipped)
x y z GAUSS a	a = filter x along monotonic dimension y with Gauss function with sigma z (NaNs are skipped)

Table 4.1 Math operators that can be used in the data-tag. Left of the operator (in uppercase) are the input value(s); on the right of the operator the output value(s).

Chapter 5

RADS utilities

5.1 Command line interface

The RADS4 commands share a very similar layout of the arguments that can be used on the command line. The common features of the command line interface, are explained in this Section. In later Sections we will explain the syntax and use of each of the RADS4 commands.

To learn the syntax of any of the RADS4 commands, simply use the argument --help after the command name. Let us start off with an **excerpt** of the help of the radsstat4 program obtained when typing:

```
$ radsstat4 --help
radsstat4 (v4.2.0): Print RADS statistics per cycle, pass or day(s)

Usage: radsstat4 [required_arguments] [rads_dataselectors] [rads_options] [program_optic

Required argument is:
  -S, --sat SAT[/PHASE]     Specify satellite [and phase] (e.g. e1/g, tx)

Optional [rads_dataselectors] are:
  -C, --cycle C0[,C1[,DC]]  Specify first and last cycle and modulo
  -L, --limits VAR=MIN,MAX  Specify edit data range for variable VAR

Common [rads_options] are:
  -v, --verbose             Increase verbosity level

Program specific [program_options] are:
  -l, --minmax              Output min and max in addition to mean and stddev
  -o, --output [OUTNAME]    Create netCDF output instead of ASCII (default filename is '
```

The command line for every RADS4 program consists of a few of the following components:

program name always starts the command line.

required arguments need to be part of the command line for proper operation; in the above example it specifies the satellite mission.

data selectors are options (in the Unix parlance those parts that start with one or two hyphens) that select the RADS data of interest.

RADS options are options that are common to most programs.

program options are options can be placed on the command line to provide certain optional functionalities specific to the program.

In this sense, RADS is very much in-line with most Unix commands. RADS also supports both so-called short options and long options, which each can have required or optional arguments. This you will also find in many other Unix commands, particularly those of the GNU flavour.

In general the order of the arguments is irrelevant, though there are some exceptions that will be explained per program.

5.1.1 Short options

Short options start with a single hyphen and are followed by a single lower case or upper case letter. In RADS a distinction is made between lower case and upper case options in that the upper case options have a common meaning and syntax between most RADS programs whereas lower case options are specific per program; this is done to harmonise the meanings across programs and help the user to easier remember and recognise the distinctive scope of the options.

Short options can have arguments (a values attached to them), which can either be required or optional. So there are three posibilities:

no argument: The option flag just switches something on or off, for example -l in radsstat4 adds the printing of minima and maxima to the output. In some cases, like -v, the option can be repeated to increase the verbosity level one step each time.

Short options without argument can be combined behind a single hyphen. Thus each of the following are equivalent:

```
-v -l -v
-l -vv
-lvv
-vlv
```

required argument: The option flag assigns some value. For example, -S j2 selects the satellite altimeter mission j2.

The option flag and the value can be separated by whitespace (one or more spaces or tabs), but they can also be pasted together. In addition, the option flag can be preceded by short options without argument under the same hyphen. So again, the following combinations all mean the same thing:

```
-v -S j2
-v -Sj2
-vS j2
-vSj2
```

optional argument: The option flag assigns some value, or switches something on with a default value. In the above example for radsstat4, the option -o can be used with or without an argument, as identified by the argument [outname] between square brackets. As the help explains, when -o is omitted, the output is in ASCII format, when -o is used without argument, then a netCDF file radsstat.nc is created, and when used with an argument (e.g. -o myfile.nc) then a netCDF file with that name is created.

Note that the use of these types of options can be tricky, as any value that does not start with a hyphen that follows can be perceived as an argument of this option, which can lead to some unintended consequences. To avoid this problem, the option list can be ended with

--. All remaining material will then be seen as verbatim arguments (for example input file names), even when they start with a hyphen.

5.1.2 Long options

To make it a bit less hard to understand (or to remember) some of the options, quite often an alternative "long option", starting with two hyphens is provided. As can be seen in the radsstat4 help, the short option -v can also be written as --verbose. But any shorter version of the long option, as long at it is unique, can be used. So --verb is allowed as well.

Just like short options, long options can have no argument, a required argument, or an optional argument. But long options cannot be combined into one, while short options can.

There are a few alternative ways of combining a long option with its required or optional argument.

- One can always use whitespace (spaces or tabs) to separate the long option from its argument.
- If the argument contains an equal sign ('=') then one can also use a colon to separate the long option and its argument.
- If the argument does not contain an equal sign, then an equal sign can be used to separate the long option and the argument.

For example, either of the following syntaxes can be used:

```
--cycle 10,20 --limits sla=-1,1
--cycle=10/20 --limits:sla=-1/1
--cyc 10-20   --lim sla=-1,1
--cyc=10,20   --lim:sla=-1,1
```

5.2 Common options

As indicated above, the RADS4 commands have a lot of options in common. Most of these use short options in upper case. Long options are always lower case.

5.2.1 Common data selectors

Below, in alphabetical order, are the command line options used to select data from the RADS database.

-A *VAR1=VAR2* **--alias** *VAR1=VAR2*
Assign *VAR1* as an alias of *VAR2*. This means that when *VAR1* is requested, *VAR2* will be used. For example --alias wet_tropo=wet_tropo_ecmwf makes wet_tropo (which is used in the construction of the sea level anomaly sla) an alias of wet_tropo_ecmwf. Hence the model wet tropospheric correction, and not the radiometer wet tropospheric correction will then be used to determine sea level.

-C *C0[,C1[,DC]]* **--cycle** *C0[,C1[,DC]]*
Select data from cycle *C0* to cycle *C1*. If the latter is omitted, only data from cycle *C0* is selected. Adding *DC* will run through the cycles with a step of *DC*, skipping the intermediate ones. The arguments *C0*, *C1* and *DC* can be separated by commas, slashes, or the minus

sign. So -C 1/3 selects cycles 1 through 3. If omitted, data from all cycles (throughout all mission phases) are selected.

-F ***VAR=FMT*** **--fmt** ***VAR=FMT*** **--format** ***VAR=FMT***
For variable *VAR* use the Fortran format *FMT* in ASCII output. E.g., -F sla=f8.4. This can also be achieved with the ctagformat tag in a RADS4 configuration file.

-L ***VAR=MIN,MAX*** **--limits** ***VAR=MIN,MAX***
Specify the valid range for the selection of a variable. Outside this range the variable will be set to NaN. This is an alternative to using the limits tag in a RADS4 configuration file.

--lat ***LAT0,LAT1***
Select data in the latitude range from *LAT0* to *LAT1* in degrees. This is equivalent to using --limits lat=*LAT0,LAT1*. The south to north range has to be specified on the interval from –90° to 90°. Examples:

--lat -40,-20 Select data between 40°S and 20°S
--lat 0,90 Selects entire northern hemisphere

When omitted, the default range specified in the RADS4 configuration file is used. By default, this is –90° to +90°.

--lon ***LON0,LON1***
Select data in the longitude range from *LON0* to *LON1* in degrees. This is the same as --limits lon=*LON0,LON1*. Note that the output longitudes are influenced by the selection the interval, be it in the range from –180° to 180° or from 0° to 360°. Examples:

--lon -40,-20 Select data between 40°W and 20°W
--lon 320,340 Same, but output will have only positive longitudes

When omitted, the default range specified in the RADS4 configuration file is used, which is normally –180° to 180°.

-P ***P0[,P1[,DP]]*** **--pass** ***P0[,P1[,DP]]***
Select data from pass *P0* to pass *P1* (out of each selected cycle). If the latter is omitted, only data from pass *P0* is selected. Adding *DP* will run through the passes with a step of *DP*, skipping the intermediate ones. The arguments *P0*, *P1* and *DP* can be separated by commas, slashes, or the minus sign. So -P 1-100 selects passes 1 through 100. If omitted, data from all passes in each of the select cycles is selected.

In addition, one can use -P a (or --pass asc) to select only ascending passes, whereas -P d (or --pass des) is used to select only descending passes.

-Q ***VAR=FLAG*** **--quality-flag** ***VAR=FLAG***
Check the variable *FLAG* when validating the variable *VAR*. This means that in any event that the variable *FLAG* is set to NaN (is out of bounds), then *VAR* is also set to NaN as well. This is identical to using the quality_flag tag in a RADS4 configuration file.

-R ***LON0,LON1,LAT0,LAT1*** **--region** ***LON0,LON1,LAT0,LAT1***
Select data within the longitude range from *LON0* to *LON1* and the latitude range of *LAT0* to *LAT1*, both in degrees. This is equivalent to using both the --lon and --lat options.

-R ***LON0,LAT0,RADIUS*** **--region** ***LON0,LAT0,RADIUS***
Select data within a circular region with a spherical radius *RADIUS* degrees, around the point *LON0,LAT0* (in degrees).

-S ***SAT[/PHASE]*** **--sat** ***SAT[/PHASE]***
Select data from a given satellite *SAT*. Many variants of the satellite name can be used. Either the 2-character code, the 3-character code, or a substring of the satellite name listed in Table 3.1.

This is a required option and needs to be given for all RADS4 programs. Optionally, one can also indicate the mission phase *PHASE*, separated from the satellite by a slash. (For backwards compatibility with RADS3 also a colon can be used, but this is discouraged).

--sla ***SLA0,SLA1***
Specify the range of for sea level anomaly in meters (if requested). This is the same as --limits sla=*SLA0,SLA1*.

--time ***T0,T1*** **--ymd** ***T0,T1*** **--doy** ***T0,T1*** **--sec** ***T0,T1***
Select data only in the time interval from *T0* to *T1*. RADS4 is generally quite efficient in figuring out the time format when using the generic --time, but best is to use instead --ymd with time in the format *[YY]YYMMDD[HHMMSS]*, --doy for *[YY]YYDDD* (year and day of year), or --sec for time in seconds from 1 Jan 1985. Decimal points to indicate fractional days or seconds are also allowed.

The following examples all select data between 1 January 1999, 12:00 UTC and 31 December 2001, 00:00 UTC:

--ymd 990101.5,20011231	Note that the century can be omitted
--doy 99001.5,2001365	when using --ymd or --doy.
--mjd 51179.5,52274.0	With --mjd or --doy, use only fractions of day,
--ymd 990101120000,011231	but with --ymd one can also use "HHMMSS".
--sec 441806400,536371200	The old-fashioned seconds since 1985 are still there as well.

One will notice that the data selection is very fast. RADS4 will not go through all the data sets to get the right data, but will use its "knowledge" of the repeat cycles and passes to quickly select the appropriate cycles and passes and scan only those.

-V ***VAR1,...*** **--var** ***VAR1,...***
Select one or more variables from the RADS database. Any number of variables can be select in one go by adding their names to the list, separated by commas or slashes. Also the old RADS3 numerical field numbers for the variables can be used!

-X ***XMLFILE*** **--xml** ***XMLFILE***
Load the configuration file *XMLFILE* in addition to the default configuration files ($RADSDATAROOT/conf/rads.xml, and—if available—$HOME/.rads/rads.xml and rads.xml in the current working directory).

-Z ***VAR=TYPE[,SCALE[,OFFSET]]*** **--compress** ***VAR=TYPE[,SCALE[,OFFSET]]***
Specify how the variable *VAR* is to be stored in a netCDF output file. *TYPE* is one of: int1, int2, int4, real, dble. *SCALE* and *OFFSET* are optional with a default of 1 and 0. This is identical to using the compress tag in a RADS4 configuration file. One can also use --cmp.

In all lists, shown as separated by commas above, one can also use slashes, as is done in the GMT program syntax.

5.2.2 Backward compatibility

For backward compatibility with RADS3 a number of options are still allowed, even though their use is deprecated.

-h ***H0,H1***
Specify the range from SLA (in meters). This has been replaced by --sla *H0,H1*.

--opt ***J***
Use field number *J* when field *J*/100 is requested. This is now achieved with --alias *VAR1=VAR2*.

--opt *I=J*
Make field number *I* (in the range 1 to 99) and alias of field number *J* (in the range 101-9999). This is now achieved with --alias *VAR1=VAR2*.

--sel *VAR1,...*
Select variables to be read. Same as --var *VAR1,...*, except that using --sel signals some additional backward compatible options.

In addition, again support backward compatibility with RADS3, the user can omit the double hyphen in front of the RADS3 data selectors like sel=, sat=, cycle=, pass=, etc. But it is highly recommended to use the new options instead.

5.2.3 Other common options

Additional common options in RADS4 are:

--args *FILENAME*
Get any of the command line arguments from a file named *FILENAME* with one option per line.

--debug *LEVEL*
Set debug/verbosity level to *LEVEL*.

--help
Print the syntax of the command and all its arguments.

--log *FILENAME*
Write statistics and other log information to file *FILENAME* (default is standard output).

-q --quiet
Suppress warning messages (but keeps fatal error messages).

-v --verbose
Increase the verbosity level (warnings and debugging). This option can be used numerous times, increasing verbosity by one level each time.

--version
Print the program version only.

--
Terminates all command line options; all following command-line arguments are considered non-option arguments, even if they begin with a hyphen.

5.3 Order of the options

When processing the data of a single satellite the order of the command line options is irrelevant, with the exception that options that appear later (further to the right) on the command line overrules any similar options that appear earlier. For example -q -v will produce verbose output, as -v overrules the earlier -q.

Yet, there is a more complicated order in which the command line options are processed by the RADS library:

1. Process some input/output options: --args, --debug, --help, --log, -q, --quiet, -v, --verbose, --version.
2. Process the satellite mission selector -S or --sat.

3. Process possible additional XML files indicated by the -X or --xml options.
4. Process any alias definitions specified by the -A or --alias options.
5. Process miscellaneous common options not mentioned elsewhere in this list.
6. Process the variable selection defined by the -V or --var options.
7. Process program-specific options.

Each of these groups of options are processed left to right. This order is common to all RADS programs discussed later in this Chapter, as they all use the RADS library to scan the command line.

When processing multiple satellites at the time (for example when creating dual-satellite crossovers with radsxogen), it matters whether you put common options before or after mission selector (-S or --sat). The order then determines whether the option is unique to a single mission, or applies to all. The rule is as follows:

- Common options ahead of the first mission selector apply to all missions.
- Common options after the first mission selector apply to the last mission selector to the right of the option.
- Program-specific options can be put anywhere.

This means, for example:

```
$ radsxogen [common options] -S e1 [e1 options] -S e2 [e2 options]
```

5.4 rads2asc4

The program rads2asc4 lists a selection of the RADS4 data base in one ASCII file or in several pass-by-pass ASCII files. These files contain a header per pass with a description of the data content followed by one record for each measurement that passes the selection criteria. The records are build up of columns, separated by white space, listing various data fields. Which data fields are listed (and in which order they are listed) is determined by the command line argument -V or --var.

In the output, the data that has not passed the editing criteria will be represented by "NaN" (Not-a-Number). If the sea level anomaly field suffers that fate, the record will not be listed in the output file, unless the -r option is used (or -r 0 or -r none).

If the argument -o *OUTNAME* or --output *OUTNAME* is used, the output will go into one file (named OUTNAME). Otherwise, pass files are created with the names `SSpPPPPcCCC.asc`, similar to what is described in Chapter 1.

This program can be called as rads2asc, or as rads2asc4 to avoid conflicts with an already installed RADS3.

For more details, see the description of the command line options in the next Sections.

5.4.1 Syntax

```
rads2asc4 (v4.2.4-2-g55d2fc7): Select RADS altimeter data and output to ASCII

Usage: rads2asc4 [required_arguments] [rads_dataselectors] [rads_options] [program_options]

Required argument is:
```

```
  -S, --sat SAT[/PHASE]     Specify satellite [and phase] (e.g. e1/g, tx)

Optional [rads_dataselectors] are:
  -A, --alias VAR1=VAR2     Use variable VAR2 when VAR1 is requested
  -C, --cycle C0[,C1[,DC]]  Specify first and last cycle and modulo
  -F, --fmt, --format VAR=FMT
                            Specify Fortran format used to print VAR (for ASCII output only)
  -L, --limits VAR=MIN,MAX  Specify edit data range for variable VAR
  --lon LON0,LON1           Specify longitude boundaries (deg)
  --lat LAT0,LAT1           Specify latitude  boundaries (deg)
  -P, --pass P0[,P1[,DP]]   Specify first and last pass and modulo; alternatively use -Pa
                            (--pass asc) or -Pd (--pass des) to restrict selection to
                            ascending or descending passes only
  -Q, --quality-flag VAR=FLAG
                            Check variable FLAG when validating variable VAR
  -R, --region LON0,LON1,LAT0,LAT1
                            Specify rectangular region (deg)
  -R, --region LON0,LAT0,RADIUS
                            Specify circular region (deg)
  --sla SLA0,SLA1           Specify range for SLA (m)
  --time T0,T1              Specify time selection (optionally use --ymd, --doy,
                            or --sec for [YY]YYMMDD[HHMMSS], [YY]YYDDD, or SEC85)
  -V, --var VAR1,...        Select variables to be read
  -X, --xml XMLFILE         Load configuration file XMLFILE in addition to the defaults
  -Z, --cmp, --compress VAR=TYPE[,SCALE[,OFFSET]]
                            Specify binary output format for variable VAR (netCDF only); TYPE
                            is one of: int1, int2, int4, real, dble; SCALE and OFFSET are
                            optional (def: 1,0)

Still working for backward compatibility with RADS3 are options:
  --h H0,H1                 Specify range for SLA (m) (now --sla H0,H1)
  --opt J                   Use field number J when J/100 requested (now -A VAR1=VAR2)
  --opt I=J                 Make field I (range 1-99) and alias for field J (now -A VAR1=VAR2)
  --sel VAR1,...            Select variables to read

Common [rads_options] are:
  --args FILENAME           Get all command line arguments from FILENAME (1 argument per line)
  --debug LEVEL             Set debug/verbosity level
  --help                    Print this syntax massage
  --log FILENAME            Send statistics to FILENAME (default is standard output)
  -q, --quiet               Suppress warning messages (but keeps fatal error messages)
  -v, --verbose             Increase verbosity level
  --version                 Print version info only
  --                        Terminates all options; all following command-line arguments are
                            considered non-option arguments, even if they begin with a hyphen

Program specific [program_options] are:
  -r, --reject-on-nan VAR   Reject records if variable VAR on -V specifier is NaN
  -r NR                     Reject records if data item number NR on -V specifier is NaN
  -r 0, -r none, -r         Do not reject records with NaN values
  -r n, -r any              Reject records if any value is NaN
                      Note: If no -r option is given -r sla is assumed
  -f                        Do not start with time,lat,lon in output (only with --sel)
  -s, --stat a|c|p          Include statistics for all data (a), per cycle (c), or per pass (p)
  --step N                  Step through records with stride N (default = 1)
  --list FILENAME           Specify file name in which to write list of output files
  --maxrec NREC             Specify maximum number of output records (default = unlimited)
  -o, --output OUTNAME      Specify name of a single output file or - for standard output or
                            (when ending in /) directory name for pass files; by default pass
                            files are created in the current directory
```

5.4.2 Common options

For a full explanation of the command line options that are common to most RADS4 programs, see Section 5.2.

5.4.3 Program specific options

The command line options that are specific to rads2asc4 are listed here.

-r *VAR* **--reject-on-nan** *VAR*

Do not output records when the value of variable *VAR*, given as an argument to the -V

or --VAR option, is set to NaN. By default records for which the sea level anomaly (if requested) is set to NaN are rejected.

-r *NR* **--reject-on-nan** *NR*
For backward compatibility with RADS3, this will eliminate records when the value on of item number *NR* on the -V option is NaN.

-r 0 -r none -r
When either of these is used, no records are eliminated and all NaN values are kept.

-r n -r any
Indicates that a line should not be printed when a value in any of the data columns is NaN.

-f This only counts when using --sel instead of --var, in order to be backward compatible with RADS3. When using--sel normally time, latitude and longitude are automatically added is output columns. With -f those columns are not automatically added, i.e., you will have to add them to the list of variables.

-s c|p
Include statistics per cycle (c) or pass (p). Those will by included in the output with lines starting with '`# min :`', '`# max :`', '`# mean:`', and '`# std :`' followed by the cycle number, the number of valid passes in the cycle (c) or the pass number itself (p), the number of valid records in that cycle (c) or pass (p), and then the statistics (minimum, maximum, mean, and standard deviation) of each column.

--step *N*
Print only one out of *N* records.

--list *FILENAME*
Specify the name of a file to which to write a list of the created output files.

--maxrec *NREC*
Specify the maximum number of output records. The default is unlimited.

-o *OUTNAME* **--output** *OUTNAME*
By default (without this option) files are created per pass with a name similar to the ones in the data base, i.e., `SSpPPPPcCCC.asc`, where `SS` is the satellite abbreviation, `PPPP` is the pass number, and `CCC` is the cycle number. This option will send the output to a single named file, or to standard output when using `-o -`. Alternatively, when *OUTNAME* ends in a slash, then it indicates the directory in which to store the pass files `SSpPPPPcCCC.asc`.

5.4.4 Example

Assuming a default configuration, we issue the following command:

```
$ rads2asc4 -S e2 -C 0 -P 901,1000,2 -R -8,42,28,48 \
    -V time_ymdhms,lat,lon,sla,swh,wind_speed -o rads2asc4.asc -v
```

The program will print to standard output information on which passes are available and have valid data points in the requested area (Mediterranean Sea), how many records were read and how many remained in the particular area, how many records were rejected based on the selection criteria on the corrections, and finally some statistics on the columns that were requested.

```
********************************************************************************
Data selection for satellite ERS-2 phase a
x = pass has no data in period and area
```

```
- = pass file does not exist
o = pass file has no valid data
# = pass file has valid data

Cycle Pass  ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...
    0  901  - - - - - - o # # - - - - - - - - - - - - - # # - - - - - - - - - - - - - # # - - - - - - - - - - - -
################################################################################################################
# Editing statistics for ERS-2 (e2)
# Created: 2016-06-14 20:12:02 UTC: rads2asc4 -S e2 -C 0 -P 901,1000,2 -V time_ymdhms,lat,lon,sla,swh,wind_spe
#
# PASSES QUERIED                                          50
#
#                                        REJECTED  SELECTED       LOWER       UPPER        STEP
# Cycle number limits                           0        50           0           0           1
# Pass number limits                            0        50         901        1000           2
# Time limits                                   0        50         NaN         NaN
# Equator longitude limits (asc)               42         8      -3.440      57.004
# Equator longitude limits (des)                0         0     -23.004      37.440
#
# PASSES AND MEASUREMENTS READ                  8     11517
#
#                                        REJECTED  SELECTED       LOWER       UPPER         MIN         MAX
# time [seconds since 1985-01-01 00:00:00          0     11517         NaN         NaN 950512/1843 950515/1932
# latitude [degrees_north]                     10533       984      28.000      48.000      30.185      47.955
# longitude [degrees_east]                      5435      6082      -8.000      42.000      -7.996      41.989
#
# MEASUREMENTS IN REQUESTED PERIOD AND REGION             851
#
#                                        REJECTED  SELECTED       LOWER       UPPER         MIN         MAX
# sea level anomaly [m]                         293       558      -3.000       3.000      -0.251       0.111
# time [yymmddhhmmss]                             0       851         NaN         NaN************************
# REAPER/COMBI orbital altitude [m]               0       851         NaN         NaN  787721.981  791023.401
# Ku-band range corrected for instr. effec        0       851  750000.000  850000.000  787647.826  791005.014
# ECMWF dry tropospheric correction [m]           0       851      -2.400      -2.100      -2.331      -2.266
# radiometer wet tropospheric correction [       11       840      -0.600       0.000      -0.598      -0.031
# NIC09 ionospheric correction [m]                0       851      -0.400       0.040      -0.025      -0.018
# MOG2D dynamic atmospheric correction [m]        0       851      -1.000       1.000      -0.144       0.195
# solid earth tide [m]                            0       851      -1.000       1.000      -0.108       0.084
# GOT4.10 ocean tide [m]                         47       804      -5.000       5.000      -1.749       0.775
# GOT4.10 load tide [m]                           0       851      -0.500       0.500      -0.019       0.045
# pole tide [m]                                   0       851      -0.100       0.100      -0.010       0.003
# BM3 sea state bias [m]                          6       845      -1.000       1.000      -0.857       0.471
# DTU13 mean sea surface height [m]               0       851         NaN         NaN       3.597      53.232
# Ku-band significant wave height [m]            16       835       0.000       8.000       0.000       7.960
# Ku-band backscatter coefficient [dB]            8       843       6.000      27.000       6.460      26.410
# altimeter wind speed [m/s]                      2       849      -1.000      30.000       0.000      24.944
# std dev of Ku-band range [m]                   66       785       0.000       0.400       0.022       0.398
# number of valid Ku-band measurements [1]       64       787      16.500      20.500      17.000      20.000
# flag word [count]                             286       565   65512.000       0.000       0.000       0.000
# std dev of Ku-band significant wave heig       63       788       0.000       2.100       0.290       2.100
# off-nadir angle squared from waveform (s        0       851      -0.050       0.050       0.000       0.000
# reference frame offset [m]                      0       851         NaN         NaN       0.055       0.057
################################################################################################################
```

The resulting ASCII output file rads2asc4.asc will look like this:

```
# RADS_ASC
# Created: 2016-06-14 20:12:02 UTC: rads2asc4 -S e2 -C 0 -P 901,1000,2 -V time_ymdhms,lat,lon,sla,swh,wind_spe
# Satellite = ERS-2
# Phase     = a
# Cycle     = 000
# Pass      = 0915
# Equ_time  =  326925796.651424 (1995-05-12 20:43:16.651424)
# Equ_lon   =   26.724487
# Original  = OPR V6 (0603) data of 1997-175T18:26:46 (ver.1)
# Col  1    = time [yymmddhhmmss]
# Col  2    = latitude [degrees_north]
# Col  3    = longitude [degrees_east]
# Col  4    = sea level anomaly [m]
# Col  5    = Ku-band significant wave height [m]
# Col  6    = altimeter wind speed [m/s]
950512205154.226807  30.661485   19.480321  -0.1002  0.120  1.737
950512205155.207275  30.719321   19.464490  -0.0560  0.000  1.815
950512205156.187500  30.777142   19.448650  -0.1571  0.330  1.616
950512205157.167969  30.834975   19.432791  -0.1572  0.560  1.858
.............  lines removed ..............................
950515192036.179321  43.127032   39.342061  -0.0023  1.260  4.894
```

```
950515192037.159790  43.184400   39.321693  -0.0053  1.140  4.576
950515192038.140259  43.241765   39.301295   0.0266  0.930  5.521
950515192039.120483  43.299112   39.280871   0.0177  0.870  5.821
```

5.5 rads2grd4

The program rads2grd4 is a quick-and-dirty gridding program for RADS data. No smoothing or interpolation is performed. The data are simply collected in cells of predefined size, after which the mean and standard deviation in each cell is computed.

The boundaries of the grid are specified by the limits set in the RADS4 configuration files, or by the appropriate common command line arguments --limits, loptlat, or --lon. The cell size is determined by the option --res. Alternatively, the options --x and --y can be used to set both the range and interval along both coordinate axis of the grid. The use of grid-node oriented or cell oriented boundaries is controlled by the --c option. The required minimum number of measurements per cell can be specified with the --min option. Cells with a number of points less than the required number will not be part of the output.

The grid program not only grid sea level anomaly in longitude-latitude space, but any data field in any other space specified by the -V (or --var) option. For example, a non-parametric sea state bias model can be created by gridding the sea level anomalies in wind-wave space, in which case -wind_speed,swh,sla is used.

Multiple variables can be gridded simultaneously on the same space. When gridding several variables at once, only records (measurements) for which every variable is valid (not NaN) will be included in the gridding process.

There are two ways to produce output:

ASCII: Without the -o or --output option the program will write to standard output a file with an ASCII header explaining the content, followed by one record for each grid cell with a valid solution. Those records contain the x- and y-coordinate of the centre of the cell, and a pair of mean and standard deviation for each z-variable. The right-most column contains the number of points in the cell. This output can be used directly in GMT's xyz2grd.

netCDF: With the -o or --output option a netCDF grid file is produced. This contains a grid for the mean and standard deviation of each variable as well as a grid with the number of points.

The output is a list of ASCII records per cell: x and y of the centre of the cell, mean of z, standard deviation of z, number of points. Cells with a number of points less than the required number are not listed. This output can be used directly in GMT.

This program can be called as rads2grd, or as rads2grd4 to avoid conflicts with an already installed RADS3.

5.5.1 Syntax

```
rads2grd4 (v4.2.4-2-g55d2fc7): Quickly grid RADS data to xyz or netCDF grid

Usage: rads2grd4 [required_arguments] [rads_dataselectors] [rads_options] [program_options]

Required argument is:
  -S, --sat SAT[/PHASE]     Specify satellite [and phase] (e.g. e1/g, tx)

Optional [rads_dataselectors] are:
  -A, --alias VAR1=VAR2     Use variable VAR2 when VAR1 is requested
```

```
  -C, --cycle C0[,C1[,DC]]  Specify first and last cycle and modulo
  -F, --fmt, --format VAR=FMT
                            Specify Fortran format used to print VAR (for ASCII output only)
  -L, --limits VAR=MIN,MAX  Specify edit data range for variable VAR
  --lon LON0,LON1           Specify longitude boundaries (deg)
  --lat LAT0,LAT1           Specify latitude  boundaries (deg)
  -P, --pass P0[,P1[,DP]]   Specify first and last pass and modulo; alternatively use -Pa
                            (--pass asc) or -Pd (--pass des) to restrict selection to
                            ascending or descending passes only
  -Q, --quality-flag VAR=FLAG
                            Check variable FLAG when validating variable VAR
  -R, --region LON0,LON1,LAT0,LAT1
                            Specify rectangular region (deg)
  -R, --region LON0,LAT0,RADIUS
                            Specify circular region (deg)
  --sla SLA0,SLA1           Specify range for SLA (m)
  --time T0,T1              Specify time selection (optionally use --ymd, --doy,
                            or --sec for [YY]YYMMDD[HHMMSS], [YY]YYDDD, or SEC85)
  -V, --var VAR1,...        Select variables to be read
  -X, --xml XMLFILE         Load configuration file XMLFILE in addition to the defaults
  -Z, --cmp, --compress VAR=TYPE[,SCALE[,OFFSET]]
                            Specify binary output format for variable VAR (netCDF only); TYPE
                            is one of: int1, int2, int4, real, dble; SCALE and OFFSET are
                            optional (def: 1,0)

Still working for backward compatibility with RADS3 are options:
  --h H0,H1                 Specify range for SLA (m) (now --sla H0,H1)
  --opt J                   Use field number J when J/100 requested (now -A VAR1=VAR2)
  --opt I=J                 Make field I (range 1-99) and alias for field J (now -A VAR1=VAR2)
  --sel VAR1,...            Select variables to read

Common [rads_options] are:
  --args FILENAME           Get all command line arguments from FILENAME (1 argument per line)
  --debug LEVEL             Set debug/verbosity level
  --help                    Print this syntax massage
  --log FILENAME            Send statistics to FILENAME (default is standard output)
  -q, --quiet               Suppress warning messages (but keeps fatal error messages)
  -v, --verbose             Increase verbosity level
  --version                 Print version info only
  --                        Terminates all options; all following command-line arguments are
                            considered non-option arguments, even if they begin with a hyphen

Program specific [program_options] are:
  -V, --var X,Y,Z[,...]     Variables for x, y and z (default: lon,lat,sla); multiple z's may be specified
  --x X0,X1[,DX]            Set x-range and interval (default: as set by default limits and --res)
  --y Y0,Y1[,DY]            Set y-range and interval (default: as set by default limits and --res)
  --res DX[,DY]             Set resolution in x and y (default: 1)
  --min MINNR               Minimum number of points per grid cell (default: 2)
  -o, --output, --grd GRIDNAME
                            Create netCDF grid (suppresses ASCII)
  --line-format FORMAT      Format to be used for ASCII output (default is determined by variables)
  -c                        Boundaries are cell oriented
  -c[x|y]                   Only [x|y]-boundaries are cell oriented
```

5.5.2 Common options

For a full explanation of the command line options that are common to most RADS4 programs, see Section 5.2. Note that the -V and --var options have a slightly different meaning in this program (see below) than in most others.

5.5.3 Program specific options

The command line options that are specific to rads2grd4 are listed here.

-V *X,Y,Z[,...]* **--var** *X,Y,Z[,...]*
Specify the variables for the ordinates X and Y, and the variable to be gridded (Z). One can specify one or more variables for Z; each will be part of the output.

--x *X0,X1[,DX]*

Set the x-range and optional interval. The default range is determined by the limits set in the RADS4 configuration file. The interval can also be set with the --res option.

--x *X0,X1[,DX]*
Set the y-range and optional interval. The default range is determined by the limits set in the RADS4 configuration file. The interval can also be set with the --res option.

--res *DX[,DY]*
Set the resolution of the x- and y-coordinate. If only one number is given the same number is used for both coordinates. The default resolution is 1×1.

--min *MINNR*
Specify the minimum number of points in the grid cell. The default is 2.

-o *GRIDNAME* **--output** *GRIDNAME*
Create a netCDF file with the name *GRIDNAME* containing the output grids. This suppresses the ASCII output. For backward compatibility with RADS3, one can also use --grd.

--line-format *FORMAT*
Format to be used for ASCII output. The default is determined by the specific Fortran formats for each variable specified in the RADS4 configuration file.

-c [x|y] Determines that the boundaries of the x-y space are cell-oriented, instead of grid-oriented. When using -c x this applies only to the x-coordinate, when using -c y it applied only to the y-coordinate. When using -c without argument, it applies to both coordinates.

5.5.4 Examples

Assuming a default configuration, we issue the following command:

```
$ rads2grd4 -S e2 -C 0 -P 901,1000,2 -R -8,42,28,48 -V wind_speed,swh,sla -c
```

This command creates to following results to print to standard output:

```
# Grid of RADS variables
# Created: 2016-06-14 20:12:02 UTC: rads2grd4 -S e2 -C 0 -P 901,1000,2 -V wind_speed,swh,sla -c
#
# Satellite : e2/a
# Cycles    :    0 -    0
# Passes    :  901 - 1000
#
# Output columns per grid cell:
#    ( 1) altimeter wind speed [m/s]
#    ( 2) Ku-band significant wave height [m]
# ( 3- 4) mean and stddev of sea level anomaly [m]
#    ( 5) nr of measurements
 0.500  0.500  -0.0101   0.1095 463
 1.500  0.500  -0.0288   0.1076 766
 2.500  0.500  -0.0389   0.1072 938
 3.500  0.500  -0.0369   0.0990 925
.....  lines removed ..........................
15.500  7.500  -0.0178   0.1117 24
16.500  7.500  -0.0242   0.1000 47
17.500  7.500  -0.0010   0.0967 73
18.500  7.500  -0.0393   0.0818 42
```

When we want to create a netCDF file as output, we use the -o option:

```
$ rads2grd4 -S e2 -C 0 -P 901,1000,2 -R -8,42,28,48 -V wind_speed,swh,sla -c \\
    -o rads2grd4.nc
```

To list its contents, use the ncdump command. The result will look like the following:

```
netcdf rads2grd4-o {
dimensions:
```

```
    wind_speed = 31 ;
    swh = 8 ;
variables:
    double wind_speed(wind_speed) ;
        wind_speed:long_name = "altimeter wind speed" ;
        wind_speed:units = "m/s" ;
        wind_speed:actual_range = -1., 30. ;
    double swh(swh) ;
        swh:long_name = "Ku-band significant wave height" ;
        swh:units = "m" ;
        swh:actual_range = 0., 8. ;
    float sla_mean(swh, wind_speed) ;
        sla_mean:long_name = "mean of sea level anomaly" ;
        sla_mean:units = "m" ;
        sla_mean:_FillValue = NaNf ;
    float sla_stddev(swh, wind_speed) ;
        sla_stddev:long_name = "std dev of sea level anomaly" ;
        sla_stddev:units = "m" ;
        sla_stddev:_FillValue = NaNf ;
    int nr(swh, wind_speed) ;
        nr:long_name = "number of points per cell" ;

// global attributes:
        :Conventions = "CF-1.5" ;
        :title = "rads2grd4-o.out" ;
        :history = "2016-06-14 20:12:03 UTC: rads2grd4 -S e2 -C 0 -P 901,1000,2 -V wind_speed,swh,sla -c -o ra
        :node_offset = 1 ;
data:

 wind_speed = -0.5, 0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5, 10.5,
    11.5, 12.5, 13.5, 14.5, 15.5, 16.5, 17.5, 18.5, 19.5, 20.5, 21.5, 22.5,
    23.5, 24.5, 25.5, 26.5, 27.5, 28.5, 29.5 ;

 swh = 0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5 ;

 sla_mean =
  _, -0.0100662, -0.02879892, -0.0388553, -0.03685031, -0.03365956,
    -0.03613032, -0.04943807, -0.04690964, -0.04301253, -0.09423293,
    0.1340371, 0.1991087, 0.3417749, 0.5385897, 0.4476786, 0.5372286,
    0.5057077, -0.088148, 0.465646, _, _, _, _, _, _, _, _, _, _, _,
.............  lines removed ..............................
  0, 0, 0, 0, 0, 0, 4, 9, 4, 24, 22, 16, 22, 53, 75, 112, 75, 143, 100, 38,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 1, 2, 9, 17, 20, 28, 24, 47, 73, 42, 1, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0 ;
}
```

5.6 rads2nc

The RADS4 program rads2nc works similar to rads2asc4 (Section 5.4). Instead of writing output to one of more ASCII files, it creates one or more netCDF files. This significantly reduces on the output size, and can be convenient for reading into third-party programs.

By default, rads2nc will create pass-by-pass netCDF files, named `SSpPPPPcCCC.nc`, with only the information of the netCDF variables given on the command line using the -V or --var options. The variables are edited given the selection criteria in the RADS4 configurations files or elsewhere on the command line. Values regarded 'invalid' will be set to the `_FillValue` in the netCDF files (or NaN in case of floating numbers). They will show up as an underscore (_) in the output of ncdump.

The output can also be captured in a single netCDF file or in a different directory than the current working directory. See the description of the option -o or --output below.

For more details, see the description of the command line options in the next Sections.

5.6.1 Syntax

```
rads2nc (v4.2.4-2-g55d2fc7): Select RADS altimeter data and output to netCDF

Usage: rads2nc [required_arguments] [rads_dataselectors] [rads_options] [program_options]

Required argument is:
  -S, --sat SAT[/PHASE]     Specify satellite [and phase] (e.g. e1/g, tx)

Optional [rads_dataselectors] are:
  -A, --alias VAR1=VAR2     Use variable VAR2 when VAR1 is requested
  -C, --cycle C0[,C1[,DC]]  Specify first and last cycle and modulo
  -F, --fmt, --format VAR=FMT
                            Specify Fortran format used to print VAR (for ASCII output only)
  -L, --limits VAR=MIN,MAX  Specify edit data range for variable VAR
  --lon LON0,LON1           Specify longitude boundaries (deg)
  --lat LAT0,LAT1           Specify latitude  boundaries (deg)
  -P, --pass P0[,P1[,DP]]   Specify first and last pass and modulo; alternatively use -Pa
                            (--pass asc) or -Pd (--pass des) to restrict selection to
                            ascending or descending passes only
  -Q, --quality-flag VAR=FLAG
                            Check variable FLAG when validating variable VAR
  -R, --region LON0,LON1,LAT0,LAT1
                            Specify rectangular region (deg)
  -R, --region LON0,LAT0,RADIUS
                            Specify circular region (deg)
  --sla SLA0,SLA1           Specify range for SLA (m)
  --time T0,T1              Specify time selection (optionally use --ymd, --doy,
                            or --sec for [YY]YYMMDD[HHMMSS], [YY]YYDDD, or SEC85)
  -V, --var VAR1,...        Select variables to be read
  -X, --xml XMLFILE         Load configuration file XMLFILE in addition to the defaults
  -Z, --cmp, --compress VAR=TYPE[,SCALE[,OFFSET]]
                            Specify binary output format for variable VAR (netCDF only); TYPE
                            is one of: int1, int2, int4, real, dble; SCALE and OFFSET are
                            optional (def: 1,0)

Still working for backward compatibility with RADS3 are options:
  --h H0,H1                 Specify range for SLA (m) (now --sla H0,H1)
  --opt J                   Use field number J when J/100 requested (now -A VAR1=VAR2)
  --opt I=J                 Make field I (range 1-99) and alias for field J (now -A VAR1=VAR2)
  --sel VAR1,...            Select variables to read

Common [rads_options] are:
  --args FILENAME           Get all command line arguments from FILENAME (1 argument per line)
  --debug LEVEL             Set debug/verbosity level
  --help                    Print this syntax massage
  --log FILENAME            Send statistics to FILENAME (default is standard output)
  -q, --quiet               Suppress warning messages (but keeps fatal error messages)
  -v, --verbose             Increase verbosity level
  --version                 Print version info only
  --                        Terminates all options; all following command-line arguments are
                            considered non-option arguments, even if they begin with a hyphen

Program specific [program_options] are:
  -r, --reject-on-nan VAR   Reject records if variable VAR on -V specifier is NaN
  -r NR                     Reject records if data item number NR on -V specifier is NaN
  -r 0, -r none, -r         Do not reject records with NaN values
  -r n, -r any              Reject records if any value is NaN
                      Note: If no -r option is given -r sla is assumed
  --step N                  Step through records with stride n (default: 1)
  --maxrec NREC             Specify maximum number of output records (default: unlimited)
  -o, --output OUTNAME      Specify name of a single output file or (when ending in /) directory
                            name for pass files; default is pass files in current directory
```

5.6.2 Common options

For a full explanation of the command line options that are common to most RADS4 programs, see Section 5.2.

5.6.3 Program specific options

The command line options that are specific to rads2nc are listed here. Notice that they have some in common with rads2asc4 (Section 5.4).

-r *VAR* **--reject-on-nan** *VAR*
Do not output records when the value of variable *VAR*, given as an argument to the -V or --VAR option, is set to NaN. By default records for which the sea level anomaly (if requested) is set to NaN are rejected.

-r *NR* **--reject-on-nan** *NR*
For backward compatibility with RADS3, this will eliminate records when the value on of item number *NR* on the -V option is NaN.

-r 0 -r none -r
When either of these is used, no records are eliminated and all NaN values are kept.

-r n -r any
Indicates that a line should not be printed when any of the selected variables is NaN.

--step *N*
Print only one out of *N* records.

--maxrec *NREC*
Specify the maximum number of output records. The default is unlimited.

-o *OUTNAME* **--output** *OUTNAME*
By default (without this option) files are created per pass with a name similar to the ones in the data base, i.e., `SSpPPPPcCCC.asc`, where `SS` is the satellite abbreviation, `PPPP` is the pass number, and `CCC` is the cycle number. This option will send the output to a single named file. Alternatively, when *OUTNAME* ends in a slash, then it indicates the directory in which to store the pass files `SSpPPPPcCCC.nc`.

5.6.4 Example

Assuming a default configuration, we issue the following command:

```
$ rads2nc -S e2 -C 0 -P 901,1000,2 -R -8,42,28,48 \
    -V time,lat,lon,sla,swh,wind_speed -o rads2nc.nc -v
```

Just as rads2asc4 the program rads2nc will now print to standard output information on which passes are available and have valid data points in the requested area (Mediterranean Sea), how many records were read and how many remained in the particular area, how many records were rejected based on the selection criteria on the corrections, and finally some statistics on the columns that were requested. At the same time a netCDF file rads2nc.nc will be created. To list its contents, use the ncdump command. The result will look like the following:

```
netcdf rads2nc {
dimensions:
    time = UNLIMITED ; // (558 currently)
variables:
    double time(time) ;
        time:long_name = "time" ;
        time:standard_name = "time" ;
        time:units = "seconds since 1985-01-01 00:00:00 UTC" ;
        time:field = 101s ;
        time:comment = "UTC time of measurement. Attribute leap_second gives time of leap second if any occurs
    int lat(time) ;
        lat:_FillValue = 2147483647 ;
        lat:long_name = "latitude" ;
        lat:standard_name = "latitude" ;
```

```
        lat:units = "degrees_north" ;
        lat:scale_factor = 1.e-06 ;
        lat:field = 201s ;
        lat:comment = "Positive latitude is North latitude, negative latitude is South latitude" ;
    int lon(time) ;
        lon:_FillValue = 2147483647 ;
        lon:long_name = "longitude" ;
        lon:standard_name = "longitude" ;
        lon:units = "degrees_east" ;
        lon:scale_factor = 1.e-06 ;
        lon:field = 301s ;
        lon:comment = "East longitude relative to Greenwich meridian" ;
    short sla(time) ;
        sla:_FillValue = 32767s ;
        sla:long_name = "sea level anomaly" ;
        sla:standard_name = "sea_surface_height_above_sea_level" ;
        sla:units = "m" ;
        sla:quality_flag = "swh sig0 range_rms range_numval flags swh_rms attitude" ;
        sla:scale_factor = 0.0001 ;
        sla:coordinates = "lon lat" ;
        sla:field = 0s ;
        sla:comment = "Sea level determined from satellite altitude - range - all altimetric corrections" ;
    short swh(time) ;
        swh:_FillValue = 32767s ;
        swh:long_name = "Ku-band significant wave height" ;
        swh:standard_name = "sea_surface_wave_significant_height" ;
        swh:units = "m" ;
        swh:scale_factor = 0.001 ;
        swh:coordinates = "lon lat" ;
        swh:field = 17s ;
    short wind_speed(time) ;
        wind_speed:_FillValue = 32767s ;
        wind_speed:long_name = "altimeter wind speed" ;
        wind_speed:standard_name = "wind_speed" ;
        wind_speed:source = "altimeter" ;
        wind_speed:units = "m/s" ;
        wind_speed:scale_factor = 0.01 ;
        wind_speed:coordinates = "lon lat" ;
        wind_speed:field = 19s ;

// global attributes:
        :Conventions = "CF-1.7" ;
        :title = "RADS 4 pass file" ;
        :institution = "Altimetrics / EUMETSAT / NOAA / TU Delft" ;
        :source = "radar altimeter" ;
        :references = "RADS Data Manual, Issue 4.2 or later" ;
        :featureType = "point" ;
        :ellipsoid = "TOPEX" ;
        :ellipsoid_axis = 6378136.3 ;
        :ellipsoid_flattening = 0.00335281317789691 ;
        :filename = "rads2nc.out" ;
        :mission_name = "ERS-2" ;
        :mission_phase = "a" ;
        :log01 = "2016-06-14 | rads2nc -S e2 -C 0 -P 901,1000,2 -V time,lat,lon,sla,swh,wind_speed -R -8,42,28
        :history = "2016-06-14 20:12:02 : rads2nc -S e2 -C 0 -P 901,1000,2 -V time,lat,lon,sla,swh,wind_speed
data:

 time = 326926314.226853, 326926315.207322, 326926316.187547,
    326926317.168015, 326926318.148484, 326926319.128953, 326926320.109178,
.............  lines removed ..............................
    1129, 1169, 1147, 1155, 1158, 1180, 1162, 1133, 1133, 1066, 1043, 1028,
    987, 957, 546, 555, 526, 529, 564, 565, 574, 611, 615, 634, 618, 644,
    634, 614, 595, 605, 595, 624, 621, 604, 611, 634, 637, 633, 610, 643,
    647, 624, 585, 579, 558, 489, 458, 552, 582 ;
}
```

5.7 radscolin4

The program radscolin4 creates collinear tracks (in a rather straightforward way). No smoothing or interpolation is performed. The data are "gridded" based on their time with respect to the equator passages. The measurements are simply collected in bins of predefined length. By default this is the 1-Hz sampling rate of the specific mission), but it can be altered with the --dt

option. Then the data falling in those bins for each specified pass are output in a way similar to rads2asc4 (Section 5.4) or rads2nc (Section 5.6). No averaging or differencing is performed.

This program can also be used to compute differences between two different altimeter missions, as long as they are collinear. Some variance to the strict collinearity is allowed when using the -f or --force option. See Section 5.3 for more info on using the command line when multiple missions are involved.

The program will output only those bins in which valid sea level anomalies can be found for all specified repeat cycles, unless the -r option is used. This behaviour of -r is significantly different from rads2asc4 and rads2nc. See the syntax description below.

The output are one record for each bin, containing the variables selected by the -V or --var option, for each of the specified cycles. The output can be either ASCII (the default) or netCDF (with the -o or --output) option. In the case of ASCII, normally only the passes with some valid data are output. Using the option -k or --keep keeps all passes, irrespectively.

This program can be called as radscolin, or as rads2colin4 to avoid conflicts with an already installed RADS3.

5.7.1 Syntax

```
radscolin4 (v4.2.4-2-g55d2fc7): Make collinear data sets from RADS

Usage: radscolin4 [required_arguments] [rads_dataselectors] [rads_options] [program_options]

Required argument is:
  -S, --sat SAT[/PHASE]     Specify satellite [and phase] (e.g. e1/g, tx)

Optional [rads_dataselectors] are:
  -A, --alias VAR1=VAR2     Use variable VAR2 when VAR1 is requested
  -C, --cycle C0[,C1[,DC]]  Specify first and last cycle and modulo
  -F, --fmt, --format VAR=FMT
                            Specify Fortran format used to print VAR (for ASCII output only)
  -L, --limits VAR=MIN,MAX  Specify edit data range for variable VAR
  --lon LON0,LON1           Specify longitude boundaries (deg)
  --lat LAT0,LAT1           Specify latitude  boundaries (deg)
  -P, --pass P0[,P1[,DP]]   Specify first and last pass and modulo; alternatively use -Pa
                            (--pass asc) or -Pd (--pass des) to restrict selection to
                            ascending or descending passes only
  -Q, --quality-flag VAR=FLAG
                            Check variable FLAG when validating variable VAR
  -R, --region LON0,LON1,LAT0,LAT1
                            Specify rectangular region (deg)
  -R, --region LON0,LAT0,RADIUS
                            Specify circular region (deg)
  --sla SLA0,SLA1           Specify range for SLA (m)
  --time T0,T1              Specify time selection (optionally use --ymd, --doy,
                            or --sec for [YY]YYMMDD[HHMMSS], [YY]YYDDD, or SEC85)
  -V, --var VAR1,...        Select variables to be read
  -X, --xml XMLFILE         Load configuration file XMLFILE in addition to the defaults
  -Z, --cmp, --compress VAR=TYPE[,SCALE[,OFFSET]]
                            Specify binary output format for variable VAR (netCDF only); TYPE
                            is one of: int1, int2, int4, real, dble; SCALE and OFFSET are
                            optional (def: 1,0)

Still working for backward compatibility with RADS3 are options:
  --h H0,H1                 Specify range for SLA (m) (now --sla H0,H1)
  --opt J                   Use field number J when J/100 requested (now -A VAR1=VAR2)
  --opt I=J                 Make field I (range 1-99) and alias for field J (now -A VAR1=VAR2)
  --sel VAR1,...            Select variables to read

Common [rads_options] are:
  --args FILENAME           Get all command line arguments from FILENAME (1 argument per line)
  --debug LEVEL             Set debug/verbosity level
  --help                    Print this syntax massage
  --log FILENAME            Send statistics to FILENAME (default is standard output)
  -q, --quiet               Suppress warning messages (but keeps fatal error messages)
  -v, --verbose             Increase verbosity level
  --version                 Print version info only
```

```
  --                          Terminates all options; all following command-line arguments are
                              considered non-option arguments, even if they begin with a hyphen

Program specific [program_options] are:
  --dt DT                     Set minimum bin size in seconds (default is determined by satellite)
  --step N                    Write out only one out of N bins along track
  -r, --reject-on-nan VAR     Base rejection criteria (below) on VAR; default is 'sla' or 1st on -V
  -r NR                       Reject stacked data when fewer than NR tracks with valid values
  -r 0, -r none, -r           Keep all stacked data points, even NaN
  -r n, -r any                Reject stacked data when data on any track is NaN (default)
                       Note:  If no -r option is given, -r any is assumed
  -k, --keep                  Keep all passes, even those that do not have data in the selected area
  -a, --mean                  Output mean in addition to pass data
  -s, --stddev                Output standard deviation in addition to pass data
  -l, --minmax                Output minimum and maximum in addition to pass data
  -d, --no-pass               Do not output pass data
  -t, --no-track              Do not print along-track data (ascii output only)
  -c, --cumul                 Output cumulative statistics (ascii output only) (implies --keep)
      --diff                  Compute difference between first and second half of selected passes
                              (implies --keep)
  -f, --force                 Force comparison, even when missions are not considered collinear
  -o, --output [FILENAME]     Create netCDF output by pass (default is ascii output to stdout).
                              Optionally specify FILENAME including "#", to be replaced by the pass
                              number. Default is "radscolin_p#.nc"
  --diff                      Compute the collinear difference between the first and second half of
                              selected tracks
```

5.7.2 Common options

For a full explanation of the command line options that are common to most RADS4 programs, see Section 5.2.

5.7.3 Program specific options

The command line options that are specific to radscolin4 are listed here. Notice that the -r option is somewhat different than that of rads2asc4 (Section 5.4) and rads2nc (Section 5.6).

--dt *DT* Set the minimum bin size in seconds. The default is the maximum 1-Hz sampling rate of the missions selected.

--step *N* Write out only one out of N bins along track.

-r *VAR* **--reject-on-nan** *VAR*
Base the count of "valid passes" on variable whether *VAR* is not NaN. By default radscolin4 looks for the validity of the variable sla, if used in the list of variables on the -V option. If sla is not used, the first variable is checked for validity.

-r *NR*
Reject stacked data when there are fewer than NR tracks with valid values. Which variable is used to determine "valid" is explained above.

-r 0 -r none -r
When either of these is used, keep the stacked data in the output, even when they are NaN.

-r n -r any
Reject stacked data when data on any track is NaN. This is the default behaviour.

-k --keep
Keep all the passes in the output, even those that do not have any valid data in the selected area. By default, passes without any valid values are removed from the output.

-a --mean
Output the mean value in each stacked bin in addition to the values in each collinear pass.

-s --stddev
Output the standard deviation in each stacked bin in addition to the values in each collinear pass.

-l --minmax
Output the minimum and maximum in each stacked bin in addition to the values in each collinear pass.

-d --no-pass
Do not output the values in each individual collinear pass.

-t --no-track
Do not print the along-track data (applies to ASCII output only).

-c --cumul
Print cumulative statistics (applies to ASCII output only).

--diff
Compute the difference between the first and the second half of the selected passes. This is practical when computing the difference between two missions. It implies --keep.

-f --force
Force collinear track comparison, even when missions are not strictly considered collinear.

-o ***[FILENAME]*** **--output** ***[FILENAME]***
Create netCDF output files per pass. The default is ASCII output to standard output. Optionally specify a *FILENAME* including the hash character ('#'), which will be replaced by the pass number. The default netCDF output file name is `radscolin_p#.nc`.

5.7.4 Examples

Assuming a default configuration, we issue the following command:

```
$ radscolin -S e2 -C 0,2 -P 915,1000,2 -R -8,42,28,48 -V lat,sla,swh
```

This command creates to following results to print to standard output:

```
# RADS collinear track file
# Created: 2016-06-14 20:12:03 UTC: radscolin4 -S e2 -C 0,2 -P 915,1000,2 -R -8,42,28,48 -V lat,sla,swh
#
# Pass      = 0915
# Satellite =     e2     e2     e2
# Cycles    =    000    001    002
#
# Column ranges for each variable:
#    1 -   3 : latitude [degrees_north]
#    4 -   6 : sea level anomaly [m]
#    7 -   9 : Ku-band significant wave height [m]
#   10       : number of measurements
#   11       : record number
#
  30.661485  30.649574  30.691639  -0.1002  -0.0295  -0.0370  0.120  0.640  0.840        3      528
  30.719321  30.707412  30.749460  -0.0560  -0.0893  -0.0322  0.000  0.880  0.760        3      529
  30.777142  30.765233  30.807295  -0.1571  -0.0769  -0.0683  0.330  0.760  0.890        3      530
  30.834975  30.823067  30.865128  -0.1572  -0.0539  -0.0816  0.560  0.560  0.960        3      531
.............  lines removed ..............................
  43.299112  43.263665  43.286058   0.0177   0.0423  -0.0285  0.870  0.520  0.410        3      747
# 42.368730  42.333236  42.355645  -0.0133   0.0405  -0.0196  1.022  0.362  0.840        0      999 # avg
#  0.572197   0.572222   0.572209   0.0387   0.0257   0.0678  0.264  0.192  0.505        0      999 # std
#       33         33         33         33         33         33         33         33         33        0      999 # nr

# RADS collinear track file
# Created: 2016-06-14 20:12:03 UTC: radscolin4 -S e2 -C 0,2 -P 915,1000,2 -R -8,42,28,48 -V lat,sla,swh -r
#
# Pass      = 0915
# Satellite =     e2     e2     e2
```

```
# Cycles    =    000    001    002
#
# Column ranges for each variable:
#    1 -   3 : latitude [degrees_north]
#    4 -   6 : sea level anomaly [m]
#    7 -   9 : Ku-band significant wave height [m]
#   10       : number of measurements
#   11       : record number
#
  30.185412        NaN  30.215553      NaN      NaN      NaN  3.740    NaN  6.140        2      520
       NaN  30.418223  30.459773      NaN      NaN      NaN    NaN    NaN  1.680        2      524
  30.487462  30.476067  30.518135      NaN      NaN      NaN  0.770  0.480  0.960        3      525
  30.545822  30.533894  30.575961      NaN      NaN      NaN  0.700  0.770  1.060        3      526
.....  lines removed ..............................
       NaN  47.161520        NaN      NaN      NaN      NaN    NaN  0.250    NaN        1      815
# 42.323495  42.345405  42.355645  -0.0135   0.0420  -0.0196  1.026  0.368  0.840        0      999 # avg
#  0.588227   0.588225   0.572209   0.0386   0.0260   0.0678  0.271  0.198  0.505        0      999 # std
#      35       35       33       35       35       33       35       35       33        0      999 # nr
```

Chapter 6

RADS library

6.1 Module rads

6.1.1 rads

SUMMARY:

```
 RADS main module
```

SYNOPSIS:

```
module rads
use typesizes
use rads_grid, only: grid

! * Parameters
! Dimensions
integer(fourbyteint), parameter :: rads_var_chunk = 100, rads_varl = 40, &
    rads_naml = 160, rads_cmdl = 320, rads_strl = 1600, rads_hstl = 3200, &
    rads_cyclistl = 50, rads_optl = 50, rads_max_branches = 5
! RADS4 data types
integer(fourbyteint), parameter :: rads_type_other = 0, rads_type_sla = 1, &
    rads_type_flagmasks = 2, rads_type_flagvalues = 3, rads_type_time = 11, &
    rads_type_lat = 12, rads_type_lon = 13, rads_type_dim = 14
! RADS4 data sources
integer(fourbyteint), parameter :: rads_src_none = 0, rads_src_nc_var = 10, &
    rads_src_nc_att = 11, rads_src_math = 20, rads_src_grid_lininter = 30, &
    rads_src_grid_splinter = 31, rads_src_grid_query = 32, &
    rads_src_constant = 40, rads_src_flags = 50, rads_src_tpj = 60
! RADS4 warnings
integer(fourbyteint), parameter :: rads_warn_nc_file = -3
! RADS4 errors
integer(fourbyteint), parameter :: rads_noerr = 0, &
    rads_err_nc_file = 1, rads_err_nc_parse = 2, rads_err_nc_close = 3, rads_err_memory = 4, &
    rads_err_var = 5, rads_err_source = 6, rads_err_nc_var = 7, rads_err_nc_get = 8, &
    rads_err_xml_parse = 9, rads_err_xml_file = 10, rads_err_alias = 11, rads_err_math = 12, &
    rads_err_cycle = 13, rads_err_nc_create = 14, rads_err_nc_put = 15
! Additional RADS4 helpers
character(len=1), parameter :: rads_linefeed = char(10), rads_noedit = '_'
! RADS3 errors or incompatibilities
integer(fourbyteint), parameter :: rads_err_incompat = 101, rads_err_noinit = 102
integer(twobyteint), parameter :: rads_nofield = -1
! Math constants
real(eightbytereal), parameter :: pi = 3.1415926535897932d0, rad = pi/180d0
! I/O parameters
integer, parameter :: stderr = 0, stdin = 5, stdout = 6

! * Variables
! I/O variables
integer(fourbyteint), save :: rads_verbose = 0       ! Verbosity level
integer(fourbyteint), save :: rads_log_unit = stdout ! Unit number for statistics logging

! * RADS4 variable structures
type :: rads_varinfo                                 ! Information on variable used by RADS
```

```
    character(len=rads_varl) :: name                ! Short name of variable used by RADS
    character(len=rads_naml) :: long_name           ! Long name (description) of variable
    character(len=rads_naml) :: standard_name       ! Optional CF 'standard' name ('' if none)
    character(len=rads_naml) :: source              ! Optional data source ('' if none)
    character(len=rads_naml) :: parameters          ! Optional link to model parameters ('' if none)
    character(len=rads_strl) :: dataname            ! Name associated with data (e.g. netCDF var name)
    character(len=rads_cmdl) :: flag_meanings       ! Optional meaning of flag values ('' if none)
    character(len=rads_cmdl) :: quality_flag        ! Quality flag(s) associated with var ('' if none)
    character(len=rads_cmdl) :: comment             ! Optional comment ('' if none)
    character(len=rads_varl) :: units               ! Optional units of variable ('' if none)
    character(len=rads_varl) :: format              ! Fortran format for output
    character(len=rads_varl) :: gridx, gridy        ! RADS variable names of the grid x and y coords
    type(grid), pointer :: grid                     ! Pointer to grid (if data source is grid)
    real(eightbytereal) :: default                  ! Optional default value (Inf if not set)
    real(eightbytereal) :: limits(2)                ! Lower and upper limit for editing
    real(eightbytereal) :: plot_range(2)            ! Suggested range for plotting
    real(eightbytereal) :: add_offset, scale_factor ! Offset and scale factor in case of netCDF
    real(eightbytereal) :: xmin, xmax, mean, sum2   ! Minimum, maximum, mean, sum squared deviation
    logical :: boz_format                           ! Format starts with B, O or Z.
    integer(fourbyteint) :: ndims                   ! Number of dimensions of variable
    integer(fourbyteint) :: brid                    ! Branch ID (default 1)
    integer(fourbyteint) :: nctype, varid           ! netCDF data type (nf90_int, etc.) and var ID
    integer(fourbyteint) :: datatype                ! Type of data (one of rads_type_*)
    integer(fourbyteint) :: datasrc                 ! Retrieval source (one of rads_src_*)
    integer(fourbyteint) :: cycle, pass             ! Last processed cycle and pass
    integer(fourbyteint) :: selected, rejected      ! Number of selected or rejected measurements
endtype

type :: rads_var                                    ! Information on variable or alias
    character(len=rads_varl), pointer :: name       ! Pointer to short name of variable (or alias)
    character(len=rads_naml), pointer :: long_name  ! Pointer to long name (description) of variable
    type(rads_varinfo), pointer :: info, inf1, inf2 ! Links to structs of type(rads_varinfo)
    logical(twobyteint) :: noedit                   ! .true. if editing is suspended
    integer(twobyteint) :: field(2)                 ! RADS3 field numbers (rads_nofield = none)
endtype

type :: rads_cyclist                                ! List of cycles
    integer(fourbyteint) :: n, i                    ! Number of elements in list, additional value
    integer(fourbyteint) :: list(rads_cyclistl)     ! List of values
endtype

type :: rads_phase                                  ! Information about altimeter mission phase
    character(len=rads_varl) :: name, mission       ! Name (1-letter), and mission description
    integer(fourbyteint) :: cycles(2), passes       ! Cycle range and maximum number of passes
    real(eightbytereal) :: start_time, end_time     ! Start time and end time of this phase
    real(eightbytereal) :: ref_time, ref_lon        ! Time and lon of equator crossing of "ref. pass"
    integer(fourbyteint) :: ref_cycle, ref_pass     ! Cycle and pass number of "reference pass"
    real(eightbytereal) :: pass_seconds             ! Length of pass in seconds
    real(eightbytereal) :: repeat_days              ! Length of repeat period in days
    real(eightbytereal) :: repeat_shift             ! Eastward shift of track pattern for near repeats
    integer(fourbyteint) :: repeat_nodal            ! Length of repeat period in nodal days
    integer(fourbyteint) :: repeat_passes           ! Number of passes per repeat period
    type(rads_cyclist), pointer :: subcycles        ! Subcycle definition (if requested)
endtype

type :: rads_sat                                    ! Information on altimeter mission
    character(len=rads_naml) :: userroot            ! Root directory of current user (i.e. $HOME)
    character(len=rads_naml) :: dataroot            ! Root directory of RADS data (i.e. $RADSDATAROOT)
    character(len=rads_varl) :: branch(rads_max_branches) ! Name of optional branches
    character(len=rads_varl) :: spec                ! Temporary holding space for satellite specs
    character(len=rads_cmdl) :: command             ! Command line
    character(len=rads_naml), pointer :: glob_att(:) ! Global attributes
    character(len=8) :: satellite                   ! Satellite name
    real(eightbytereal) :: dt1hz                    ! "1 Hz" sampling interval
    real(eightbytereal) :: frequency(2)             ! Frequency (GHz) of primary and secondary channel
    real(eightbytereal) :: inclination              ! Satellite inclination (deg)
    real(eightbytereal) :: eqlonlim(0:1,2)          ! Equator lon limits for asc. and desc. passes
    real(eightbytereal) :: centroid(3)              ! Lon, lat, distance (in rad) selection criteria
    real(eightbytereal) :: xover_params(2)          ! Crossover parameters used in radsxoconv
    integer(fourbyteint) :: cycles(3),passes(3)     ! Cycle and pass limits and steps
    integer(fourbyteint) :: error                   ! Error code (positive = fatal, negative = warning)
    integer(fourbyteint) :: pass_stat(7)            ! Stats of rejection at start of rads_open_pass
    integer(fourbyteint) :: total_read, total_inside ! Total nr of measurements read and inside region
    integer(fourbyteint) :: nvar, nsel              ! Nr of available and selected vars and aliases
    logical :: n_hz_output                          ! Produce multi-Hz output
    character(len=2) :: sat                         ! 2-Letter satellite abbreviation
    integer(twobyteint) :: satid                    ! Numerical satellite identifier
    type(rads_cyclist), pointer :: excl_cycles      ! Excluded cycles (if requested)
```

```
    type(rads_var), pointer :: var(:)                ! List of available variables and aliases
    type(rads_var), pointer :: sel(:)                ! List of selected variables and aliases
    type(rads_var), pointer :: time, lat, lon        ! Pointers to time, lat, lon variables
    type(rads_phase), pointer :: phases(:)           ! Definitions of all mission phases
    type(rads_phase), pointer :: phase               ! Pointer to current phase
endtype

type :: rads_file                                    ! Information on RADS data file
    integer(fourbyteint) :: ncid                     ! NetCDF ID of pass file
    character(len=rads_cmdl) :: name                 ! Name of the netCDF pass file
endtype

type :: rads_pass                                    ! Pass structure
    character(len=rads_strl) :: original             ! Name of the original (GDR) pass file(s)
    character(len=rads_hstl), pointer :: history     ! File creation history
    real(eightbytereal) :: equator_time, equator_lon ! Equator time and longitude
    real(eightbytereal) :: start_time, end_time      ! Start and end time of pass
    real(eightbytereal), pointer :: tll(:,:)         ! Time, lat, lon matrix
    integer(twobyteint), pointer :: flags(:)         ! Array of engineering flags
    logical :: rw                                    ! NetCDF file opened for read/write
    integer(fourbyteint) :: cycle, pass              ! Cycle and pass number
    integer(fourbyteint) :: nlogs                    ! Number of RADS3 log entries
    integer(fourbyteint) :: ndata                    ! Number of data points (1-Hz)
    integer(fourbyteint) :: n_hz, n_wvf              ! Size second/third dimension (0=none)
    integer(fourbyteint) :: first_meas, last_meas    ! Index of first and last point in region
    integer(fourbyteint) :: time_dims                ! Dimensions of time/lat/lon stored
    integer(fourbyteint) :: trkid                    ! Numerical track identifiers
    type (rads_file) :: fileinfo(rads_max_branches)  ! File information for pass files
    type (rads_sat), pointer :: S                    ! Pointer to satellite/mission structure
    type (rads_pass), pointer :: next                ! Pointer to next pass in linked list
endtype

type :: rads_option                                  ! Information on command line options
    character(len=rads_varl) :: opt                  ! Option (without the - or --)
    character(len=rads_cmdl) :: arg                  ! Option argument
    integer :: id                                    ! Identifier in form 10*nsat + i
endtype

! These command line options can be accessed by RADS programs
type(rads_option), allocatable, target, save :: &
    rads_opt(:)                                      ! List of command line options
integer(fourbyteint), save :: rads_nopt = 0          ! Number of command line options saved
```

PURPOSE:

```
 This module provides the main functionalities for the RADS4 software.
 To use any of the following subroutines and functions, add the following
 line in your Fortran 90 (or later) code:
   use rads
```

COPYRIGHT:

6.1.2 rads_init

SUMMARY:

```
 Initialize RADS4
```

SYNTAX:

```
subroutine rads_init (S, sat, xml)
type(rads_sat), intent(inout) :: S <or> S(:)
character(len=*), intent(in), optional :: sat <or> sat(:)
character(len=*), intent(in), optional :: xml(:)
```

PURPOSE:

```
This routine initializes the <S> struct with the information pertaining
to given satellite/mission phase <sat>, which is to be formed as 'e1',
or 'e1g', or 'e1/g'. If no phase is specified, all mission phases will be
queried.

The <S> and <sat> arguments can either a single element or an array. In the
latter case, one <S> struct will be initialized for each <sat>.
To parse command line options after this, use rads_parse_cmd.

Only if the <sat> argument is omitted, then the routine will parse
the command line for arguments in the form:
--sat=<sat> --cycle=<lo>,<hi>,<step> --pass=<lo>,<hi>,<step>
--lim:<var>=<lo>,<hi> --lat=<lo>,<hi> --lon=<lo>,<hi> --alias:<var>=<var>
--opt:<value>=<value> --opt=<value>,... --fmt:<var>=<value>
or their equivalents without the = or : separators after the long name,
or their equivalents without the initial --, or the short options -S, -C,
-P, -L, -F

The routine will read the satellite/mission specific setup XML files and
store all the information in the stuct <S>. The XML files polled are:
  $RADSDATAROOT/conf/rads.xml
  ~/.rads/rads.xml
  rads.xml
  <xml> (from the optional array of file names)

If more than one -S option is given, then all further options following
this argument until the next -S option, plus all options prior to the
first -S option will pertain to this mission.

Execution will be halted when the dimension of <S> is insufficient to
store information of multiple missions, or when required XML files are
missing.

The verbosity level can be controlled by setting rads_verbose before
calling this routine (default = 0). The output unit for log info can
be controlled by setting rads_log_unit up front (default = stdout).
```

ARGUMENTS:

```
S        : Satellite/mission dependent structure
sat      : (optional) Satellite/mission abbreviation
xml      : (optional) Array of names of additional XML files to be loaded
```

6.1.3 rads_end

SUMMARY:

```
End RADS4
```

SYNTAX:

```
subroutine rads_end (S)
type(rads_sat), intent(inout) :: S <or> S(:)
```

PURPOSE:

```
This routine ends RADS by freeing up all <S> space and other allocated
global arrays.
```

ARGUMENT:

```
S        : Satellite/mission dependent struct or array of structs
```

6.1.4 rads_get_var

SUMMARY:

```
Read variable (data) from RADS4 file
```

SYNTAX:

```
recursive subroutine rads_get_var (S, P, var, data, noedit)
type(rads_sat), intent(inout) :: S
type(rads_pass), intent(inout) :: P
character(len=*) :: var
<or> integer(fourbyteint) :: var
<or> type(rads_var), intent(in) :: var
real(eightbytereal), intent(out) :: data(:)
logical, intent(in), optional :: noedit
```

PURPOSE:

```
This routine loads the data from a single variable <var> into the
buffer <data>. This command must be preceeded by <rads_open_pass>.
The variable <var> can be addressed as a variable name, a RADS3-type
field number or a varlist item.

The array <data> must be at the correct size to contain the entire
pass of data, i.e., it must have the dimension P%ndata.
If no data are available and no default value and no secondary aliases
then NaN is returned in the array <data>.
```

ARGUMENTS:

```
S        : Satellite/mission dependent structure
P        : Pass dependent structure
var      : (string) Name of the variable to be read.
                    If <var> ends with % editing is skipped.
           (integer) Field number.
           (type(rads_var)) Variable struct (e.g. S%sel(i))
data     : Data returned by this routine
noedit   : (optional) Set to .true. to skip editing on limits and/or
           quality flags; set to .false. to allow editing (default)
```

ERROR CODE:

```
S%error  : rads_noerr, rads_err_var, rads_err_memory, rads_err_source
```

6.1.5 rads_stat

SUMMARY:

```
Print the RADS statistics for a given satellite
```

SYNTAX:

```
subroutine rads_stat (S)
type(rads_sat), intent(in) :: S <or> S(:)
integer(fourbyteint), intent(in), optional :: unit
```

PURPOSE:

```
This routine prints out the statistics of all variables that were
processed per mission (indicated by scalar or array <S>), to the output
on unit <rads_log_unit>.
```

ARGUMENTS:

```
S        : Satellite/mission dependent structure
```

6.1.6 rads_init_sat_struct

SUMMARY:

```
Initialize empty rads_sat struct
```

SYNOPSIS:

```
pure subroutine rads_init_sat_struct (S)
type(rads_sat), intent(inout) :: S
```

PURPOSE:

```
This routine initializes the <S> struct with the bare minimum.
It is later updated in rads_init_sat_0d.
```

ARGUMENTS:

```
S        : Satellite/mission dependent structure
```

6.1.7 rads_init_pass_struct

SUMMARY:

```
Initialize empty rads_pass struct
```

SYNOPSIS:

```
subroutine rads_init_pass_struct (S, P)
type(rads_sat), target, intent(in) :: S
type(rads_pass), intent(inout) :: P
```

PURPOSE:

```
This routine initializes the <P> struct with the bare minimum.
This is only really necessary prior to calling rads_create_pass.
```

ARGUMENTS:

```
S        : Satellite/mission dependent structure
P        : Pass dependent structure
```

6.1.8 rads_set_options

SUMMARY:

```
Specify the list of command specific options

SYNPOSIS
```

PURPOSE:

```
Add the command specific options to the list of common RADS options.
The argument <optlist> needs to have the same format as in the routine
<getopt> in the <rads_misc> module. The short options will be placed
before the common ones, the long options will be placed after them.
```

ARGUMENT:

```
optlist  : (optional) list of command specific short and long options
```

6.1.9 rads_open_pass

SUMMARY:

```
Open RADS pass file
```

SYNOPSIS:

```
subroutine rads_open_pass (S, P, cycle, pass, rw)
use netcdf
use rads_netcdf
use rads_time
use rads_misc
use rads_geo
type(rads_sat), intent(inout) :: S
type(rads_pass), intent(inout) :: P
integer(fourbyteint), intent(in) :: cycle, pass
logical, intent(in), optional :: rw
```

PURPOSE:

```
This routine opens a netCDF file for access to the RADS machinery.
However, prior to opening the file, three tests are performed to speed
up data selection:
(1) All passes outside the preset cycle and pass limits are rejected.
(2) Based on the time of the reference pass, the length of the repeat
    cycle and the number of passes per cycle, a rough estimate is
    made of the temporal extent of the pass. If this is outside the
    selected time window, then the pass is rejected.
(3) Based on the equator longitude and the pass number of the reference
    pass, the length of the repeat cycle and the number of passes in
    the repeat cycle, an estimate is made of the equator longitude of
    the current pass. If this is outside the limits set in S%eqlonlim
    then the pass is rejected.

If the pass is rejected based on the above critetia or when no netCDF
file exists, S%error returns the warning value rads_warn_nc_file.
If the file cannot be read properly, rads_err_nc_parse is returned.
Also, in both cases, P%ndata will be set to zero.

By default the file is opened for reading only. Specify perm=nf90_write to
open for reading and writing.
The file opened with this routine should be closed by using rads_close_pass.
```

ARGUMENTS:

```
S        : Satellite/mission dependent structure
P        : Pass structure
cycle    : Cycle number
pass     : Pass number
rw       : (optional) Set read/write permission (def: read only)
```

ERROR CODE:

```
S%error  : rads_noerr, rads_warn_nc_file, rads_err_nc_parse
```

6.1.10 rads_close_pass

SUMMARY:

```
Close RADS pass file
```

SYNOPSIS:

```
subroutine rads_close_pass (S, P, keep)
use netcdf
use rads_netcdf
type(rads_sat), intent(inout) :: S
type(rads_pass), intent(inout) :: P
logical, intent(in), optional :: keep
```

PURPOSE:

```
This routine closes a netCDF file previously opened by rads_open_pass.
The routine will reset the ncid element of the <P> structure to
indicate that the passfile is closed.
If <keep> is set to .true., then the history, flags, time, lat, and lon
elements the <P> structure are kept. I.e., they are not deallocated
but only their links are removed. Otherwise, they are deallocated along
with the log entries.
A second call to rads_close_pass without the keep argment can subsequently
deallocate the time, lat and lon elements of the <P> structure.
```

ARGUMENTS:

```
S        : Satellite/mission dependent structure
P        : Pass structure
keep     : Keep the P%tll matrix (destroy by default)
```

ERROR CODE:

```
S%error  : rads_noerr, rads_err_nc_close
```

6.1.11 rads_read_xml

SUMMARY:

```
Read RADS4 XML file
```

SYNOPSIS:

```
subroutine rads_read_xml (S, filename)
use netcdf
use xmlparse
use rads_time
use rads_misc
type(rads_sat), intent(inout) :: S
character(len=*), intent(in) :: filename
```

PURPOSE:

```
This routine parses a RADS4 XML file and fills the <S> struct with
information pertaining to the given satellite and all variable info
encountered in that file.

The execution terminates on any error, and also on any warning if
fatal = .true.
```

ARGUMENTS:

```
S        : Satellite/mission dependent structure
filename : XML file name
fatal    : If .true., then all warnings are fatal.
```

ERROR CODE:

```
S%error  : rads_noerr, rads_err_xml_parse, rads_err_xml_file
```

6.1.12 rads_set_alias

SUMMARY:

```
Set alias to an already defined variable
```

SYNOPSIS:

```
subroutine rads_set_alias (S, alias, varname, field)
```

```
use rads_misc
type(rads_sat), intent(inout) :: S
character(len=*), intent(in) :: alias, varname
integer(twobyteint), intent(in), optional :: field(2)
```

PURPOSE:

```
This routine defines an alias to an existing variable, or up to three
variables. When more than one variable is given as target, they will
be addressed one after the other.
If alias is already defined as an alias or variable, it will be overruled.
The alias will need to point to an already existing variable or alias.
Up to three variables can be specified, separated by spaces or commas.
```

ARGUMENTS:

```
S        : Satellite/mission dependent structure
alias    : New alias for (an) existing variable(s)
varname  : Existing variable name(s)
field    : (optional) new field numbers to associate with alias
```

ERROR CODE:

```
S%error  : rads_noerr, rads_err_alias, rads_err_var
```

6.1.13 rads_set_limits

SUMMARY:

```
Set limits on given variable
```

SYNOPSIS:

```
subroutine rads_set_limits (S, varname, lo, hi, string, iostat)
use rads_misc
type(rads_sat), intent(inout) :: S
character(len=*), intent(in) :: varname
real(eightbytereal), intent(in), optional :: lo, hi
character(len=*), intent(in), optional :: string
integer(fourbyteint), intent(out), optional :: iostat
```

PURPOSE:

```
This routine set the lower and upper limits for a given variable in
RADS.
The limits can either be set by giving the lower and upper limits
as double floats <lo> and <hi> or as a character string <string> which
contains the two numbers separated by whitespace, a comma or a slash.
In case only one number is given, only the lower or higher bound
(following the separator) is set, the other value is left unchanged.
```

ARGUMENTS:

```
S        : Satellite/mission dependent structure
varname  : Variable name
lo, hi   : Lower and upper limit
string   : String of up to two values, with separating whitespace
           or comma or slash.
iostat   : (optional) iostat code from reading string
```

ERROR CODE:

```
S%error  : rads_noerr, rads_err_var
```

6.1.14 rads_set_region

SUMMARY:

```
 Set latitude/longitude limits or distance to point
```

SYNOPSIS:

```
subroutine rads_set_region (S, string)
use rads_misc
type(rads_sat), intent(inout) :: S
character(len=*), intent(in) :: string
```

PURPOSE:

```
 This routine set the region for data selection (after the -R option).
 The region can either be specified as a box by four values "W/E/S/N",
 or as a circular region by three values "E/N/radius". Separators
 can be commas, slashes, or whitespace.

 In case of a circular region, longitude and latitude limits are set
 accordingly for a rectangular box surrounding the circle. However, when
 reading pass data, the distance to the centroid is used as well to
 edit out data.
```

ARGUMENTS:

```
 S        : Satellite/mission dependent structure
 string   : String of three or four values with separating whitespace.
            For rectangular region: W/E/S/N.
            For circular region: E/N/radius (radius in degrees).
```

ERROR CODE:

```
 S%error  : rads_noerr, rads_err_var
```

6.1.15 rads_set_format

SUMMARY:

```
 Set print format for ASCII output of given variable
```

SYNOPSIS:

```
subroutine rads_set_format (S, varname, format)
type(rads_sat), intent(inout) :: S
character(len=*), intent(in) :: varname, format
```

PURPOSE:

```
 This routine set the FORTRAN format specifier of output of a given
 variable in RADS.
```

ARGUMENTS:

```
 S        : Satellite/mission dependent structure
 varname  : Variable name
 format   : FORTRAN format specifier (e.g. 'f10.3')
```

ERROR CODE:

```
 S%error  : rads_noerr, rads_err_var
```

Bibliography

Francis, C. R. (1990), The ERS-1 radar altimeter, paper presented at the 2nd ERS-1 PI meeting, Noordwijk,The Netherlands.

Francis, C. R., et al. (1995), The ERS-2 spacecraft and its payload, *ESA Bulletin*, *83*, 13–31.

Francis, C. R., et al. (1991), The ERS-1 spacecraft and its payload, *ESA Bulletin*, *65*, 26–48.

Fu, L.-L., E. J. Christensen, C. A. Yamarone, M. Lefebvre, Y. Ménard, M. Dorrer, and P. Escudier (1994), TOPEX mission overview, *J. Geophys. Res.*, *99*(C12), 24,369–24,382.

Lambin, J., et al. (2010), The OSTM/Jason-2 mission, *Mar. Geod.*, *33*(S1), 4–25, doi:10.1080/01490419.2010.491030.

Ménard, Y., L.-L. Fu, P. Escudier, B. J. Haines, G. Kunstmann, F. Parisot, J. Perbos, P. Vincent, and S. D. Desai (2003), The Jason-1 mission, *J. Mar. Geod.*, *Jason-1*.

Wingham, D. J., et al. (2006), CryoSat: A mission to determine the fluctuations in Earth's land and marine ice fields, *Adv. Space Res.*, *37*(4), 841–871, doi:10.1016/j.asr.2005.07.027.

Index