The EUMETSAT Network of Satellite Application Facilities

ROM SAF
Radio Occultation Meteorology

ROM SAF CDOP-2

The Radio Occultation Processing Package (ROPP) User Guide Part I: Input/Output module

Version 7.0

31 July 2013

Danish Meteorological Institute (DMI)
European Centre for Medium-Range Weather Forecasts (ECMWF)
Institut d'Estudis Espacials de Catalunya (IEEC)
Met Office (MetO)

Ref: SAF/ROM/METO/UG/ROPP/002

Document Author Table

	Name	Function	Date	Comment
Prepared by:	I. Culverwell	ROM SAF Project Team	31 July 2013	
Reviewed by:	D. Offiler	ROM SAF Project Team	31 July 2013	
Approved by:	K.B. Lauritsen	ROM SAF Project Manager	31 July 2013	

Document Change Record

Issue/Revision	Date	By	Description
Version 0.1	01 Nov 2004	CM	ROPP User Guide: Preliminary release for I–RR
Version 0.7	04 Jun 2005	CM	ROPP User Guide: Intermediate release for CPM.
Version 0.8	17 Oct 2005	DO	ROPP User Guide: First Beta release (v0.8)
Version 0.9	10 Nov 2006	DO	ROPP User Guide: Second Beta release (v0.9)
Version 1.0	01 Mar 2007	DO	ROPP User Guide: First Full release (v1.0)
Version 1.1	01 Mar 2008	DO	ROPP User Guide: Updates to First Full release (v1.1)
Version 1.2	01 Jul 2008	HL	ROPP User Guide: Updates to First Full release (v1.2)
Version 2.0	01 Dec 2008	HL	ROPP User Guide: Second Full release (v2.0)
Version 3.0	01 Jun 2009	HL	ROPP User Guide: Third Full release (v3.0)
Version 4.0	01 Nov 2009	HL	ROPP User Guide: Fourth Full release (v4.0)
Version 4.1	01 Jun 2010	HL	ROPP User Guide: Updates to Fourth Full release (v4.1)
Version 5.0	14 Jun 2011	IC	ROPP User Guide: Fifth Full release (v5.0)
Version 6.0	31 Oct 2011	IC	ROPP User Guide: Sixth Full release (v6.0)
Version 6.1	31 Jan 2013	IC	ROPP User Guide: Updates to Sixth Full release (v6.1)
Version 7.0	31 Jul 2013	IC	ROPP User Guide: Seventh Full release (v7.0)

ROM SAF

The ROM SAF is the EUMETSAT Satellite Application Facility responsible for operational processing of radio occultation data from the Metop satellites. It delivers bending angle, refractivity, temperature, pressure, and humidity profiles in near-real time and offline for NWP and climate users. The offline profiles are further processed into climate products consisting of gridded monthly zonal means of bending angle, refractivity, temperature, humidity, and geopotential heights together with error descriptions.

The ROM SAF also maintains the Radio Occultation Processing Package (ROPP) which contains software modules that will aid users wishing to process, quality-control and assimilate radio occultation data from any radio occultation mission into NWP and other models.

The ROM SAF Leading Entity is the Danish Meteorological Institute (DMI), with Cooperating Entities: i) European Centre for Medium-Range Weather Forecasts (ECMWF) in Reading, United Kingdom, ii) Institut D'Estudis Espacials de Catalunya (IEEC) in Barcelona, Spain, and iii) Met Office in Exeter, United Kingdom. To get access to our products or to read more about the project please go to http://www.romsaf.org.

Intellectual Property Rights

Contents

1 Introduction

1.1 Purpose of this document

This document provides a User Guide for the data handling tools provided as part of the Radio Occultation Processing Package (ROPP). A generic ROPP data format and software to read and write radio occultation data are provided as part of ROPP. These are described in this document.

The ROPP User Guide Part II (2013a) provides details of the relevant forward model and retrieval software provided with ROPP for users to perform 1D–Var retrievals using refractivity or bending angle data. The ROPP User Guide Part III (2013b) provides details of the pre-processing software included within ROPP to generate refractivity and bending angle profiles from radio occultation data.

An overview of the ROPP modules and the installation, build and test procedures for ROPP are provided in the appendices. Detailed build and install instructions are contained in the release notes of the individual ROPP software modules.

1.2 ROPP

The aim of ROPP is

> *. . . to provide Users with a comprehensive software package, containing all necessary functionality to pre-process RO data from Level 1a (Phase), Level 1b (Bending Angle) or Level 2 (Refractivity) files, plus RO-specific components to assist with the assimilation of these data in NWP systems.*

ROPP is a collection of software modules (provided as source code), supporting data files and documentation, which aids users wishing to assimilate radio occultation data into their NWP models. As far as is practical, the ROPP software is generic, in that it can handle any standard GNSS–LEO configuration radio occultation mission (GRAS, COSMIC, CHAMP, GRACE, C/NOFS, SAC–C, TerraSAR–X, TanDEM-X, ROSA, PAZ, etc).

The software is distributed in the form of a source code library written in Fortran 90. ROPP is implemented using Fortran modules and derived types, enabling the use of object oriented techniques such as the overloading of routines. The software is split into several modules. Figure 1.1 illustrates the interrelationships between each module. Users may wish to integrate a subset of ROPP code into their own software applications, individually linking modules to their own code. These users may not require the complete ROPP distribution package. Alternatively, users may wish to use the executable tools provided as part of each module as stand-alone applications for RO data processing. These users should download the complete ROPP release.

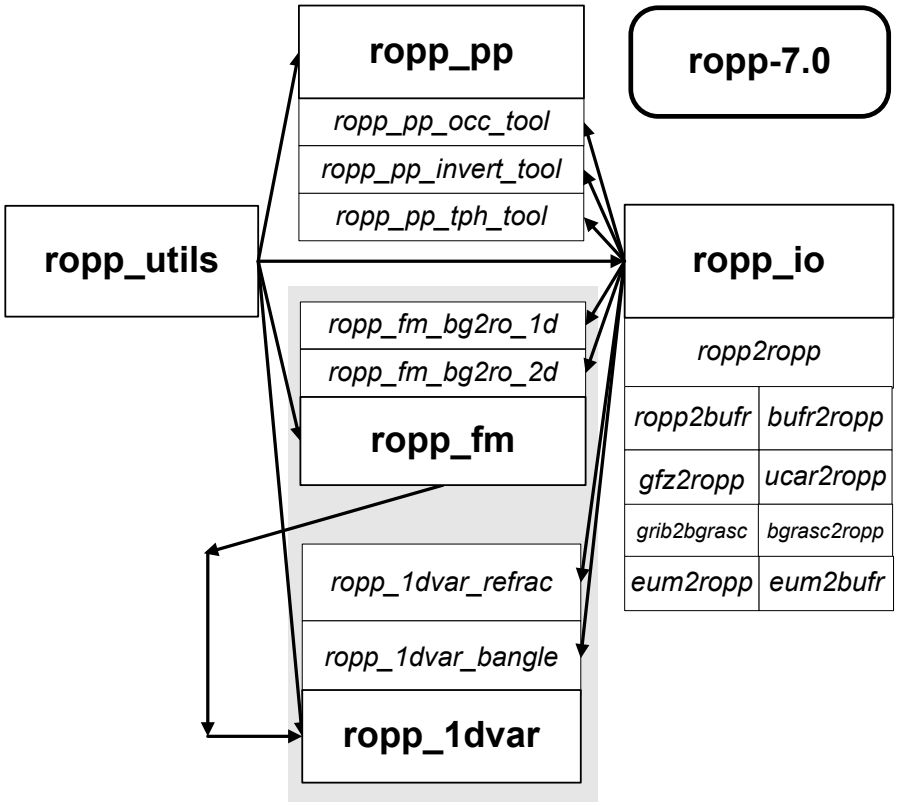

Figure 1.1: ROPP modules and their relationships. Stand-alone applications provided with each module are shown in italics. Connecting lines indicate module dependencies, with arrow heads pointing to the modules or stand-alone tools which require that module.

ROPP contains support for a generic data format for radio occultation data (`ropp_io`), pre-processor routines (`ropp_pp`), one- and two-dimensional forward models (`ropp_fm`), tools for quality control (immersed within the other modules, principally `ropp_1dvar`) and routines for the implementation of 1D–Var retrievals (`ropp_1dvar`). Utility routines used by some or all of the ROPP modules are provided in an additional module (`ropp_utils`). This structure (Figure 1.1) reflects the various degrees of interdependence of the difference ROPP modules. For example, the subroutines and functions in `ropp_io` and `ropp_fm` modules are mutually independent, whereas routines in `ropp_1dvar` depend on `ropp_fm`. Sample standalone implementations of `ropp_pp`, `ropp_fm` and `ropp_1dvar` (which then require `ropp_io` for file interfaces, reading and writing data) are provided with those modules and documented in the relevant User Guides.

1.3 User documentation

A full list of user documentation is provided in Table E.1 and E.2. These documents are available via the ROM SAF website at http://www.romsaf.org.

The ROPP distribution website has a Release Notes (html) file in the root directory which provides a 'Quick Start' guide to the package. This should be read before downloading the package files. Detailed build and install instructions are contained in the release notes of the individual ROPP software modules.

This ROPP IO User Guide provides documentation to the generic ROPP data format and the software used to read and write radio occultation data. An overview on how to install, build and use it, along with a description of the ROPP tools used by other ROPP modules are also provided.

Module-specific user guides for the Pre-processor (ROM SAF, 2013b) and Forward Model and 1D–Var (ROM SAF, 2013a) modules describe the algorithms and routines used in those modules. These provide the necessary background and descriptions of the ROPP software for users to process radio occultation data from excess phase to bending angle or refractivity, and to perform 1D–Var retrievals of radio occultation data and implement ROPP in their own applications.

More detailed Reference Manuals are also available for each module for users wishing to write their own interfaces to the ROPP routines, or to modify the ROPP code. These are provided in the associated module distribution files.

Further documentation can be downloaded in PDF format from the ROPP section of the ROM SAF web site http://www.romsaf.org. The full user documentation set is listed in Table E.1.

In addition to these PDF documents, most of the stand-alone application programs have Unix-style 'man page' help files which are installed during the build procedures. All such programs have summary help information which is available by running the command with the `-h` switch.

Any comments on the ROPP software should in the first instance be raised via the ROM SAF Helpdesk at http://www.romsaf.org.

References

ROM SAF, The Radio Occultation Processing Package (ROPP) User Guide. Part II: Forward model and 1dVar modules, SAF/ROM/METO/UG/ROPP/003, Version 7.0, 2013a.

ROM SAF, The Radio Occultation Processing Package (ROPP) User Guide. Part III: Pre-processor module, SAF/ROM/METO/UG/ROPP/004, Version 7.0, 2013b.

2 ROPP data format

The ROPP data format is a generic data format for radio occultation and related data beginning with level 1a products, i.e. amplitude and (excess) phase data obtained from GNSS radio occultation measurements. For example, an individual radio occultation measurement (or profile), when stored in ROPP format, may be thought of as consisting of the following parts[1]:

Header: Meta data like date, time and location of the occultation

Level 1: "raw" radio occultation data:

- **Level 1a:** SNR, (excess) phases and POD data as function of time
- **Level 1b:** bending angle as function of impact parameter

Level 2: "processed" geophysical data:

- **Level 2a:** refractivity on "observed" height levels
- **Level 2b:** temperature, humidity pressure and geopotential height on "model" levels
- **Level 2c:** surface pressure, surface geopotential height
- **Level 2d:** additional data describing the vertical level structure (e.g., level coefficients for vertical hybrid coordinates)

Apart from the header, all other parts are optional. Thus, a ROPP file may contain only the Level 1a part, i.e. signal–to–noise ratios, excess phases and orbit data; another file (consisting of the header and Levels 1a, 1b and 2a) might contain the raw data, but also bending angle, refractivity and dry temperature profiles calculated from these. Finally, a refractivity-based 1D–Var retrieval, together with the refractivity it was calculated from, would be put into an ROPP data file which contains a header and the Level 2a and 2b parts.

In a similar way, auxiliary data, e.g. vertical profiles of temperature and humidity obtained from NWP analyses or short range forecasts, might be stored in an ROPP file. In this case, only Levels 2b and possibly 2c might be present. If such data serves as background for a variational retrieval as implemented in ROPP, Levels 2b, 2c and 2d (if the background data is given on some sort of model levels instead of fixed pressure or altitude levels) will be present. On the other hand, forward–simulated radio occultation "measurements", e.g. refractivity, bending angles or even forward–simulated amplitudes and phases for a given orbit configuration might also be stored as ROPP file by using the appropriate levels of the data model.

[1] The 'Level' characterisation used in the description of the ROPP data model is not related to the formal data levels of any specific radio occultation instrument.

Technically, the ROPP data format is implemented using the netCDF scientific data format (Unidata), which is a platform–independent, self–describing format for scientific data. Variables (which may be scalars or arrays with several dimensions) are accessed by name; along with each variable, meta–data such as the data's physical units and valid numerical ranges can be stored as "attributes" of the data.

The netCDF data model has no way to represent structures. Therefore, all structure elements from the `ropp_io` data types are mapped to a set of unique variable names in the netCDF data file. The netCDF variable names and their attributes (see below) can be inspected using the **ncdump(1)** utility that is built along with the netCDF interface library — for instance `ncdump -c file.nc`. The mapping from internal netCDF variable names and the ROPP structure names will be obvious as the last part of the structure name is used for the netCDF variable name. For instance, `ROdata%Lev1a%phase_L1` maps simply to `phase_L1`. The exception is the various `ROdata%Lev??%Npoints` structure variables which map to `dim_lev??` in the netCDF file.

All variables in a netCDF based ROPP data file have standard attributes according to the CF convention[2]. This includes the `long_name` and `valid_range` attributes giving a longer description of the variable's content and the physical validity range of the variable. The latter should be given in the same units as the variable itself. Physical units of each variable are described by the `units` attribute which facilitates automated unit conversions (Section 2.4.3). CF-compliant "standard names" are not yet included in ROPP data structures.

In order to facilitate the multifile option, all data files, including singlefiles, exhibit an unlimited netCDF dimension (or record). Thus, scalar structure elements are mapped into 1–dimensional arrays in the netCDF data representation, and 1–dimensional arrays in the `ROprof` structure are technically 2–dimensional variables in the netCDF. In singlefiles, the unlimited dimension has only one element; in multifiles, the number of records is the number of profiles stored in the data file. This needs to be taken into account when reading `ropp_io` data with means other than the `ropp_io` interface. But as long as the `ropp_io` library is used for reading and writing, the actual format for the data files is not relevant for the user as the access is transparent.

The `ropp_io` module defines a dedicated structure (or 'derived type' in Fortran terminology) `ROprof` to hold data related to a vertical radio occultation profile. It is therefore mandatory to load the `ropp_io` module at the beginning of each program unit using `ropp_io` data types. A detailed definition of the elements of an `ROprof` structure is given in Section 2.3.

```
USE ropp_io
TYPE(ROprof) :: ro_data
```

For applications requiring the use of two-dimensional background data (e.g. `ropp_fm` 2D forward operators), a `ROprof2d` structure is defined. This is equivalent to the profile `ROprof` structure, but Level2b and Level2c data may have an additional horizontal dimension to hold a plane of background information.

```
USE ropp_io
TYPE(ROprof2d) :: ro_data
```

[2] See http://cf-pcmdi.llnl.gov.

Note that all `ropp_io` functionality described below may be applied to either `ROprof` or `ROprof2d` input structures using the same interfaces. Examples in this User Guide are only shown using `ROprof` as input for illustration. Thinning of `ROprof2d` data is not currently provided however.

2.1 Reading data

The subroutine `ropp_io_read` reads the contents of an ROPP data file, and puts its data into the structure `ro_data`. The name of the file to be read is given as the second (mandatory) argument:

```
CALL ropp_io_read(ro_data, filename)
```

On return, `ro_data` will contain all data of the profile contained in the data file in its relevant sublevels. Any previous contents of `ro_data` will be lost. The read ROPP data can then be accessed through the respective elements of the `ROprof` data structure.

If certain data levels were not contained in the data file, the returned structure will have zero elements, and the `...%Npoints` element of the respective level will be set to zero. This provides a mechanism to test if the data file contains data for any specific level. For example, if further processing depends on the existence of Level 1b (bending angle) data, an application might be written as

```
USE ropp_io
TYPE(ROprof) :: ro_data
   ...
CALL ropp_io_read(ro_data, filename)
IF (ro_data%Lev1b%Npoints > 0) THEN
   ! ... process the data ...
ELSE
   !      ... or issue an error message.
END IF
```

The read routine automatically recognises the actual technical file format the data has been written in, and calls the appropriate internal read routines to fill the data structure `ro_data`. If the data file represents a 'multifile' containing more than one radio occultation profile, the first profile in the data set will be read by default. There are additional options to the `ropp_io_read` routine which allow reading other than the first profile (see Section 2.4.8).

2.2 Writing data

Writing data to an ROPP file requires the following steps:

- *Declare a structure that holds the content of the ROPP data file*
- *Initialise the `ROprof` structure*

All array elements of an ROprof structure are pointers; they need to be allocated before they can hold data. The subroutine ropp_io_init is provided for that purpose:

```
CALL ropp_io_init(ro_data, n_lev_1a, n_lev_1b, &
                           n_lev_2a, n_lev_2b, n_lev_2c, n_lev_2d)
```

- *Populate the data structure*

  The elements of the data structure can now be filled by the user application with the data to be written; for details of the elements on the ROprof structure see Section 2.3.

- *Write the data*

  The contents of an ROprof data structure are finally written to a file with

  ```
  CALL ropp_io_write(ro_data, filename)
  ```

  The second parameter defining the name of the file to be written is optional; if it is not specified, the ropp_io library sets the file depending on the occultation id information given in the header. See Section 2.3.6 for details. A ROPP netCDF file is written. Independent of the way the file name is specified, the routine will overwrite any existing file of the same name (but note the append keyword for multifiles; see Section 2.4.8).

2.3 Radio occultation profiles

2.3.1 Header

The header contains meta data (static, non-profile) about the occultation, including information about the processing centre and applied algorithms, and general georeferencing data. Table 2.1 shows the elements of the ROprof header and their description, along with default units and ranges.

Note that the header contains information about the georeferencing, i.e. the profile's location and time; these may not be available if only Level 1a data are contained in a data file, as the georeferencing information is determined during the Level 1a to Level 1b processing. For the ROPP data model, it is assumed that the profile's header location is formally derived as the longitude and latitude of a single tangent point adopted at some point during the occultation. It is further assumed that the centre and radius of curvature are calculated for the same tangent point. The ...%GEORef%... substructure also contains a variable describing at what point in time (relative to the start of the occultation) the georeferencing information was calculated. For all applications which interpret a radio occultation measurement as a vertical profile, the georeferencing information for the profile should be taken from the header. Note that the georeferencing method — and therefore the longitude and latitude coordinates for the profile — may differ between data providers.

The background meta data contained in the header is used to identify any *a priori* data used for the various processing steps. This may include information on any climatology used within a 'statistical optimisation' of ionospheric corrected bending angles prior to calculating refractivity, or a description of a short range Numerical Weather Prediction forecast used as first guess in a 1D–Var retrieval of geophysical parameters.

In the latter case, the forecast lead time as well as the time when the forecast is assumed to be valid should be given. If climatology data is used, the verification time should be used to indicate the month at which the climatology is considered to be valid, with all other time information set to missing values.

Finally, time stamps (given in UTC) denote the start times of the occultation as well as the processing. An overall quality parameter (see 2.3.4) and a bit field for Product Confidence Data (see 2.3.5) are also part of the header.

2.3.2 Level 1 profile data

Level 1 data of the ROPP data format consists of time and height dependent 'basic' RO parameters.

Level 1a: SNR, (excess) phases and POD data as function of time

Level 1b: bending angle as function of impact parameter

Level 1a contains the raw GNSS data after (possibly) any clock offsets and differencing is applied to the raw GNSS data as collected by the receiver. It is recommended that the samples shall be in increasing time order; all times are measured with respect to the beginning of the occultation as given in the header.

With respect to the orbit data contained in Level 1a, we recommend that all velocities are given in Earth Centred Inertial (ECI) coordinates in order to make sure that differential rotational effects have not been accidentally included by performing a transformation to the Earth Centred, Earth Fixed (ECF) reference system. Satellite positions, however, may be given in ECF coordinates to allow an easier plotting of the sub satellite points without introducing the need to convert to Earth fixed coordinates. Other choices for the orbital data reference system (e.g., both given with respect to ECI or ECF) are certainly also possible, and might be preferred by the data provider. The data provider should make sure that the corresponding meta data attributes specifying the reference system for each orbit parameter are set properly; see Section 2.4.6 for details. Any processing based on level 1a data should ensure that the coordinate systems are interpreted correctly and, if necessary, converted accordingly.

Level 1b (Table 2.3) data contains bending angles as function of impact parameter. The ROPP profile structure leaves room for four different impact parameter / bending angle pairs. One possible utilisation is to provide separate bending angle and impact parameters as derived from the L1 and L2 frequencies, and store the ionospheric corrected bending angle as the 'generic' bending angle / impact parameter pair. If a data provider chooses to provide 'statistically optimised' ionospheric bending angles, these could also be stored in the 'generic' variables. Note that the statistical optimisation process requires the use of some *a priori* data or assumption. We recommend to use an increasing impact parameter order.

2.3.3 Level 2 profile data

Level 2 data in the `ROprof` structure consist of vertical profiles of refractivity (level 2a; Table 2.3) as well as meteorological quantities (see Table 2.4).

Refractivity data are assumed to be provided in the usual N–units, such that the refractive index of air

Identifiers				
Structure element	Parameter	Description	Range	Units
...%leo_id	LEO ID	LEO ID code (4 characters). The following ID codes are currently envisaged: `META` = MetOp–A `METB` = MetOp–B `METC` = MetOp–C `COnn` = COSMIC–*nn* `CHMP` = CHAMP `GRAA` = GRACE–A `GRAB` = GRACE–B `TSRX` = TerraSAR–X `TDMX` = TanDEM–X `CNOF` = C/NOFS `SACC` = SAC–C `GPSM` = GPS/MET `OERS` = Oerstedt `EQUA` = EQUARS `SUNS` = SunSat `PAZE` = PAZ `OSAT` = OceanSat–2 `CNOF` = C/NOFS Other LEO codes may be defined in the future.	[A-Z,0-9]	
...%gns_id	GNSS ID	Letter identification (4 characters) and PRN of the occulting GNSS satellite ('`Innn`')	[A-Z,0-9]	
...%stn_id	Station ID	Ground station ID used for differencing (if any; IGS-style 4–character code)	[A-Z,0-9]	
...%occ_id	Occultation ID	Unique occultation ID; see section 2.3.6	[A-Z,0-9]	
Processing				
Structure element	Parameter	Description	Range	Units
...%FmtVersion	Format version	Exact text	`ROPP V1.1`	
...%processing_centre	Processing Centre	Text indicating processing centre (40 characters)	[A-Z,0-9]	
...%software_version	Software Version	String indicating the version of the processing software	[A-Z,0-9]	
...%pod_method	POD algorithm	Text strings (40 characters) indicating algorithms used for deriving precise orbit, excess phase / amplitude, bending angle, refractivity and meteorological data	[A-Z,0-9]	
...%phase_method	Level 1 a algorithm		[A-Z,0-9]	
...%bangle_method	Level 1 b algorithm		[A-Z,0-9]	
...%refrac_method	Level 2 a algorithm		[A-Z,0-9]	
...%meteo_method	Level 2 b, c algorithm		[A-Z,0-9]	
...%thin_method	Profile thinning algorithm (version ID)			

Table 2.1: Contents of an `ROprof` header (to be continued).

Background meta data				
Structure element	Parameter	Description	Range	Units
...%bg%source	Background source	Source of meteorological or atmospheric data used as background ("ancillary") data	[A-Z,0-9]	
...%bg%year ...%bg%month ...%bg%day ...%bg%hour ...%bg%minute	Verification time	Verification time of background data (if applicable)	1995 01 01 00 00 — 2099 12 31 23 59	
...%bg%fcperiod	F/C period	Forecast period of background data (if applicable)	0 – 24	hours
Time stamps				
Structure element	Parameter	Description	Range	Units
...%DTocc%year ...%DTocc%month ...%DTocc%day ...%DTocc%hour ...%DTocc%minute ...%DTocc%second ...%DTocc%msec	Date / time of occultation	Time stamp at **start** of occultation (UTC)	1995 01 01 00 00 00 000 — 2099 12 31 23 59 59 999	
...%DTpro%year ...%DTpro%month ...%DTpro%day ...%DTpro%hour ...%DTpro%minute ...%DTpro%second ...%DTpro%msec	Date / time of processing	Time stamp of processing (UTC)	1995 01 01 00 00 00 000 — 2099 12 31 23 59 59 999	

Table 2.1: Contents of an ROprof header (cont'd).

Georeferencing				
Structure element	Parameter	Description	Range	Units
...%GEOref%time_offset	Time since start	Time since start of occultation to the time when georeferencing data and radius of curvature are determined.	0 – 239.999	s
...%GEOref%lat	Latitude	Position of tangent point as used for georeferencing	-90 ... 90	deg
...%GEOref%lon	Longitude		-180 ... 180	deg
...%GEOref%roc	Radius of curvature	Radius of curvature value	$6.2 - 6.6 \times 10^6$	m
...%GEOref%r_coc	Centre of curvature	Centre of curvature coordinates (ECF; X, Y, Z)	± 10000	m
...%GEOref%azimuth	Line of sight	GNSS to LEO azimuth direction w.r.t. true North	0 – 360.0	deg_T
...%GEOref%undulation	Geoid undulation	Deviation of geoid (EGM–96) from the ellipsoid (WGS–84)[a]	± 150	m
Quality				
Structure element	Parameter	Description	Range	Units
...%PCD	Product confidence	Product confidence data (see Section 2.3.5)		bit flags
...%overall_qual	Data quality	Overall summary data quality	0 – 100	%
Additional Variables				
Structure element	Parameter	Description	Range	Units
...%vlist%VlistD0d	Scalar extra variables	Parameters of 0D extra variables (see Section 2.4.10)		
...%vlist%VlistD1d	Vector extra variables	Parameters of 1D extra variables (see Section 2.4.10)		
...%vlist%VlistD2d	Array extra variables	Parameters of 2D extra variables (see Section 2.4.10)		

[a] If a height h_G is expressed with respect to the EGM–96 geoid, the height h_E with respect to the WGS-84 ellipsoid is given by $h_E = h_G + U$ where U is the undulation.

Table 2.1: Contents of an ROprof header (cont'd).

Level 1a				
Structure element	Parameter	Description	Range	Units
...%Lev1a%dtime	Time since start	Time offset from time in header	-1.0 – 239.999	s
...%Lev1a%snr_L1ca	Signal to noise ratio L1 (ca–code)	Relative signal amplitude for L1 (ca–code)	0 – 50000	V/V
...%Lev1a%snr_L1p	Signal to noise ratio L1 (p–code)	Relative signal amplitude for L1 (p–code)	0 – 50000	V/V
...%Lev1a%snr_L2p	Signal to noise ratio L2 (p–code)	Relative signal amplitude for L2 (p–code)	0 – 50000	V/V
...%Lev1a%phase_L1	Excess phase L1	L1 phase corrected for geometry	±10000	m
...%Lev1a%phase_L2	Excess phase L2	L2 phase corrected for geometry	±10000	m
...%Lev1a%r_gns	Transmitter position	Earth centred Earth fixed,[a] phase centre (X, Y, Z)[b]	±43000000	m
...%Lev1a%v_gns	Transmitter velocity	Earth centred inertial,[a] phase centre (X, Y, Z)[b]	±10000	m/s
...%Lev1a%r_leo	Receiver position	Earth centred earth fixed,[a] phase centre (X, Y, Z)[b]	±10000000	m
...%Lev1a%v_leo	Receiver velocity	Earth centred inertial,[a] phase centre (X, Y, Z)[b]	±10000	m/s
...%Lev1a%phase_qual	Quality	Percentage confidence value	0 – 100	%

[a] Using the Earth Centred Fixed (ECF) and Earth Centred Inertial (ECI) reference frames for satellite positions and velocities, respectively, are the *default* settings; these can be changed, e.g. to use ECF for both positions and velocities.
[b] Position and velocity variables are 3–dimensional arrays with dimension `(/n,3/)` in Fortran.

Table 2.2: Contents of the `ROprof` Level 1a data.

Level 1b

Structure element	Parameter	Description	Range	Units
...%Lev1b%lat_tp	Latitude	Longitude and latitude w.r.t. the WGS 84 ellipsoid of the tangential point of the generic bending angle profile	± 90	deg
...%Lev1b%lon_tp	Longitude		± 180	deg
...%Lev1b%azimuth_tp	Azimuth	GNSS to LEO azimuth w.r.t. true North at tangent point	$0 - 360.0$	deg_T
...%Lev1b%impact_L1	Impact parameter (L1)	Impact parameter derived from L1 signal	$6.2 \times 10^6 - 6.6 \times 10^6$	m
...%Lev1b%impact_L2	Impact parameter (L2)	Impact parameter derived from L2	$6.2 \times 10^6 - 6.6 \times 10^6$	m
...%Lev1b%impact	Impact parameter	Impact parameter (generic)	$6.2 \times 10^6 - 6.6 \times 10^6$	m
...%Lev1b%impact_Opt	Impact parameter (Opt)	Impact parameter for optimised Bending Angles	$6.2 \times 10^6 - 6.6 \times 10^6$	m
...%Lev1b%bangle_L1	Bending angle (L1)	Bending angle derived from L1	$-0.001 - 0.1$	rad
...%Lev1b%bangle_L2	Bending angle (L2)	Bending angle derived from L2	$-0.001 - 0.1$	rad
...%Lev1b%bangle	Bending angle	Bending angle (generic)	$-0.001 - 0.1$	rad
...%Lev1b%bangle_Opt	Bending angle (Opt)	Bending angle optimised (usually smoothed) prior to performing the Abel Transform	$-0.001 - 0.1$	rad
...%Lev1b%bangle_L2_sigma	Bending angle errors (L1)	Estimated errors (one σ) of L1 bending angle values	$0 - 0.01$	rad
...%Lev1b%bangle_L2_sigma	Bending angle errors (L2)	Estimated errors (one σ) of L2 bending angle values	$0 - 0.01$	rad
...%Lev1b%bangle_sigma	Bending angle errors	Estimated errors (one σ) of bending angle values	$0 - 0.01$	rad
...%Lev1b%bangle_Opt_sigma	Bending angle errors (Opt)	Estimated errors (one σ) of optimised bending angle values	$0 - 0.01$	rad
...%Lev1b%bangle_L1_qual	Bending angle quality	Percentage confidence values for L1 bending angles	$0 - 100$	%
...%Lev1b%bangle_L2_qual	Bending angle quality	Percentage confidence values for L2 bending angles	$0 - 100$	%
...%Lev1b%bangle_qual	Bending angle quality	Percentage confidence values for bending angles	$0 - 100$	%
...%Lev1b%bangle_Opt_qual	Bending angle quality	Percentage confidence values for optimised bending angles	$0 - 100$	%

Level 2a

Structure element	Parameter	Description	Range	Units
...%Lev2a%alt_refrac	Height	Geometric height above geoid (EGM–96)	$-1000 - 150000$	m
...%Lev2a%geop_refrac	Geopotential height	Geopotential height above geoid (EGM–96)	$-1000 - 150000$	gpm
...%Lev2a%refrac	Refractivity	Derived refractivity	$0 - 500$	N–units
...%Lev2a%refrac_sigma	Refractivity error	Estimated errors (one σ) of refractivity values	$0 - 10$	N–units
...%Lev2a%refrac_qual	Refractivity quality	Percentage confidence value	$0 - 100$	%
...%Lev2a%dry_temp	Dry temperature	Derived dry temperature	$150 - 350$	K
...%Lev2a%dry_temp_sigma	Dry temperature error	Estimated errors (one σ) of dry temperature values	$0 - 50$	K
...%Lev2a%dry_temp_qual	Dry temperature quality	Percentage confidence value	$0 - 100$	%

Table 2.3: Contents of the ROprof Level 1b (bending angle) and 2a (refractivity) data.

can be calculated using

$$n = 1 + N \times 10^{-6} \ .$$

The altitude referencing for refractivity is given in two ways, one using the geometric altitude above the reference geoid (EGM–96), and the other one being geopotential height. Note that geopotential height is related to potential energy within the gravity field of the Earth. It is therefore related to a particular model of the gravity field of the Earth, which we take to be the EGM–96 model. The proper SI unit for the geopotential is m^2/s^2. For convenience, geopotential heights are by default stored as heights in 'geopotential metres', which by WMO convention are given by the geopotential divided by the standard value of gravity, namely 9.80665 m/s.

Level 2a data of the ROPP format does not include a latitude / longitude referencing for the refractivity profile. However, this information can easily be reconstructed from Level 1b data. The impact parameter p is first calculated from altitude above the geoid h using

$$p = (r_c + h) \cdot n(r_c + h)$$

where r_c is the radius of curvature (available from the header), and n denotes the refractive index as given above. Latitude and longitude coordinates of individual refractivity data points can then be calculated by interpolating the impact parameter geolocation information contained in the level 1b data.

Temperature, geopotential height, pressure, and (if available) moisture data are contained in the level 2b part (Table 2.4). Depending on the data provider, these may consist of the results of a 1D–Var retrieval of atmospheric parameters from level 1b or level 2a data. Other providers might provide dry temperatures and pressures, or specific humidity and the temperature used to derive the latter from refractivity, or some mixture of the latter.

1D–Var retrievals may also provide some information on surface (or another reference) pressure which, together with data on the geopotential height of the (model) surface, is contained in the Level 2c data (see Table 2.4). The remarks on issues with geopotential height made above also apply to the surface geopotential height. In addition, there might be inconsistencies in the way surface geopotential heights have been calculated from Digital Elevation Model data; for example, surface orography (i.e. elevation of the surface above sea level) is often taken instead of a properly calculated geopotential. More information on the details of the corresponding conversions should be made available by the data provider.

If the vertical level structure of retrieved or background meteorological profiles is based on NWP forecast model levels, the surface pressure and geopotential height will also be required to calculate all information on the actual model levels. An example are the terrain following hybrid vertical coordinates as used by ECMWF, or the terrain following altitude coordinates as used in the Met Office system. It is recommended to store profiles in ascending height order.

2.3.4 Quality

The ROPP–format uses the concept of 'Quality' as a parameter in all of the sublevels and a 'Summary Quality' value in the header. In `BUFR` (see [RD.1]), the term 'Percentage Confidence' is equivalent. The

Level 2b

Structure element	Parameter	Description	Range	Units
...%Lev2b%geop	Geopotential height	Geopotential height above geoid (EGM–96)	$-1000 - 100000$	gpm
...%Lev2b%geop_sigma	Geopotential height error	Estimated error (one σ) of geopotential heights	$0 - 500$	gpm
...%Lev2b%press	Pressure	Retrieved pressure	$0.0001 - 1100$	hPa
...%Lev2b%press_sigma	Pressure error	Estimated error (one σ) of retrieved pressure	$0 - 5$	hPa
...%Lev2b%temp	Temperature	Retrieved temperature	$150 - 350$	K
...%Lev2b%temp_sigma	Temperature error	Estimated error (one σ) of retrieved temperature	$0 - 5$	K
...%Lev2b%shum	Specific humidity	Retrieved specific humidity	$0 - 50$	g / kg
...%Lev2b%shum_sigma	Specific humidity error	Estimated error (one σ) of retrieved specific humidity	$0 - 5$	g / kg
...%Lev2b%meteo_qual	Quality	Overall percentage confidence value	$0 - 100$	%

Level 2c

Structure element	Parameter	Description	Range	Units
...%*Lev2c%lat_2d*	Latitude position	Latitude position (*ROprof2d structure only*)	$-90 - 90$	deg
...%*Lev2c%lon_2d*	Longitude position	Longitude position (*ROprof2d structure only*)	$-180 - 180$	deg
...%*Lev2c%dtheta*	Angle	Angle between profiles (*ROprof2d structure only*)	$0 - \pi$	rad
...%Lev2c%geop_sfc	Geopotential height	Geopotential height of surface above geoid (EGM–96)	$-1000 - 10000$	gpm
...%Lev2c%press_sfc	Surface pressure	Retrieved surface (or reference) pressure	$250 - 1100$	hPa
...%Lev2c%press_sfc_sigma	Surface pressure error	Estimated error (one σ) of retrieved surface pressure	$0 - 5$	hPa
...%Lev2c%press_sfc_qual	Quality	Percentage confidence value	$0 - 100$	%

Level 2d

Structure element	Parameter	Description	Range	Units
...%Lev2d%level_type	level type	Level type; currently, only one of HYBRID ECMWF, ECMWF HYBRID, HYBRID, ECMWF or METOFFICE are currently supported.		
...%Lev2d%level_coeff_a	α coefficients	Level coefficients α (hybrid vertical levels only)	0 – 2000	hPa
...%Lev2d%level_coeff_b	β coefficients	Level coefficients β (hybrid vertical levels only)	0 – 2	n/a

Table 2.4: Contents of the ROprof (*ROprof2d*) Level 2b (free atmospheric parameters), 2c (surface) and 2d (level coefficients) data.

meaning of this parameter depends on the context and to a large degree on the specific processing algorithms used (and hence its derivation has to be defined by the data processing centre), but we may give some hints as to the intent.

At a user-level, Quality (or 'Percentage Confidence') can be used as a first indicator of whether a whole profile or a particular datum point is likely to be useful. In NWP, the concept of 'probability of gross error' is often used for quality control in variational assimilation systems. The objective is to reject observations with extreme errors, particularly if they do not lie within the normal (assumed) Gaussian error distribution.

In the simplest case, Quality (Q) could be derived from the estimated error(s) in the associated 'observed' parameter. Assuming an estimated observed error variance O and an *a priori* variance E, then

$$Q = 100e^{-O/E}$$

would be a suitable value. Errors for several observables may be combined to form an overall probability value.

Other Quality values might be derived from other QC information — for instance the Quality for bending angles might be nominally derived as above, but assigned a lower value (even zero) if loss–of–lock (fly wheeling) condition is detected. In this case, the observables may have apparently sensible values but there is little or no confidence in them. The end–user can then make the decision on whether to use such data or not.

2.3.5 Product Confidence Data (PCD)

Product confidence data (PCD) (in BUFR: Quality flags for RO) are held as sets of bit flags within a single 32-bit / 4-byte integer value and indicate various product quality states or tests. Bits are numbered from 1. This format allows for 15 "information" bits (values 0 – 32727); setting bit 16 (values 32768 – 65535) indicates missing PCD data (i.e., none of the bit settings are valid). If some items are valid and others are not, invalid items should have their assigned bit clear (zero). The module `ropp_io` defines several integer parameters that are intended to be used with Fortran's internal `BTEST`, `IBCLR` and `IBSET` bit functions (see below). The defined bits and their Fortran parameter names are shown in Table 2.5. These bit flags are designed to be generic in that they can be applied to any mission and any processing scheme. Certain processing dependence is inevitable (e.g., meteorological quantities might be derived directly from bending angles, but it is likely that refractivity will be derived anyway, and the flagging can be mapped to the actual processing). The Software ID parameter and Processing Centre ID together can be used to trace back the detail of the particular processing algorithms in order to interpret the specific meaning of these flags.

The Fortran parameters shown in Table 2.5 become available after loading the `ropp_io_types` module in the variable declaration part of a Fortran subroutine or function. Testing the PCD flags can be accomplished with Fortran's `BTEST` function. To test if the overall data quality is nominal, for example, a construction like

```
IF (BTEST(ro_data%PCD, PCD_summary)) THEN
   ! ... do what needs to be done for non-nominal ...
```

Bit	Variable	Description	Meaning if	
			unset (0)	set (1)
1	PCD_summary	Quality	nominal	non–nominal
2	PCD_offline	Product type	NRT	off line
3	PCD_rising	Occultation type	setting	rising
4	PCD_phase	Excess phase processing	nominal	non–nominal
5	PCD_bangle	Bending angle processing	nominal	non–nominal
6	PCD_refrac	Refractivity processing	nominal	non–nominal
7	PCD_met	Meteorological processing	nominal	non–nominal
8	PCD_open_loop	Open Loop	not used	used
9	PCD_reflection	Surface reflections detected	no	yes
10	PCD_l2_signal	L2P or L2C GNSS signal used	L2P	L2C
11	PCD_reserved_11	Reserved		
12	PCD_reserved_12	Reserved		
13	PCD_reserved_13	Reserved		
14	PCD_bg	Background profile	nominal	non–nominal
15	PCD_occultation	Profile type	observed	background
16	PCD_missing	PCD missing; bits 1–15. . .	valid	invalid

Table 2.5: Product confidence data definition. PCD_summary is a summary bit which shall be set if any of Bits 4,5,6,7 or 14 is set. Note that the PCD_* variables become available by USE'ing the module ropp_io_types.

```
ELSE
   !       ... and for nominal data quality.
END IF
```

allows the user to take appropriate action in a Fortran program, where ro_data is a structure of type ROprof. If using the bit number directly, rather than the ROPP parameter name, remember that Fortran BTEST numbers bits from zero.

When writing data to a ROPP data file, the user needs to set the PCD bits explicitly. This is required because initialising an ROprof structure sets its PCD value to 65535, indicating that the PCD is currently invalid. If only a few bits need to be set, the quickest way is to set the PCD to zero (all bits unset), and then to set the required bits. As an example, a profile retrieved off–line from a rising radio occultation measurement for which the overall quality as well as processing steps are nominal requires both the PCD_offline and the PCD_rising flags to be set. This can be achieved by

```
ro_data%PCD = 0
ro_data%PCD = IBSET(ro_data%PCD, PCD_offline)
ro_data%PCD = IBSET(ro_data%PCD, PCD_rising)
```

When storing simulated or background data in a ROPP file, the PCD_occultation flag must be set to indicate that the file does not contain an actual occultation measurement:

```
ro_data%PCD = IBSET(ro_data%PCD, PCD_occultation) ! Background data, NOT an occultation
 ...                                              !       measurement!
```

This is particularly important as the setting of the Occultation ID (and therefore the generation of automated file names during `ropp_io_write`, see the following section) relies on proper setting of the `PCD_occultation` bit.

The summary bit `PCD_summary` shall be set if any of the other 'nominal' bits are set:

```
IF (BTEST(ro_data%PCD, PCD_phase)  .OR.
    BTEST(ro_data%PCD, PCD_bangle) .OR.
    BTEST(ro_data%PCD, PCD_refrac) .OR.
    BTEST(ro_data%PCD, PCD_met)    .OR.
    BTEST(ro_data%PCD, PCD_bg)) THEN
  ro_data%PCD = IBSET(ro_data%PCD, PCD_summary)
```

2.3.6 Occultation IDs and file names

The occultation ID in the header of a `ROprof` structure should be a unique identifier for a particular occultation event that could be used for cataloguing, filtering etc. This identifier could also be used as a file naming convention. In fact, writing data using the `ropp_write` interface without specifying a filename will result in the creation of a file with a name derived from the occultation ID.

It is recommended that the following format be used for the occultation ID field in the header of ROPP-format files, and that this also forms the base for the file names:

```
tt_yyyymmddhhmmss_llll_gggg_pppp
```

where

- `tt` denotes the type of the profile data (e.g., '`OC`' or '`BG`' for occultation and background data, respectively);
- `yyyymmdd` is the date (year, month, day) of the occultation event;
- `hhmmss` is the time (hour, minute, second) of the occultation event;
- `llll`, gggg and pppp denote LEO, GNSS and Processing Centre ID's.

File names, regardless whether generated from an occultation ID or otherwise, should have an extension indicating their technical file type. It is recommended to be '`.nc`' for the netCDF–based version of ROPP data files.

For convenience, the routine `ropp_io_occid` generates an occultation ID from the header information, and sets the `occ_id` parameter in the header to a value consistent with the above recommendations. This requires that all other elements of the header are set up properly:

```
CALL ropp_io_occid(ro_data) !  Sets ro_data%occ_id
```

The write routine `ropp_io_write` will auto–generate a file name from the occultation ID if no explicit file is given by appending the occultation ID in the header with an extension appropriate for the file type. If the occultation ID is not available (i.e., if `...%occ_id` is set to 'UNKNOWN' or empty), a suitable ID will be generated using the above convention.

2.4 Advanced use

2.4.1 Initialisation

Initialisation of `ropp_io` data types includes allocating pointers as well as setting the relevant integer elements of (sub–)structures to the number of elements in these arrays. The latter is of particular importance, as the software uses the counters to determine if a subsection contains data or not; for example, if `...%Npoints` equals zero, no data for the respective level is written. As already mentioned in Section 2.2, the easiest way to initialise a data structure for an RO profile is to call

```
CALL ropp_io_init(ro_data, n_lev_1a, n_lev_1b, &
                           n_lev_2a, n_lev_2b, n_lev_2c, n_lev_2d)
```

where the parameters `n_lev_1a`, `n_lev_1b`, `n_lev_2a`, `n_lev_2b`, `n_lev_2c`, and `n_lev_2d` denote the number of elements in the arrays for each sublevel. Setting one or more of these numbers to zero or a negative number indicates that the corresponding sub–level does not include any data; this will also free all memory associated with these sub–parts should they have been used earlier.

Depending on circumstances, it might be more appropriate to initialise the relevant parts of an `ROprof` separately. This is possible by calling `ropp_io_init` with that part of the structure, and a single mandatory integer number defining the number of elements for that subpart. Examples are

```
CALL ropp_io_init(ro_data%Lev1a, n_lev_1a)
CALL ropp_io_init(ro_data%Lev1b, n_lev_1b)
CALL ropp_io_init(ro_data%Lev2a, n_lev_2a)
CALL ropp_io_init(ro_data%Lev2b, n_lev_2b)
CALL ropp_io_init(ro_data%Lev2c, n_lev_2c)
CALL ropp_io_init(ro_data%Lev2d, n_lev_2d)
```

The above subroutine calls initialise the `Npoints` elements of each subpart to the appropriate number, allocates all arrays within the `ROprof` structure, and fills the allocated arrays with predefined "missing values". Note that the scalar surface part (Level 2c) does not require a specific initialisation of arrays. However, for surface data the number of Level 2c elements must be set to one (the only allowed value apart from 0, indicating that no surface pressure data is contained in the profile data). While all allocations could alternatively been undertaken by the user, calling the provided subroutines for this task will ensure consistency of the data.

2.4.2 Freeing memory

Occasionally, it will be required to free (deallocate) all memory in a given data structure. The call

```
CALL ropp_io_free(ro_data)
```

will deallocate all arrays from all sublevels, and set the corresponding `...%Npoints` variables to zero. It will also set all scalar variables (including all information in the header) to default missing values.

The individual sublevels of an ROprof structure can be freed with one of

```
CALL ropp_io_free(ro_data%Lev1a)
CALL ropp_io_free(ro_data%Lev1b)
CALL ropp_io_free(ro_data%Lev2a)
CALL ropp_io_free(ro_data%Lev2b)
CALL ropp_io_free(ro_data%Lev2c)
CALL ropp_io_free(ro_data%Lev2d)
```

Note, however, that the above sequence is *not* equivalent to freeing the same data structure at once, as the individual calls to ropp_io_free will still leave the header untouched.

The main intended use of the subroutine is to allow the re–use of the same variable for different profiles, possibly with a different number of data values. Therefore, meta–data describing units and valid ranges are *not* reset, in order to keep settings the user might have changed from the default. Also note that the current interface only allows deallocating scalar data structures; if an array of structures is to be freed, a loop construction as in

```
do i = 1, size(ro_profiles)
   CALL ropp_io_free(ro_profiles(i))
end do
```

is required.

2.4.3 Unit conversion

Data files written by the ropp_io library contain data in standard physical units which are documented in the units attribute for each variable. These standard units are listed in Tables 2.1, 2.2, 2.3 and 2.4.

A user application may require the data in different (compatible) physical units. Therefore, the ROprof structure contains variables specifying the units of its elements. For details, see the reference documentation. After having been declared, the unit specifying variables contain the standard set of units. By changing the standard values of the unit specification to the desired ones *before* reading data from an ROPP data file with ropp_io_read, a user can request the conversion of the data in the file to the desired physical units. For example, to specify the altitude for refractivity data in units of kilometres, and the geopotential in units of m^2/s^2,

```
ro_data%Lev2a%units%alt_refrac  = 'km'
ro_data%Lev2a%units%geop_refrac = 'm^2/s^2'
 ...
CALL ropp_io_read(ro_data, filename)
```

The ranges (maximum and minimum valid data values for each parameter) are also converted to the new units, so range–checking in the new units remains consistent. The mechanism also works when the ROPP

data file contains units different from the standard units, as long as the `units` and `range` attributes in the netCDF based ROPP data file are consistent with the actual physical units of the data. All unit conversions within `ropp_io` are performed using the `unitconvert` utility library provided in the `ropp_utils` module (see Appendix A).

Similarly, a program might internally produce variables in physical units different from ROPP's standard units. The `ropp_io` library will take care of the necessary unit conversions if the respective unit specifying variables of the `ROprof` structure are set to the ones used by the program *before* the data is written by `ropp_io_write`. For example, consider a program which produces a retrieval of meteorological quantities with temperature units of degree Celsius, pressure units of Pa, specific humidity units of kg/kg, and geopotential height units of geopotential kilometre. The same units are used for the estimated uncertainties. Exploiting the conversion capability of the `ropp_io` library would require

```
ro_data%Lev2b%units%press            = 'Pa'
ro_data%Lev2b%units%press_sigma      = 'Pa'
ro_data%Lev2b%units%temp             = 'degreeC'
ro_data%Lev2b%units%temp_sigma       = 'degreeC'
ro_data%Lev2b%units%shum             = 'kg/kg'
ro_data%Lev2b%units%shum_sigma       = 'kg/kg'
ro_data%Lev2b%units%geop             = 'geopotential km'
ro_data%Lev2b%units%geop_sigma       = 'geopotential km'
ro_data%Lev2c%units%press_sfc        = 'Pa'
ro_data%Lev2c%units%press_sfc_sigma  = 'Pa'
ro_data%Lev2c%units%geop_sfc         = 'geopotential km'
 ...
CALL ropp_io_write(ro_data, filename)
```

2.4.4 Range checking

The `ropp_io` module contains a range checking routine which checks every parameter in the `ROprof` structure — including header parameters and strings as well as numeric values — for validity. Numeric parameters have attributes of range (minimum and maximum valid limits) against which data values are tested. Invalid (out-of-range) points are set to 'missing'. String parameters are tested against a valid character set or set of valid options. Further, profile basic coordinates (time for Level 1a, impact parameter, geometric or geopotential heights for Levels 1b, 2a and 2b, respectively) are tested for validity; invalid points are removed from the profile, and the `Npoints` value set to the number of valid levels in the profile. Note that this procedure does *not* remove levels with invalid observed data points (bending angles, etc), only those with invalid basic coordinates (ie, independent variables).

This validity checking is accomplished by calling:

```
CALL ropp_io_rangecheck(ro_data)
```

This operation is performed within the ROPP thinner routine `ropp_io_thin()` and may be performed

on read or write by setting the optional `ranchk` keyword argument in the call to `ropp_io_read()` or `ropp_io_write()`.

2.4.5 Missing data

Special values are reserved to indicate that data are 'missing' or invalid; for most parameters this is set to the 'missing data flag value' defined by `ropp_MDFV` (some parameters are better set to zero `ropp_ZERO`). Currently, this special value is `-99999000.0`, but this may change at a future release.

Having done a range check operation, the parameter 'missing data test value' `ropp_MDTV` can be used to test for missing data thus:

```
IF ( ro_data%georef%undulation < ropp_MDTV ) THEN
    ! ... can't apply geoid/ellipsoid conversion ...
ELSE
    ! ... apply geoid/ellipsoid conversion ...
END IF
```

These three missing data related parameters are defined in the `ropp_utils` module. The `ropp_MDFV` value is written to output files as the global attribute `_FillValue`. Note that no checking of data is performed against user-defined `_FillValue` attributes on reading an input file in the current release.

Parameters set to MDTV are exempt from units conversion, so will always retain this special value for consistent testing.

2.4.6 Reference systems for POD data

Precise Orbit Determination (POD) data are given with respect to reference terrestrial or stellar reference systems; typical ones are "Earth Centred, Earth Fixed" (ECF) or "Earth Centred Inertial" (ECI) coordinates. Another coordinate system sometimes used in satellite Geodesy is the "True of Date System" (TDS). The `ROprof` data structure provides a means for data producers to specify which reference system is used for the provided POD data; the variables

```
ro_data%Lev1a%reference_frame%r_gns  = 'ECF' | 'ECI' | ...
ro_data%Lev1a%reference_frame%v_gns  =          ...
ro_data%Lev1a%reference_frame%r_leo  =          ...
ro_data%Lev1a%reference_frame%v_leo  =          ...
```

are intended to be used by data producers to specify which reference systems is used. Note that the default settings are

```
ro_data%Lev1a%reference_frame%r_gns  = 'ECF'
ro_data%Lev1a%reference_frame%v_gns  = 'ECI'
ro_data%Lev1a%reference_frame%r_leo  = 'ECF'
ro_data%Lev1a%reference_frame%v_leo  = 'ECI'
```

for the reasons discussed in Section 2.3.

2.4.7 Output precision

ROPP data files contain most floating point variables as single precision. The exception are variables holding internal time, bending angle, heights and POD data which are stored as `double` to cover the possible physical range of the data to sufficient numerical precision. The default behaviour can be changed by setting the optional `output_precision` argument to the `ropp_io_write` routine, as in

```
CALL ropp_io_write(ro_data, filename, output_precision='double')
```

This will cause all floating point variables to be written as double precision netCDF variables. Note that double precision output may increase the size of data files considerably.

2.4.8 Multi–profile data files

The examples presented so far all assumed that an individual ROPP data file contains only a single profile. It is also possible to store multiple profiles in one data file. In the following, we will call such 'multi–profile' data file a *multifile*, in contrast to *singlefiles* holding exactly one profile.

A major (but only) restriction for multifiles is that the number of data points in any given level must be equal for all profiles. Thus, it is possible to combine retrievals (say, levels 2b, c, and d) of atmospheric parameters on a fixed set of altitude or pressure levels into a multifile. Level 1a data from several profiles, each having a different number of raw data samples, can only be merged into a multifile if the individual data arrays are padded with missing values to exhibit the same number of elements. This means that the first profile must be the (joint) largest. The latter comes at the expense of increased memory requirements for the data.

There are several ways to read and write multifiles, all described in the following subsections. An example for the use of the multifile interface as described above can be found in the source code of the `ropp2ropp` program (see Section 2.6 on page 35), which is part of `ropp_io` and fully supports copying, merging and splitting multifiles.

Sequential read

In a database–like terminology, the profiles in a multifile correspond to one of many records in the netCDF file; a singlefile is simply a data file with a single record. Assuming the following variable declarations,

```
USE ropp_io
TYPE(ROprof)         :: ro_data
INTEGER              :: n_prof, i
CHARACTER (LEN=...) :: filename = '...'
```

the number of records (i.e., profiles) is available through

```
n_prof = ropp_io_nrec(filename)
```

The *i*–th profile can be read by

```
CALL ropp_io_read(ro_data, filename, rec=i)
```

and the entire data set could be processed with

```
n_prof = ropp_io_nrec(filename)
   ...
DO i = 1, n_prof
   CALL ropp_io_read(ro_data, filename, rec=i)
   ! Process the data as required...
END DO
```

If `rec` is not specified, the first record present in the data file is read.

Read–at–once

Data contained in a multifile can also be ingested in a single `ropp_io_read` call. This requires that the variable holding the set of profiles is declared as a 1–dimensional *pointer* of type `ROprof`, i.e.

```
TYPE(ROprof), DIMENSION(:), POINTER :: ro_data_set => NULL()
```

where `ro_data_set` is explicitly nullified in the declaration. Reading the entire data set at once then only requires

```
CALL ropp_io_read(ro_data_set, filename)
```

The subroutine `ropp_io_read` will allocate sufficient amounts of memory to hold all profiles in `filename`. The elements of the `ROdata` structure for individual profiles can be accessed via

```
ro_data_set(i)%...
```

and the number of profiles read is given by

```
n_prof = SIZE(ro_data_set)
```

Sequential write

The default write mode is to overwrite already existing data files. Instead, data can be appended to an already existing data file with

```
CALL ropp_io_write(ro_data, filename, append=.true.)
```

Note that a new file will still be created if the file `filename` does not exist. In effect, this allows the user to sequentially add profiles to an existing data file. Similar to `ropp_io_read`, the optional parameter `rec` is also supported: adding profile data as *i*–th record in an existing data file can be achieved with

```
CALL ropp_io_write(ro_data, filename, rec=i, append=.true.)
```

Note that the `append` argument needs to be given as well.

Write–at–once

Similar to read–at–once, an entire data set of profiles can be written into an ROPP multifile in one go by declaring

```
TYPE(ROprof), DIMENSION(:), POINTER :: ro_data_set => NULL()
```

After `ro_data_set` has been allocated and filled with data, the entire data set can be written with

```
CALL ropp_io_write(ro_data_set, filename)
```

The `append` and `rec` arguments can be used in a similar way as in sequential writing. For example,

```
CALL ropp_io_write(ro_data_set, filename, append=.true.)
```

will append the `ro_data_set` to an already existing data file (instead of overwriting it), while

```
CALL ropp_io_write(ro_data_set, filename, rec=i, append=.true.)
```

will append the data set to an already existing file, *starting* at the *i*–th record.

NetCDF operators (`nco`)

The netCDF operators (`nco`)[3] are a versatile set of Unix–style command line tools to manipulate netCDF data files. While `ropp_io` does not require them to be available, we highly recommend `nco` as generic tool set for netCDF data files. The operator `ncrcat` can be used to merge several ROPP data file into a single one (i.e., to create a multifile) with

```
ncrcat <file_1.nc> <file_2.nc> ... <file_n.nc> -o <multifile.nc>
```

As an example, consider that a directory `daily` contains ROPP data files from a single day. Combining all of them into a single data file `today.nc` would require

```
ncrcat daily/* -o today.nc
```

Individual profiles (or ranges of profiles) can be extracted from a multifile using the `ncks` operator exploiting its `-d` option:

[3] See http://nco.sourceforge.net.

```
ncks -d dim_unlim,<start>[,<end>] <multifile.nc> -o <resultfile.nc>
```

where start and end denote the start and end indexes of the range of profiles to be extracted. Note that, by default, dimension indexes for ncks are zero based as in C, i.e. 0 corresponds to the first profile, 1 to the second etc. (Fortran-like indexing — 1, 2, etc — can be invoked by using the -F option in most routines.) To extract the second profile from a file named mropp_test.nc, therefore, the appropriate ncks command would be

```
ncks -d dim_unlim,1 mropp_test.nc -o profile_2.nc
```

Note that ncks also supports an index convention similar to Fortran with the -F option. Thus, the command

```
ncks -F -d dim_unlim,2 mropp_test.nc -o profile_2.nc
```

yields the same result as the previous example.

In general, the use of ncrcat and ncks for merging and splitting ROPP files provides the same data content as manually created single- and multifiles. Differences will occur due to different entries in the global history attribute of the generated data set. For the nco operators, adding information to the history attribute can be disabled with -h; the results should then be identical to those obtained with, e.g., ropp2ropp.

2.4.9 Arrays of structures

The previous section introduced arrays of structures of type ROprof, e.g. variables declared like

```
TYPE(ROprof), DIMENSION(:), POINTER :: ro_data_set => NULL()
```

It is important to remember that two allocation or deallocation steps have to be undertaken when dealing with structure arrays: First, an array of structures needs to be allocated; then, the elements of the individual structure elements need to be initialised. Similarly, freeing the memory taken up by an array of structures such as ro_data_set in the above examples requires that the allocated memory of the individual structures is free before the array of structures itself is finally deallocated. A complete memory allocation and deallocation cycle would look like the following example:

```
! 1. Allocate an array of structures
! -----------------------------------
  ALLOCATE(ro_data_set(n_prof))

! 2. Initialise individual structures
! -----------------------------------
  DO i = 1, n_prof
     CALL ropp_io_init(ro_data_set(i), ...)
  END DO
```

```
! 3. Fill individual structures
! ------------------------------
! Individual code goes here...

! 4. Free individual structures
! ------------------------------
  DO i = 1, n_prof
     CALL ropp_io_free(ro_data_set(i))
  END DO

! 5. Deallocate array of structures
! ---------------------------------
  DEALLOCATE(ro_data_set)
```

2.4.10 Extending the `ropp_io` data type

NetCDF data is ultimately referenced by its variable name; additional data items can be added at any time as long as the added data uses variable names different from the already stored data. The presence of such additional data does not compromise the ability of existing readers to ingest the variables they know of. Thus, netCDF (and therefore ROPP) data files can easily be extended beyond the core set of variables as defined in the original `ROprof` structure, simply by adding additional variables to the data file. The only restriction is that the netCDF names of the additional variables may not coincide with the netCDF names used for the ROPP core variables.

In order to facilitate a simple way to write such additional (or non–core) variables, `ropp_io` offers the ability to extend the `ROprof` data type: prior to writing, additional variables can be "appended" to an existing profile structure by specifying the data to be written and attributes required for the later storage in the netCDF data file. The required functionality is provided by the `ropp_io_addvar` routine. The following example shows how to add the value of the cost function of a 1D-Var retrieval and the Probability of Gross Error.

```
USE typesizes, only: dp => EightByteReal
USE ropp_io
TYPE(ROprof) :: ro_data
REAL(wp)     :: cost
REAL(wp)     :: pge
   ...
CALL ropp_io_addvar(ro_data,                                          &
                    name      = "J",                                  &
                    long_name = "Cost function value at convergence", &
                    units     = "",                                   &
                    range     = (/ 0.0_dp, 9999.0_dp/),               &
```

```
                        data      = cost)
  CALL ropp_io_addvar(ro_data,                                  &
                        name      = "pge",                      &
                        long_name = "Probability of gross error", &
                        units     = "%",                        &
                        range     = (/ 0.0_dp, 100.0_dp/),      &
                        data      = pge)
```

where `cost` and `pge` are the program's Fortran variables holding the respective values. If `ro_data` is later written to a netCDF data file with `ropp_io_write`, additional variables named `J` and `pge` will be created in the netCDF and filled with the contents of the corresponding `data`. The netCDF standard attributes for `long_name`, `units` and `valid_range` are taken from the `long_name`, `units` and `range` arguments to the `ropp_io_addvar` routine, respectively.

The shape of the `data` arguments defines the shape of the variables written to the netCDF data file: if the argument to `data` is scalar (like in the above example), it is assumed that each profile within the data file obtains an additional scalar value. Similarly, one and two dimensional `data` arguments will create vector or matrix shaped additional variables. Note that the current implementation only supports double precision scalar, one or two–dimensional floating point data.

If non-core variables are present in a netCDF file to be read in with `ropp_io_read` the `ROprof` data type is similarly extended, so that all variables in a netCDF file are preserved on read/write using `ropp_io`.

Additional variables are stored as a linked list in the `ROprof%vlist` substructure. The idea is shown in Table 2.6.

2.4.11 Alternative ways to read ROPP files

Since ROPP data files are regular netCDF files, they can be read with any software or application that is able to read netCDF data files; all that is required is the knowledge about the variables in the netCDF data file and their meaning. Apart from the native netCDF interface supporting C, C++, Fortran 77 and Fortran 90/95, graphics systems like IDL, PV Wave, Matlab, R or NCAR Graphic's ncl can read (and write) netCDF data. There are also APIs for other programming languages like Perl, Python or Java. For an exhaustive list on programming environment and graphical applications that do support netCDF, see the netCDF website (Unidata).

A particularly simple interface for reading (and also writing) netCDF data from Fortran is the freely available `ncdf90` high level interface to netCDF. In fact, the `ropp_io` library is built on top of `ncdf90`, and contains most of its source code. Thus, all variables contained in an ROPP data file can also be read (from Fortran) using the `ncdf` interface which is part of `ropp_io`. Compared to the standard read method using `ropp_io_read()` this can yield performance improvements if only a small subset of the data is required by the application. For example, to read refractivity and its geopotential height from the Level 2a part, along with its coordinates and the date and time of the occultation, the following would suffice (the example assumes that the Fortran variables `refrac` and `geop` are one dimensional real pointers):

Additional variables				
Structure element	Parameter	Description	Range	Units
...%vlist%VlistD0d%name	Name	Name of 1st 0D extra variable	[A-Z,0-9]	
...%vlist%VlistD0d%long_name	Long name	Long name of 1st 0D extra variable	[A-Z,0-9]	
...%vlist%VlistD0d%units	Units	Units of 1st 0D extra variable	[A-Z,0-9]	
...%vlist%VlistD0d%range	Range	Range of 1st 0D extra variable		
...%vlist%VlistD0d%DATA	Data	Value of 1st 0D extra variable		
...%vlist%VlistD0d%next%name (etc)	Name (etc)	Name (etc) of 2nd 0D extra variable	[A-Z,0-9]	
...%vlist%VlistD0d%next%next%name (etc)	Name (etc)	Name (etc) of 3rd 0D extra variable	[A-Z,0-9]	
...%vlist%VlistD1d%name	Name	Name of 1st 1D extra variable	[A-Z,0-9]	
...%vlist%VlistD1d%long_name	Long name	Long name of 1st 1D extra variable	[A-Z,0-9]	
...%vlist%VlistD1d%units	Units	Units of 1st 1D extra variable	[A-Z,0-9]	
...%vlist%VlistD1d%range	Range	Range of 1st 1D extra variable		
...%vlist%VlistD1d%DATA	Data	Value of 1st 1D extra variable		
...%vlist%VlistD1d%next%name (etc)	Name (etc)	Name (etc) of 2nd 1D extra variable	[A-Z,0-9]	
...%vlist%VlistD1d%next%next%name (etc)	Name (etc)	Name (etc) of 3rd 1D extra variable	[A-Z,0-9]	
...%vlist%VlistD2d%name	Name	Name of 1st 2D extra variable	[A-Z,0-9]	
...%vlist%VlistD2d%long_name	Long name	Long name of 1st 2D extra variable	[A-Z,0-9]	
...%vlist%VlistD2d%units	Units	Units of 1st 2D extra variable	[A-Z,0-9]	
...%vlist%VlistD2d%range	Range	Range of 1st 2D extra variable		
...%vlist%VlistD2d%DATA	Data	Value of 1st 2D extra variable		
...%vlist%VlistD2d%next%name (etc)	Name (etc)	Name (etc) of 2nd 2D extra variable	[A-Z,0-9]	
...%vlist%VlistD2d%next%next%name (etc)	Name (etc)	Name (etc) of 3rd 2D extra variable	[A-Z,0-9]	

Table 2.6: Contents of the ROprof additional variables.

```
USE ncdf
CALL ncdf_open(filename)
   ...
CALL ncdf_getvar('lon', lon,  rec=i)
CALL ncdf_getvar('lat', lat,  rec=i)
CALL ncdf_getvar('js',  time, rec=i, units='seconds since 2004-10-01 12:00')
   ...
CALL ncdf_getvar_alloc('refrac',      refrac, rec=i)
CALL ncdf_getvar_alloc('geop_refrac', geop,   rec=i , units='m^2/s^2')
   ...
CALL ncdf_close()
```

In a similar way, the J and pge data added to a ROPP data file (as described in the previous section) might be read using

```
USE ncdf
CALL ncdf_open(filename)
   ...
CALL ncdf_getvar('J', J, rec=i)
   ...
CALL ncdf_getvar_alloc('pge', pge, rec=i)
   ...
CALL ncdf_close()
```

where J is a scalar, and pge a one–dimensional double precision pointer variable. Note that the specification of the `rec` argument is mandatory in the above example, as precisely one profile is to be read. The variable `i` denotes the record number, and might be 1 for singlefiles.

For details on the various `ncdf` routines, see the ROPP I/O Reference Manual.

2.4.12 NcView and NcBrowse

NcView[4] is a freely available visual browser for netCDF format files. Users may find this package useful for a quick and easy way to look at the contents of a netCDF file. NcView runs on unix/Linux under the X-windows system. NcBrowse[5] is an alternative graphical netCDF file browser, and being written in Java is a cross-platform application, so is particularly useful on MS Windows and Mac OS-X.

2.5 Plotting ROPP data

In the following sections, we give a few hints on how to read and plot data in the ROPP format into various graphics programs that support direct input of netCDF data. For an exhaustive list on graphical

[4] See http://meteora.ucsd.edu/~pierce/ncview_home_page.html.
[5] See http://www.epic.noaa.gov/java/ncBrowse.

applications that support netCDF, see the netCDF website (Unidata). The `ropp_io/contrib` directory of the source code distribution contains some tools for data ingestion for `IDL` and `PV-Wave` (which are not fully supported, however). Note that the plots generated by the code in the following examples are not intended to be publication quality figures. They are provided for illustration purposes only.

2.5.1 IDL

The commercial graphics language IDL[6] (Interactive Data Language) provides native read and write access to netCDF data files. The directory `ropp_io/contrib/idl` contains the IDL procedure `ncdf_getvar.pro` to read data from generic netCDF data files. The following sequence of commands would create a simple plot of a retrieved temperature profile (note that the variable `ncdf` is assumed to be set to the file name of the ROPP data file):

```
ncdf_getvar, ncdf, 'press', press ; Read pressure.
ncdf_getvar, ncdf, 'temp', temp   ; Read temperature.

temp = temp - 273.15              ; Convert to degree C.

plot, temp, press, xtitle = "Temperature (K)", xrange = [-90., 30.], $
                   ytitle = "Pressure (hPa)", yrange = [1000., 0.1], /ylog
```

Note that `ncdf_getvar` requires the netCDF variable names; it has no knowledge of the `ROprof` data structure. For the full documentation of `ncdf_getvar`, run

```
ncdf_getvar, /help
```

at the IDL prompt or read the full documentation in the routine's header. In order to run `ncdf_getvar` from the IDL command line prompt, the file `ncdf_getvar.pro` must have been installed somewhere in IDL's search path, or the current directory; see the IDL documentation for details. Note that the routines in the `contrib` directory are provided for convenience only, and are not fully supported by the ROM SAF.

2.5.2 PV-Wave

PV–Wave[7] is another proprietary graphics system, originally derived from an early version of IDL. Reading netCDF data files is supported as part of the HDF interface, although the latter is only poorly documented. The `ropp_io/contrib/wave` directory contains some routines that mimic IDL's read interface to netCDF data files, and also the routine `ncdf_getvar`. Creating a simple plot is very similar to IDL; the commands corresponding to the above example would be

```
ncdf_getvar, ncdf, 'press', press ; Read pressure.
ncdf_getvar, ncdf, 'temp', temp   ; Read temperature.
```

[6] See http://www.ittvis.com/language/en-US/ProductsServices/IDL.aspx.
[7] See http://www.roguewave.com/products/pv-wave-family.aspx.

```
temp = temp - 273.15                ; Convert to degree C.

plot_io, temp, press, xtitle = "Temperature (K)", xrange = [-90., 30.], $
                      ytitle = "Pressure (hPa)", yrange = [1000., 0.1]
```

In order to run `ncdf_getvar` from the PV–Wave command line prompt, the file `ncdf_getvar.pro` as well as the other support files must have been installed somewhere in Wave's search path, or in the current directory; see the Wave documentation for details. Again, note that all routines in the directory `contrib` have only been provided as a matter of convenience, and are not fully supported by the ROM SAF.

2.5.3 R

R[8] is a freely available environment for statistical computing and graphics. There are various packages available from on the Comprehensive R Archive Network[9] which facilitate the reading and writing of netCDF data; a good choice is `RNetCDF`. A plot similar to the ones discussed for the previous graphics packages can be generated by (assuming that the `RNetCDF` library has been installed):

```
library(RNetCDF)                          # Load the RNetCDF library.
nc    <- open.nc(ncdf)                    # Read the data.
temp  <- var.get.nc(nc, "temp") - 273.15
press <- var.get.nc(nc, "press")
close.nc(nc)

plot(temp, press, log = "y", type = "l", ylim=c(1000,0.1))  # Make the plot.
```

2.5.4 ncl

The NCAR Graphics Command Language[10] (ncl) is a freely available graphics system which supports netCDF as one of its main data formats. Creating a similar plot as in the IDL and PV–Wave examples before would require (the variable `ncdf` contains the file name of an ROPP singlefile, and the file `gsn_code.ncl` must have been loaded):

```
ro_profile = addfile(ncdf, "r")       # Read the data.
press = ro_profile->press
temp  = ro_profile->temp

temp = temp - 273.15                  # Convert to degree C.

wks = gsn_open_wks("x11", "example")
```

[8] See http://www.r-project.org.
[9] See http://cran.r-project.org.
[10] See http://www.ncl.ucar.edu.

```
res            = True                # Set resources.
res@trYLog     = True
res@trYReverse = True
res@trXMinF    =  -90.
res@trXMaxF    =   30.
res@trXMinF    =    0.1
res@trXMaxF    = 1000.

plot = gsn_xy(wks, temp, press, res) # Make the plot.
```

2.6 Data handling and conversion tools

2.6.1 ropp2ropp

The program ropp2ropp provides a management tool for ROPP netCDF files. The program supports the merging of several ropp files into a single multifile, as well as the splitting of multifiles into several files, each holding a single individual profile. The program may therefore be used to handle simple data organisation tasks. Note that individual files which are to be merged must fulfil the requirements for data in multifiles, i.e. the various data levels must exhibit the same number of elements across all data files, or an error will occur. Summary help information can be displayed by:

```
> ropp2ropp -h
```

or for more detailed information consult the Unix manual page:

```
man ropp2ropp
```

or the ropp_io Reference Manual.

2.6.2 bufr2ropp and ropp2bufr

If a supported BUFR library is available on the user's computer system (see Appendix B), the programs ropp2bufr and bufr2ropp will be available as part of ropp_io. These programs convert between the ROPP netCDF and WMO BUFR formats and vice versa. For details on their usage, the -h switch works as for ropp2ropp or consult the respective Unix manual pages or the ropp_io Reference Manual.

For example, to encode and decode RO data:

```
> ropp2bufr <input_filename.nc> -o <output_filename.bufr>
> bufr2ropp <input_filename.bufr> -o <output_filename.nc>
```

Note that the BUFR format, while saving disk space and transmissison bandwidth because of its in-built compression, has certain limitations compared to the netCDF binary format. In particular, orbit data cannot be stored in BUFR with sufficient numerical precision or vertical resolution for most RO applications. Also, being a specialist format designed for operational use by NWP centres, it contains only a sub-set of all the possible parameters defined in the ROPP netCDF files; low–level 'raw' data like excess phases, signal amplitudes (or Signal–to–Noise ratios), for instance, are not present in BUFR.

Currently there are two supported BUFR libraries that these tools can interface with — the Met Office 'MetDB' and ECMWF packages. Only one need be installed; see the ROPP Release Notes for details. The -v switch can be used to confirm which library these tools were built with.

2.6.3 ucar2ropp

This tool converts the UCAR 'atmPhs', 'atmPrf', 'sonPrf', 'ecmPrf', 'ncpPrf' or 'gfsPrf' netCDF files to standard ROPP netCDF format. Summary help information can be displayed by:

```
> ucar2ropp -h
```

For example, to convert a UCAR format `atmPrf` file containing Level1b and Level2 data to ROPP netCDF (e.g. for use as input to a 1D-Var routine):

```
> ucar2ropp <atmPrf_filename.nc> -o <outputROPP_filename.nc>
```

Similarly, to convert a UCAR format `atmPhs` file containing Level1a data to ROPP netCDF (e.g. for use as input to the `ropp_pp` module routines):

```
> ucar2ropp <atmPhs_filename.nc> -o <outputROPP_filename.nc>
```

See the program's summary help, man pages and I/O Reference Manuals for more information on usage.

2.6.4 `gfz2ropp`

This tool converts GFZ native text files — the '`.dat`' and '`.dsc`' pair to standard ROPP netCDF format. Summary help information can be displayed by:

```
> gfz2ropp -h
```

For example, to convert a GFZ format pair of files containing Level1b and Level2 data to ROPP netCDF (e.g. for use as input to a 1D-Var routine):

```
> gfz2ropp <inputGFZ_filename.dat> -o <outputROPP_filename.nc>
```

It is assumed that a companion '`.dsc`' file with the same name exists in the same directory as the '`.dat`' file specified on the command line. See the program's summary help, man pages and I/O Reference Manuals for more information on usage.

2.6.5 `grib2bgrasc`

If a supported `GRIB_API` library is available on the user's computer system (see Appendix B), the program `grib2bgrasc` will be available as part of `ropp_io`. This program reads gridded data in ECMWF GRIB format (edition 1 or 2), and outputs a profile of background fields in ascii format — a Fortran namelist, in fact. This namelist can be converted to ROPP format by the `ropp_io` sister program `bgrasc2ropp`. For example, to extract a background profile of RO data:

```
> grib2bgrasc <ifile.grib> -lat <lat> -lon <lon>  -o <ofile.nml>
```

The GRIB files must be gridded at a uniform longitudinal and and latitudinal grid, and must contain temperature (K), specific humidity (kg/kg), ln(surface pressure (Pa)). The surface geopotential height (m^2/s^2) must also be provided. Regularly gridded GRIB files may be downloaded from the ECMWF data portal at http://www.ecmwf.int/products/data/. The surface geopotential is not routinely provided with analysis or forecast data, but it can be found in reanalysis products, for example. To handle this complication, `grib2bgrasc` allows the user to specify an auxiliary file which contains it, thus:

```
> grib2bgrasc <ifile.grib> -z <ifilez.grib> -lat <lat> -lon <lon> -o <ofile.nml>
```

Very coarse (4°) resolution GRIB datasets are provided with the ROPP distribution, for testing purposes. Their contents and structure may be investigated by tools such as `grib_dump` and `grib_ls`, which are built as part of the standard `GRIB_API`, and which may be found in `$ROPP_ROOT/<compiler>/bin`. The program has been succesfully tested on datasets as fine as $1/4^\circ$ resolution.

`grib2bgrasc` also attempts to calculate the undulation (see ROPP User Guide Part II (2013)) provided that the environment variables `GEOPOT_COEF` and `GEOPOT_CORR`, which specify files that define the shape of the EGM-96 geoid, are specified. If the ropp_pp module has been downloaded, suitable files are `ropp_pp/data/egm96.dat` and `ropp_pp/data/corrcoef.dat` for `GEOPOT_COEF` and `GEOPOT_CORR` respectively. Alternatively, the user may specify the undulation directly by using the -u option. The user may also choose to specify the radius of curvature of the tangent plane by means of the -r option.

Time interpolation between the fields in two GRIB files is also supported. In this case the desired date and time need to be specified, thus:

```
> grib2bgrasc <ifile1.grib> <ifile2.grib> -lat <lat> -lon <lon> \
>             -date <date> -time <time> -o <ofile.nml>
```

For details on the usage of `grib2bgrasc`, consult the respective Unix manual pages or the `ropp_io` Reference Manual.

2.6.6 `bgrasc2ropp`

The program `bgrasc2ropp` converts ascii data (in the form of a Fortran namelist) into an ROPP-formatted netCDF file. This data might, for instance, be generated by the `ropp_io` sister program `grib2bgrasc`. For example, to extract a background profile of RO data:

```
> bgrasc2ropp <ifile.nml>  -o <ofile.nc>
```

For details on the usage of `bgrasc2ropp`, consult the respective Unix manual pages or the `ropp_io` Reference Manual.

2.6.7 `eum2ropp` and `eum2bufr`

If the `netCDF4/HDF5` library is available on the user's computer system (see Appendix B), the program `eum2ropp` will be available as part of `ropp_io`. This program reads 'EUMETSAT-formatted' RO data, which exploits the netCDF4 features of data 'groups' and which is used to hold data from the ROSA instrument on OceanSat-2, and converts it into standard ROPP format. An example 'EUMETSAT-formatted' netCDF4 data file is provided with the distribution (it is used to test these tools). For example, to convert EUMETSAT-formatted RO data to standard format:

```
> eum2ropp <input_filename.n4>  -o <output_filename.nc>
```

For details on the usage of `eum2ropp`, consult the respective Unix manual pages or the `ropp_io` Reference Manual.

If, in addition, the ECMWF `BUFR` library is also available (see Appendix B again), the program `eum2bufr` will be available as part of `ropp_io`. This program reads EUMETSAT-formatted RO data and converts it to (ECMWF) BUFR format. For example, to convert EUMETSAT-formatted RO data to BUFR format:

```
> eum2bufr <input_filename.n4>  -o <output_filename.bufr>
```

For details on the usage of `eum2bufr`, consult the respective Unix manual pages or the `ropp_io` Reference Manual.

2.7 ROPP Thinner

The ropp_io module contains several routines which can be used to thin Level 1b, Level2a and Level2b data. Thinning aims to reduce the amount of data without reducing the information content. Details on the ROPP thinning algorithms are provided by (ROM SAF, 2009) and (ROM SAF, 2007).

ROPP supports several different thinning algorithms. For details see ROM SAF (2007).

NONE	: no thinning.
SAMPLE	: Uniform sampling (select or reject 1-in-N), up to a specified maximum number of thinned levels.
LIN	: Linear interpolation to fixed levels (no smoothing).
LOG	: Logarithmic interpolation to fixed levels (no smoothing).
SGLIN	: Savitzky-Golay smoothing filter with linear interpolation to fixed number of thinned levels.
SGLOG	: Savitzky-Golay smoothing filter with log interpolation to fixed number of thinned levels.
ASGLIN	: Adaptive S-G smoothing filter with linear interpolation to fixed number of thinned levels.
ASGLOG	: Adaptive S-G smoothing filter with log interpolation to fixed number of thinned levels.

The thinner may be called from user programs as

```
USE ropp_io
TYPE(ROprof) :: ro_data
  ...
CALL ropp_io_thin(ro_data, ThinFile)
```

where ThinFile is the name of a thinning control file. This is a plain-text file which should be formatted as follows.

```
Title=<title>      ! free-text description
Method=<method>    ! Thinning method selected from options above
Nlevels=<nlev>     ! Maximum number of levels for output
Hlevels=           ! Required set of fixed levels as impact heights
  <level1>
  <level2>
  ...
  <levelnlev>
```

The impact heights (Hlevels) on which data are to be thinned are not used by the SAMPLE method. Otherwise, the height values should increase monotonically.

The ROPP thinner may be optionally called by the ropp2ropp, ucar2ropp and ropp2bufr tools. This is achieved by specifying the ThinFile with the '-p' command line argument.

```
> ropp2ropp input_data.nc -o thinned_data.nc -p ThinFile.dat
```

Note that when using one of the log-based thinners, only those variable which vary roughly exponentially with height (bending angle, refractivity, humidity and pressure) are interpolated this way. All other variables are interpolated linearly.

References

ROM SAF, Mono-dimensional data thinning for GPS Radio Occultations, SAF/GRAS/METO/REP/GSR/001, 2007.

ROM SAF, ROPP Thinner Algorithm, SAF/GRAS/METO/REP/GSR/008, 2009.

ROM SAF, The Radio Occultation Processing Package (ROPP) User Guide. Part II: Forward model and 1dVar modules, SAF/ROM/METO/UG/ROPP/003, Version 7.0, 2013.

Unidata, netCDF website, http://www.unidata.ucar.edu/software/netcdf/index.html.

A ropp_utils library

The ropp_utils library contains a collection of utility routines which are used by other ROPP modules. They are not intended to be called directly by user applications, so they are not documented in any detail here. This chapter gives only an overview of the library which is divided into sub-libraries by related functionality. The reader is directed to the ROPP Utils Reference Manual (SAF/ROM/METO/RM/ROPP/001).

A.1 Missing data values

The ropp_utils module defines a set of parameters to indicate and test a 'missing' or 'invalid' data value used by any ROPP routine. These are set in the main module file ropp_utils.f90.

- ropp_MDFV (=-99999000.0) is used to set missing (invalid) data for real ROPP parameters
- ropp_MDTV (=-9999.0) is used for testing invalid real parameter values. Anything less than this value can be assumed to be 'missing'
- ropp_ZERO (=0.0) is used to set parameters to zero
- ropp_ZDTV (=1.0e-10) is used to test for (almost) zero values
- ropp_MIFV (=-999) is used to set missing (invalid) data for integer ROPP parameters
- ropp_MITV (=-99) is used for testing invalid integer parameter values. Anything less than this value can be assumed to be 'missing'

A.2 ropp_messages

These routines provide an interface to write information and error messages to stdout or stderr from ROPP routines. The utilities were originally written by Christian Marquardt as personal code, independent of the ROM SAF. The author grants a free-use licence for all of this code. The utilties have since been modifed and extended for ROPP.

A msg_MODE parameter is used to control the level of output diagnostic information output by ROPP routines. The available options are

- QuietMode - only output error messages to stdout, no info/warnings
- NormalMode - output all info and warnings and errors to stdout
- VerboseMode - as NormalMode, but also output diagnostic/debug messages to stdout

The required msg_MODE may be altered either within a program routine, e.g.

```
USE messages
msg_MODE = VerboseMode           ! Enable all messages
```

```
CALL message(msg_diag, ''The result is...'')
msg_MODE = NormalMode            ! Re-set to normal output level
```

or by implicitly setting the default value in the `ropp_messages/messages.f90` file before compiling, or by setting the environment variable `ROPP_MSG_MODE` on the command line. Note that VerboseMode can be selected when running any of the stand-alone tools provided with ROPP by running it with a '-d' command-line option.

A.3 Unitconvert

A collection of low-level F90 routines for converting data between standard units used within ROPP modules. The conversion scaling factors and offsets for a given unit conversion operation are defined in routine `ropp_unit_conversion.f90`. The set of available unit conversions provided may be extended by a user as required. These unit conversion routines are called automatically from within `ropp_io` module read and write routines, and from within `ropp_fm` and `ropp_1dvar` routines to ensure variables have required units.

A.4 Coordinates

A collection of low-level F90 coordinate manipulation routines. Functionality includes routines to convert cartesian position vectors between Earth Centred Fixed and Earth Centred Inertial reference frames, convert between cartesian and geodetic position description, compute impact parameter, occultation point, radius of curvature and undulation (ie, difference between the EGM96[1] geoid and the WGS84[2] ellipsoid). Vector manipulation routines (vector product, vector angle, rotation) are also included.

A.5 Datetime

A collection of low-level F90 date and time conversion routines. The utilities were developed within the Met Office outside the context of the ROM SAF and represents pre-existing software (PES). This code is Crown copyright.

A.6 Geodesy

A collection of low-level F90 geodetic conversion routines to convert to/from geometric/geopotential heights and compute Earth's effective radius and gravity. These routines are based on Somigliana's equation to compute height scales relative to the WGS-84 reference ellipsoid.

[1] See http://cddis.nasa.gov/926/egm96/egm96.html.
[2] See http://earth-info.nga.mil/GandG/wgs84/index.html.

A.7 Arrays

A collection of low-level F90 array manipulation routines supporting all data types. The utilities were written by Christian Marquardt as personal code, independent of the ROM SAF. The author grants a free-use licence for all of this code.

A.8 Misc

Miscellaneous utilities used by other ROPP modules. `FileDelete.f90` and `GetIOUnit.f90` are low-level F90 file handling routines. `ToUpper.f90` and `ToLower.f90` are low-level F90 string handling routines.

A.8.1 typeSizes

`typeSizes.f90` is a public-domain F90 module written by Robert Pincus (Cooperative Institute for Meteorological Satellite Studies, Madison) which provides named kind parameters for use in declarations of real and integer variables with specific byte sizes. It is a copy of the same file included in the 3rd party netCDF package, but is bundled with, and used by, ROPP as a stand-alone tool to provide a standardised type naming convention. This is 'freeware' provided complete and 'as-is' under the terms of usage. Users of `ropp_utils` must respect the same conditions in turn.

B Installing and using ROPP

B.1 Software requirements

ROPP is written in standard Fortran 95. Thus, compilation and use of the routines forming ROPP require the availability of standard ISO-conforming compilers. Fortran 95 was preferred over Fortran 90 because it has a number of convenient features. In particular, it allows elemental functions and pointers can be nullified when they are declared.

B.2 Software release notes

The latest ROPP distribution is available for download via the ROM SAF website http://www.romsaf.org. The ROPP Release Notes available from the ROPP download page and provided with the main ROPP download tarfile gives instructions for unpacking and installing the complete ROPP package, or individual modules. Users are strongly recommended to refer to the ROPP Release Notes and use the build and configure tools described therein. The information contained here are intended to complement the ROPP Release Notes. Where any contradiction between the User Guide and ROPP Release Notes exist, the ROPP Release Notes page is considered to be the most up-to-date latest information.

B.3 Third-party packages

To fully implement ROPP, the code uses some standard third-party packages. These are all non-commercial and cost-free. Note that third-party codes are only needed by the `ropp_io` and `ropp_pp` modules, so are optional if these modules are not required by the user.

All third-party code or packages used by ROPP are, by definition, classed as 'Pre-Existing Software' and all rights remain with the originators. Separate rights licences may be part of these distributions — some may have a licence which may impose re-distribution restrictions — and such licences must be adhered to by users.

If a third-party package is required, this must be built and installed before attempting to build the ROPP code. For convenience, these packages should be installed to the same root path as ROPP. It is highly recommended that the package is compiled using the same compiler and using the same compiler flags as will be used to build the ROPP code. Example configure scripts for supported compilers are provided in the `ropp_build` module available from the ROPP download website. See Section B.4 for further details.

B.3.1 NetCDF

The input/output library `ropp_io` uses Unidata's `netCDF` data format. Thus, the `netCDF` library and its associated utility programs (like `ncdump`, `ncgen`) are required and must be properly installed on the user's system before the compilation of the `ropp_io` package can be attempted. `netCDF` may also be used for reading MSIS climatology data as part of the `ropp_pp` module.

The SAF provides versions of the `netCDF` distribution, which have been successfully integrated with ROPP, alongside the ROPP distribution. This may not be the most recent distribution. Latest versions are freely available from

http://www.unidata.ucar.edu/software/netcdf/

Note that the `tests` subdirectory of the `ropp_io` distribution contains a simple test to check if the `netCDF` installation works; see Section B.7 for details.

A very useful complementary set of tools for handling and manipulating netCDF data files are the netCDF Operators `nco`.[1] While the latter are not required for using `ROPP` libraries and sample applications, we highly recommend them.

Some example and test programs provided with the `ropp_pp`, `ropp_fm` and `ropp_1dvar` packages read data via `ropp_io`. A complete installation of the `ropp_io` library is therefore required if the test programs or one of the sample applications are to be run. As a consequence, the complete installation of these packages also requires the availability of `netCDF`. Note, however, that the libraries `libropp_pp.a`, `libropp_fm.a` and `libropp_1dvar.a` can be compiled and installed without `ropp_io` and thus `netCDF`; the configuration script will recognise the absence of these libraries and only compile and install the core pre-processor, forward model or 1DVar routines (i.e. those with no dependencies on `netCDF` or `ropp_io`).

B.3.2 BUFR (optional)

The GNSS-RO BUFR encoder/decoder tools `ropp2bufr` and `bufr2ropp` in `ropp_io` require either the Met Office's 'MetDB' or the ECMWF BUFR library to be pre-installed. If neither BUFR library is detected by the installation configure script, then these tools will not be built.

The MetDB BUFR package is available without charge on request from the ROPP Development Team but with some licence restrictions. The ECMWF package is licenced under the GNU/GPL and can be downloaded from:

http://www.ecmwf.int/products/data/software/bufr.html

Both libraries generate essentially identical data when decoded (there may be non-significant round-off differences due to use of single– vs. double–precision interfaces). While the MetDB library is easier to install from a portability point of view, the ROPP `buildpack` script makes the ECMWF installation compatibly with ROPP more transparent. Therefore users can employ whichever BUFR package they prefer. Thus, the MetDB library could be built with

[1] See `http://nco.sourceforge.net/`.

```
> buildpack bufr <compiler>
```

or

```
> buildpack mobufr <compiler>
```

while the ECMWF library would be be built with

```
> buildpack ecbufr <compiler>
```

In order to install BUFR tables and related files, and for the applications to find them at run-time, an environment variable must be pre-defined to the path to these files. For instance, for the MetDB library:

```
> export BUFR_LIBRARY=<path>/data/bufr/
```

or for the ECMWF library:

```
> export BUFR_TABLES=<path>/data/bufr/
```

Note that in both cases, the path must currently be terminated with a '/' character, although this restriction will be relaxed for later (v20+) releases of the MetDB BUFR library. By default, the `buildpack` script will set `<path>` to be `ROPP_ROOT`.

B.3.3 GRIB_API (optional)

The GRIB background reading tool `grib2bgrasc` in `ropp_io` requires the ECMWF GRIB_API library to be pre-installed. If it fails to be detected by the installation configure script, then this tools will not be built.

The ECMWF GRIB_API package is licenced under Apache (2.0), and can be downloaded from:

https://software.ecmwf.int/wiki/display/GRIB/Releases

The ROPP `buildpack` script allows installation of the GRIB_API by means of:

```
> buildpack grib <compiler>
```

B.3.4 netCDF4/HDF5 (optional)

The 'EUMETSAT-formatted' RO reading tools `eum2ropp` and `eum2bufr` in `ropp_io` require the installation of a netCDF4 library, with its attendant HDF5 and zlib libraries.

These can be found from

http://www.unidata.ucar.edu/software/netcdf/,

http://www.hdfgroup.org/ftp/HDF5/releases

and

http://www.zlib.net/

respectively.

The ROPP `buildpack` script allows installation of these libraries as follows:

```
> buildpack zlib <compiler>
> buildpack hdf5 <compiler>
> buildpack netcdf4 <compiler>
```

(They would need to be installed in this order, since each depends on the one before.) Note that the `ropp_io` tool `eum2bufr` has only been interfaced to the ECMWF BUFR library. Note too that the `zlib`, and possibly also the `HDF5` libraries may already be installed as part of a standard Linux distribution, in which case the user need not of course build a local version.

B.3.5 RoboDoc (optional)

The ROPP Reference Manuals have been auto-generated using the RoboDoc documentation tool[2] All source code, scripts, etc. have standardised header comments which can be scanned by RoboDoc to produce various output formats, including LaTeX and HTML. If code (and in particular the header comments) is modified, RoboDoc can optionally be used to update the documentation. This tool is not required in order to build the ROPP software.

B.3.6 autoconf and automake (optional)

The `automake` and `autoconf` tools, common on most Linux and Unix systems, are not necessary to build the ROPP package as provided, but are useful if any modifications are made to the code or build systems to re-generate the package `configure` files. Versions at, or higher than, v1.9 are required to support some of the m4 macros defined in the ROPP build system.

B.4 BUILDPACK script

The ROPP package distribution includes a collection of configure and build scripts for a number of compilers and platforms suitable for ROPP and the dependency packages. A top-level BASH shell script `buildpack` is provided which may be used to automate the build of any ROPP module or dependency package in a consistent way, using the appropriate configure scripts. Use of `buildpack` is therefore highly recommended for first time build and less experienced users. Summary usage can be obtained using

```
> buildpack -h
```

In general, to build and install a package,

```
> buildpack <package> <comp> [[NO]CLEAN]
```

[2] See `http://rfsber.home.xs4all.nl/Robo/robodoc.html`.

where <package> is on of the supported package names (e.g. `ropp_fm`, `ropp_io`, `netcdf`, `mobufr`, `ecbufr`, etc.) and <comp> is the required compiler (e.g. `ifort`, `nag`, `xlf95`, etc.).

The `buildpack` script assumes that all tarball files and configure scripts provided with the ROPP distribution are placed in the same working directory. Packages will be decompressed here and installed to the `ROPP_ROOT/<comp>` target directory. The script automates the `configure` — `make` — `make install` build cycle described below. Further information on the `buildpack` script are provided in the ROPP Release Notes.

Other shell scripts `build*` are provided which can be used to further automate the build process by calling `buildpack` with a pre-determined sequence of packages or compilers. Users should review and edit these to suit their requirements.

B.5 Building and installing ROPP manually

The low-level build sequence performed by `buildpack` may be implemented manually by more experienced users. After unpacking, all packages are compiled and installed following the `configure` — `make` — `make install` cycle.

1. First run the command `configure` to check for the availability of all required libraries. `configure` allows the user to specify compiler options, paths to libraries and the location where the software shall eventually be installed, on the command line or as environment variables. Based on this information, `configure` generates user specific `Makefiles`, allowing a highly customised configuration and installation of the software.
2. Compilation is then initiated with the command `make`.
3. If building the software was successful, a `make install` will install libraries, header and module files as well as any executables in the directories specified by the user via the configure step.

Note that the ROPP modules partially depend on each other. In particular, all packages require that `ropp_utils` has been installed successfully. This package therefore needs to be compiled and installed first. Most packages make use of the `ropp_io` package for sample applications and testing, and should therefore be installed next if these are required. Note that users wishing to use ROPP source code directly in their own applications need not install the `ropp_io` module. If the `ropp_io` module is not available at build time, only the source code libraries will be compiled. We thus recommend the following build order:

i) Third-party packages
ii) `ropp_utils`
iii) `ropp_io` (if required)
iv) `ropp_pp` (if required)
v) `ropp_fm` (if required)
vi) `ropp_1dvar` (if required)

Note that *all* libraries need to be built with the same Fortran compiler, and preferably with the same version of the compiler as well.

Supported Fortran (and C) compilers are listed in the Release Notes distributed with the ROPP package.

B.5.1 Unpacking

Once the required third-party software packages have been installed successfully, the ROPP packages can be installed. The complete ROPP package and individual modules are distributed as gzipped tar (.tar.gz) files. The complete package file name consists of the version name (e.g. ropp-7.0.tar.gz). This file contains the complete ROPP distribution. The module file names consist of the package's name (e.g. ropp_utils) and version (e.g. 7.0), as in ropp_utils-7.0.tar.gz. If GNU tar is available (as on Linux systems), gzipped tar files can be unzipped with

```
> tar -xvzf ropp-7.0.tar.gz
```

Older, or non-GNU, versions of tar might need

```
> gunzip -c ropp-7.0.tar.gz | tar -xv
```

In all cases, a new subdirectory named (in the above example) ropp-7.0 will be created which contains the source code of the complete package.

B.5.2 Configuring

Details on the installation procedure for the individual packages can be found in the files README.unix and README.cygwin for the installation under Unix and Windows (with Cygwin), respectively. Here, we provide a brief example for a Unix or Linux system.

Unpacking the ropp_build package will create the configure/ sub-directory containing a number of mini-scripts for local build configuration. The files have names <package>_configure_<compiler>_<os> where <package> is the package name (ropp, netcdf), <compiler> is the compiler ID (ifort, nag, pgf, g95...) and <os> is the operating system ID, as output by the uname(1) command but entirely in lower case (linux, cygwin, hp-ux...). Note these configure mini-scripts are also used by the high-level buildpack script. The example configure scripts for specific platforms and compilers may need to be edited for optimal local use, or users may create their own following one of the examples.

The main configure scripts provided assume that the external libraries are all installed under /usr/local, i.e. the libraries can be found in the directory /usr/local/lib, and header and module files in /usr/local/include. It is further assumed that the individual ROPP packages are to be installed under $ROPP_ROOT, i.e. libraries are intended to reside in $ROPP_ROOT/lib, programs in $ROPP_ROOT/bin, and headers and module files in $ROPP_ROOT/include. The $ROPP_ROOT location should be specified as an environment variable, e.g,

```
> export ROPP_ROOT=$HOME
```

For most compilers, this means that the two paths to the header and module files need to be specified via the proper compiler options – usually via the -I option. The linker also needs to know where libraries are located; on most Unix systems, this can be achieved by specifying the -L option at link time. Users are referred to the examples provided in the configure package for further details.

Running the appropriate script from `configure/` will set the required compiler flags and specify the header, module and library paths before running the `configure` script. For example if the Fortran 95 compiler is named (say) `ifort`, the following command would be sufficient to configure a package for later compilation:

```
> cd ropp_<module>
> ../configure/ropp_configure_ifort_linux
```

The `configure` script will check for all required libraries and add the required options for the linker. If `configure` is not successful finding the required libraries, an error message will be produced, and further compilation will not be possible. Should the configuration step fail entirely, the file `config.log` created during the run of `configure` usually gives some clues on what went wrong; the most likely reason for failing is that compiler or linker options (and in particular paths to include files or libraries) are not set correctly.

Note that `ropp_io` may optionally use other external libraries in order to support additional features. For example, the `ropp_io` library will provide two conversion tools from `ROPP` to `BUFR` and back if a supported `BUFR` library is found. The existence of such additional libraries is also checked during `configure`. If these libraries are missing, however, the installation can nevertheless proceed – the parts related to the missing library are simply not built. Should the build process fail in correctly finding the, e.g., `BUFR` libraries and therefore not build the `BUFR` tools, `config.log` should again provide evidence on what went wrong.

B.5.3 Compiling

If configuration was successful, the software can be built with the command

```
> make
```

This will compile all relevant source code, but may take several minutes. The resulting object library archive will be located in the `build` subdirectory. It will be named similar to the package following usual Unix conventions; for example, the `ropp_utils` library is named `libropp_utils.a`. Sample applications and test programs or scripts will also have been built in the relevant subdirectories. Sample and test runs can be performed without installing the software; for details on available test programs, see B.7.

Currently supported Fortran compilers include (on Linux unless otherwise stated): Intel's `ifort` (v9.x, v10.x and v11.x, XE (v12.x)); NAG's `f95` (v5.1), `nagfor` (v5.2); Portland Group's `pgf95` (v7, v11); SUN's `sunf95` (v8 with SunStudio 12); GNU's `g95` and `gfortran` on Linux and under Cygwin on MS Windows; IBM's (xlf95) on AIX. This list may be contracted or extended in the future. For a full list please refer to the ROPP Release Notes and README files in each sub-package.

B.5.4 Installing

After building the software successfully, the command

```
> make install
```

will install libraries in {prefix}/lib, Fortran modules in {prefix}/include, and any application programs in {prefix}/bin. Here, {prefix} is the prefix directory given as argument to the --prefix option of the configure command. In the above example, this would be the home directory. If no --prefix is given, the installation root directory defaults to /usr/local which would normally require root (sudo) privileges.

B.5.5 Cleaning up

The temporary files created during the compilation of any ROPP package can be removed from the package directory tree with

```
> make clean
```

Note that this will keep the information gathered during configuration as well as the build libraries and executables intact. Thus, a new build can be attempted using make without the need for another configure. To remove all data related to the build and install process, run

```
> make distclean
```

which will restore the original state of the unpacked package, but with all potential user modifications to the source code still in place.

If the software has been installed previously, but shall be removed from the user's computer, this can be accomplished with the command

```
> make uninstall
```

performed in the source code distribution directory. Note that this requires a configuration which is identical to the one used for the original installation of the software. It is not necessary to rebuild the software again before uninstalling it.

B.6 Linking

If one (or more) ROPP packages have been installed successfully, linking your application's code against the ROPP libraries requires the specification of all ROPP and all external libraries. For example, to create an executable from your own application.f90 and the ropp_io libraries, something like

```
> ifort -o application application.f90 -L/usr/local/lib -L$ROPP_ROOT/lib \
                    -lropp_io -lropp_utils -lnetcdf (-lnetcdff)
```

will be required. (Since netCDF-4.1.1, the netCDF C and Fortran routines have been split, with the latter held in libnetcdff.a. Hence, if compiling Fortran routines against a recent version of netCDF, -lnetcdff must be included in the list of libraries to be linked.)

B.7 Testing

The ROPP software has undergone formal testing before distribution, as will all future modifications and improvements. A subset of the test procedures and some reference files are provided with the source code in order to facilitate quick tests whether the compilation was completed successfully. Users can run these tests to ensure that there are no major problems. It should be kept in mind, though, that not all of the functionality of the corresponding package is fully tested. Note also that several of the test scripts attempt to run IDL to verify the output and display test results using standard viewer utilities (e.g. xv). (Future releases may use Python instead.) Tests that generate graphical output automatically display a corresponding reference figure (part of the distribution), against which the test result can be compared. By setting the environment variable $ROPP_PAUSE to TRUE, the user can examine the two figures at leisure, before allowing the test script to move on to the next figure by hitting any key.

B.7.1 ropp_utils

Tested as part of the other modules, mainly with ropp_io.

B.7.2 ropp_io

The subdirectory tests of the ropp_io distribution contains several test programs and scripts to test various aspects of the software. A test is provided to check the user's installation of the netCDF library. They can be run after a successful compilation of the ropp_io package with

```
> make test_netcdf
```

from within the tests subdirectory. The program executed for this test does not use ropp_io, but is exclusively based on the native Fortran 90 interfaces for netCDF. Failure of this test strongly indicates that there is a problem with the installation or setup of the external library, which needs to be fixed before ropp_io can be used.

A second test can be run with

```
> make test_ropp
```

which runs a script performing several conversions between ROPP data files. Running this test through make has the advantage that the results of the conversions are interpreted properly and result in 'success' or 'failure' messages.

If a supported BUFR library is available, the tests subdirectory will also contain a test script for the two programs ropp2bufr and bufr2ropp which convert ROPP data files to and and from BUFR format data files. Issuing the command

```
> make test_bufr
```

will run a number of conversions and provide some verbose information on the content of the BUFR files and the encoding and decoding process. The script finally also compares the results. Its output should be fairly self-explanatory. Note that due to limitations of the BUFR format, non-significant loss of precision may be detected and flagged as differences from the reference file; this is normal.

The gfz2ropp and ucar2ropp tools to convert GFZ native text files or UCAR netCDF files to ropp-standard netcdf are tested with the commands

```
> make test_gfz
> make test_ucar
```

The grib2bgrasc and bgrasc2ropp tools which extract background profiles from GRIB-format gridded data and convert to ascii format, and then convert this to a ROPP-format netCDF file, are tested with the commands

```
> make test_grib
> make test_bgrasc
```

The eum2ropp and eum2bufr tools to convert 'EUMETSAT-format' RO data into standard ROPP netCDF or BUFR files, are tested with the commands

```
> make test_eum
> make test_eumbufr
```

Finally, the command

```
> make test
```

will run all of the above described tests.

The test of the ropp_io library and tools can also be tested manually by running, for example,

```
> t_ropp2ropp -t -n
```

which will create a series of different files. These should be intercompared (e.g., using diff) according to the advice given through the program's execution. Users can safely ignore numerical differences in the order of the cutoff in the text representation of the ROPP data files. Also note that different file names will show up in the first line of the text representation of netCDF data files (files created by the test script with the extension .cdl) and can be ignored. The test_ropp target actually does the same, but interprets the differences between the files with the above issues in mind. Note that the output of t_ropp2ropp can be found in the file t_ropp2ropp.log when run through make.

B.7.3 ropp_pp

The subdirectory tests of the ropp_pp distribution contains testing software, to compare the geometric optic and wave optic processing with known output, check the consistency of the Abel integral routines

and compare the ionospheric correction processing with known output. It also checks the tropopause height tool. Run

```
> make test
```

to check if solutions agree to within expected limits.

B.7.4 `ropp_fm`

The subdirectory `tests` of the `ropp_fm` distribution contains testing software. Run

```
> make test
```

to check if everything is working correctly. A series of tests are run to run the 1D and 2D operator applications to generate simulated refractivity and bending angle profiles, which are compared with pre-calculated data. Also included are tests of the consistency of the 1D and 2D tangent linear and adjoint routines. Warning messages are written to `stdout` if the operator, tangent linear and adjoint routines do not meet the expected consistency checks.

B.7.5 `ropp_1dvar`

A simple test is provided to check the correct running of the 1D–Var stand-alone application. This inputs a file of 'observations' (refractivity profiles) simulated from a set of ECMWF model background profiles. The same backgrounds are used in the 1D–Var retrieval. Hence the expected retrieved output profiles should be identical to the background (within rounding errors).

The subdirectory `tests` of the `ropp_1dvar` distribution contains the testing software. Run

```
> make test
```

to check if everything is working correctly. The test runs the 1D–Var application to generate the result file, then runs an IDL script to read the background and results files, perform mean difference calculations and generate a plot file. Finally, XV is run to display the plot. If all is well, IDL-reported mean percentage differences should be, for all practical purposes, zero and the plots should show a series of vertical lines (each line is one profile, and each is offset from zero by a different amount for clarity) showing the bias in retrieved temperature, specific humidity and pressure with height.

B.8 Troubleshooting

If something goes wrong during the configuration step, carefully check the full output of the last unsuccessful `configure` run to get an idea why the software could not be built; this can be found in the file `config.log`. This also applies if parts of ROPP are not built (e.g. the `BUFR` tools), although the required additional libraries are available.

During compilation, warnings that indicate unused variables (e.g. with the NAG compiler) or the potential trimming of character variables (with Intel compilers) can safely be ignored. If compiling is fine, but installation fails, make sure you have write permissions on the installation directories.

If linking against ROPP libraries fails because of unresolved externals, make sure that *all* relevant libraries – *including all external ones* – are specified in the correct order (some linkers are not able to recursively browse through several libraries in order to resolve externals) with lower-level libraries following higher-level (ROPP) ones.

If the BUFR encoding or decoding fail with messages about missing run-time BUFR tables, check that the appropriate environment variable `BUFR_LIBRARY` (for the MetDB library) or `BUFR_TABLES` (for the ECMWF library) have been correctly set to the path of the installed BUFR tables, and that the path ends with a '/' character.

If an ROPP module compiles and runs satisfactorily, but produces unexpected results, an easy first step in tracking down the problem is to print out extra diagnostic information. Most of the ROPP tools provide the facility to do this by means of the '`-d`' option. `ropp_pp`, `ropp_1dvar` and `ropp_fm` also allow the user to add sets of pre-defined variables to the `ROprof` structure, which are written out in `netCDF` format with the usual variables. The first two modules do this by means of a option in a configuration file; the third by means of a command line option in (one of) the tools. In fact, all ROPP modules allow the user to add specified variables to the `ROprof` structure in this way, by calling `ropp_io_addvar`, as described in the ROPP I/O user Guide. This obviously requires the code to be recompiled.

C `ropp_io` program files

The `ropp_io` module provides support for a generic data format for radio occultation data. Routines are provided for flexible netCDF I/O of RO data via simple interfaces with a file management conversion tool, plus data thinning. Application tools include a BUFR encoder and decoder and conversion from UCAR and GFZ format data files to ROPP netCDF and a test data generator. A tool is provided to extract background profiles from ECMWF GRIB format gridded datasets; another tool converts the ascii output of this into ROPP format netCDF files.

Files listed in bold correspond to executable stand-alone tools. These call lower-level routines. In order to build this module the required packages must be first installed. Routines having additional dependencies on other packages or ROPP modules are listed with the required modules given in brackets. If the additional (optional) packages are not recognised by the configure script, only the core functions will be compiled and installed.

- Required packages: `ropp_utils`, `netcdf`
- Optional packages: `BUFR` (MetDB or ECMWF); `GRIB_API` (ECMWF); `netCDF4/HDF5`;

- Stand-alone tools and test programs († — requires MetDB or ECMWF `BUFR`; ‡ — requires ECMWF `GRIB_API`; § — requires `netCDF4/HDF5`).

tools/

- **gfz2ropp.f90**
- **ropp2ropp.f90**
- **test2ropp.f90**
- **ucar2ropp.f90**
- **bufr2ropp.f90** †
- **ropp2bufr.f90** †
- **grib2bgrasc.f90** ‡
- **bgrasc2ropp.f90**
- **eum2ropp.f90** §
- **eum2bufr.f90** §†(ECMWF BUFR only)

tests/

- **t_gfz2ropp.sh**
- **t_netcdf.f90**
- **nc_diff.f90**
- **nml_diff.f90**
- **t_ropp2ropp.sh**
- **t_roppthin.sh**
- **t_ucar2ropp.sh**
- **t_roppbufr.sh** †
- **t_grib2bgrasc.sh** ‡
- **t_bgrasc2ropp.sh**
- **t_eum2ropp.sh** §
- **t_eum2bufr.sh** §†(ECMWF BUFR only)

- Integrated code

```
bufr/
      bufr2ropp_mod.f90
      bufrutils.f90
      convertcodes.f90
      gtshdrs.f90
      ropp2bufr_mod.f90

ropp/
      ropp_io.f90
      ropp_io_addvar.f90
      ropp_io_ascend.f90
      ropp_io_assign.f90
      ropp_io_free.f90
      ropp_io_init.f90
      ropp_io_isinrange.f90
      ropp_io_nrec.f90
      ropp_io_occid.f90
      ropp_io_rangecheck.f90
      ropp_io_read.f90
      ropp_io_read_ncdf_get.f90
      ropp_io_shrink.f90
      ropp_io_types.f90
      ropp_io_vlist_print.f90
      ropp_io_vlist_read.f90
      ropp_io_vlist_size.f90
      ropp_io_write.f90
      ropp_io_write_ncdf_def.f90
      ropp_io_write_ncdf_put.f90

ncdf/
      is_netcdf.f90
      ncdf.f90
      ncdf_close.f90
      ncdf_create.f90
      ncdf_date_and_time.f90
      ncdf_datmode.f90
      ncdf_defdim.f90
      ncdf_defmode.f90
      ncdf_defvar.f90
      ncdf_error_handler.f90
      ncdf_getatt.f90
      ncdf_getatt_alloc.f90
      ncdf_getgroupid_n3.f90
      ncdf_getgroupid_n4.f90
      ncdf_getnrec.f90
      ncdf_getshape.f90
      ncdf_getsize.f90
      ncdf_getvar.f90
      ncdf_getvar_alloc.f90
      ncdf_isatt.f90
      ncdf_isvar.f90
      ncdf_open.f90
      ncdf_putatt.f90
      ncdf_putvar.f90
      ncdf_renvar.f90
      ncdf_save.f90
      ncdf_sync.f90

thin/
      ropp_io_thin.f90
      ropp_io_thin_fixed.f90
      ropp_io_thin_select.f90
      ropp_io_thin_sg.f90
      ropp_io_thin_skip.f90
```

D ROPP extra data

For reference and for completeness, the listings of the all ROPP modules' extra variables are listed below.

D.1 ropp_io_addvar

The general form of the extra data, appended to the `RO_prof` structure by `ropp_io_addvar`, is described in Table D.1.

ROprof (Additional variables requested by call to `ropp_io_addvar`, throughout ROPP)	
Structure element	Description
`...%vlist%VlistD0d%name`	Name of 1st 0D extra variable
`...%vlist%VlistD0d%long_name`	Long name of 1st 0D extra variable
`...%vlist%VlistD0d%units`	Units of 1st 0D extra variable
`...%vlist%VlistD0d%range`	Range of 1st 0D extra variable
`...%vlist%VlistD0d%DATA`	Value of 1st 0D extra variable
`...%vlist%VlistD0d%next%name (etc)`	Name (etc) of 2nd 0D extra variable
`...%vlist%VlistD0d%next%next%name (etc)`	Name (etc) of 3rd 0D extra variable
`...%vlist%VlistD1d%name`	Name of 1st 1D extra variable
`...%vlist%VlistD1d%long_name`	Long name of 1st 1D extra variable
`...%vlist%VlistD1d%units`	Units of 1st 1D extra variable
`...%vlist%VlistD1d%range`	Range of 1st 1D extra variable
`...%vlist%VlistD1d%DATA`	Value of 1st 1D extra variable
`...%vlist%VlistD1d%next%name (etc)`	Name (etc) of 2nd 1D extra variable
`...%vlist%VlistD1d%next%next%name (etc)`	Name (etc) of 3rd 1D extra variable
`...%vlist%VlistD2d%name`	Name of 1st 2D extra variable
`...%vlist%VlistD2d%long_name`	Long name of 1st 2D extra variable
`...%vlist%VlistD2d%units`	Units of 1st 2D extra variable
`...%vlist%VlistD2d%range`	Range of 1st 2D extra variable
`...%vlist%VlistD2d%DATA`	Value of 1st 2D extra variable
`...%vlist%VlistD2d%next%name (etc)`	Name (etc) of 2nd 2D extra variable
`...%vlist%VlistD2d%next%next%name (etc)`	Name (etc) of 3rd 2D extra variable

Table D.1: Additional elements of `ROprof` structure, available throughout ROPP

D.2 PPDiag

The extra data which are output to the netCDF file if `config%output_diag` is set to `.TRUE.` in `ropp_pp`, are described in Table D.2.

PPDiag (`config%output_diag` = `TRUE` in `ropp_pp`)	
Structure element	Description
`...%CTimpact`	CT processing impact parameter (m)
`...%CTamplitude`	CT processing amplitude
`...%CTamplitude_smt`	CT processing smoothed amplitude
`...%CTimpactL2`	CT processing L2 impact parameter (m)
`...%CTamplitudeL2`	CT processing L2 amplitude
`...%CTamplitudeL2_smt`	CT processing smoothed L2 amplitude
`...%ba_ion`	Ionospheric bending angle in L1 (rad)
`...%err_neut`	Error covariance of neutral bending angle (rad^2)
`...%err_ion`	Error covariance of ionospheric bending angle (rad^2)
`...%wt_data`	Weight of data (data:data+clim) in profile
`...%sq`	SO badness score: `MAX`[`err_neut`$^{1/2}/\alpha_N$].100%
`...%L2_badness`	L2 phase correction badness score

Table D.2: Elements of `PPDiag` structure, available from `ropp_pp`

D.3 ropp_fm_bg2ro

The extra data which are appended to the ROprof structure if the `ropp_fm` tool `ropp_fm_bg2ro_1d` is called without the '`-f`' option, are described in Table D.3.

ROprof (Absence of '`-f`' option in call to `ropp_fm_bg2ro_1d`, in `ropp_fm`)	
Structure element	Description
`...%gradient_refrac`	$\partial N_i/\partial x_j$ matrix
`...%gradient_bangle`	$\partial \alpha_i/\partial x_j$ matrix

Table D.3: Additional elements of `ROprof` structure, available from `ropp_fm`. See Table D.1 for the detailed structure.

D.4 VarDiag

The extra data which are output to the netCDF file if `config%extended_1dvar_diag` is set to `.TRUE.` in `ropp_1dvar`, are described in Table D.4.

VarDiag (`config%extended_1dvar_diag` = `TRUE` **in** `ropp_1dvar`)	
Structure element	Description
`...%n_data`	Number of observation data
`...%n_bgqc_reject`	Number of data rejected by background QC
`...%n_pge_reject`	Number of data rejected by PGE QC
`...%bg_bangle`	Background bending angle
`...%bg_refrac`	Background refractivity
`...%OmB`	Observation minus background
`...%OmB_sigma`	OmB standard deviation
`...%pge_gamma`	PGE check gamma value
`...%pge`	Probability of Gross Error along profile
`...%pge_weights`	PGE weighting values
`...%ok`	Overall quality flag
`...%J`	Cost function value at convergence
`...%J_scaled`	Scaled cost function value ($2J/m$)
`...%J_init`	Initial cost function value
`...%J_bgr`	Background cost function profile
`...%J_obs`	Observation cost function profile
`...%B_sigma`	Forward modelled bg standard deviation
`...%n_iter`	Number of iterations to reach convergence
`...%n_simul`	Number of simulations
`...%min_mode`	Minimiser exit mode
`...%res_bangle`	Analysis bending angle
`...%res_refrac`	Analysis refractivity
`...%OmA`	Observation minus analysis
`...%OmA_sigma`	OmA standard deviation

Table D.4: Elements of `VarDiag` structure, available from `ropp_1dvar`

E ROPP user documentation

Title	Reference	Description
ROPP Overview	SAF/ROM/METO/UG/ROPP/001	Overview of ROPP and package content and functionality
ROPP User Guide. Part I: Input/output module	SAF/ROM/METO/UG/ROPP/002	Description of `ropp_io` module content and functionality
ROPP User Guide. Part II: Forward model and 1D–Var modules	SAF/ROM/METO/1DVAR/ROPP/003	Description of `ropp_fm` and `ropp_1dvar` module content and functionality
ROPP User Guide. Part III: Pre-processor module	SAF/ROM/METO/PP/ROPP/004	Description of `ropp_pp` module content and functionality
ROPP UTILS Reference Manual	SAF/ROM/METO/RM/ROPP/001	Reference manual for the `ropp_utils` module
ROPP IO Reference Manual	SAF/ROM/METO/RM/ROPP/002	Reference manual for the `ropp_io` module
ROPP FM Reference Manual	SAF/ROM/METO/RM/ROPP/003	Reference manual for the `ropp_fm` module
ROPP 1D–Var Reference Manual	SAF/ROM/METO/RM/ROPP/004	Reference manual for the `ropp_1dvar` module
ROPP PP Reference Manual	SAF/ROM/METO/RM/ROPP/005	Reference manual for the `ropp_pp` module
WMO FM94 (BUFR) Specification for Radio Occultation Data	SAF/ROM/METO/FMT/BUFR/001	Description of BUFR template for RO data

Table E.1: ROPP user documentation

Title	Reference	Description
Mono-dimensional thinning for GPS Radio Occultations	SAF/GRAS/METO/REP/GSR/001	Technical report on profile thinning algorithm implemented in ROPP
Geodesy calculations in ROPP	SAF/GRAS/METO/REP/GSR/002	Summary of geodetic calculations to relate geometric and geopotential height scales
ROPP minimiser - minROPP	SAF/GRAS/METO/REP/GSR/003	Description of ROPP-specific minimiser, minROPP
Error function calculation in ROPP	SAF/GRAS/METO/REP/GSR/004	Discussion of impact of approximating erf in ROPP
Refractivity calculations in ROPP	SAF/GRAS/METO/REP/GSR/005	Summary of expressions for calculating refractivity profiles
Levenberg-Marquardt minimisation in ROPP	SAF/GRAS/METO/REP/GSR/006	Comparison of Levenberg-Marquardt and minROPP minimisers
Abel integral calculations in ROPP	SAF/GRAS/METO/REP/GSR/007	Comparison of 'Gorbunov' and 'ROM SAF' Abel transform algorithms
ROPP thinner algorithm	SAF/GRAS/METO/REP/GSR/008	Detailed review of the ROPP thinner algorithm
Refractivity coefficients used in the assimilation of GPS radio occultation measurements	SAF/GRAS/METO/REP/GSR/009	Investigation of sensitivity of ECMWF analyses to empirical refractivity coefficients and non-ideal gas effects
Latitudinal Binning and Area-Weighted Averaging of Irregularly Distributed Radio Occultation Data	SAF/GRAS/METO/REP/GSR/010	Discussion of alternative spatial averaging method for RO climate data
ROPP 1D–Var validation	SAF/GRAS/METO/REP/GSR/011	Illustration of ROPP 1D–Var functionality and output diagnostics
Assimilation of GPSRO Data in the ECMWF ERA-Interim Re-analysis	SAF/GRAS/METO/REP/GSR/012	Assimilation of GPSRO Data in the ECMWF ERA-Interim Re-analysis
ROPP_PP validation	SAF/GRAS/METO/REP/GSR/013	Illustration of ROPP_PP functionality and output diagnostics
A review of the geodesy calculations in ROPP	SAF/ROM/METO/REP/RSR/014	Comparison of various potential geodesy calculations
Improvements to the ROPP refractivity and bending angle operators	SAF/ROM/METO/REP/RSR/015	Improved interpolation in ROPP forward models

Table E.2: ROM SAF Reports

Title	Reference	Description
Product Requirements Document (PRD)	SAF/GRAS/METO/MGT/PRD/001	

Table E.3: Applicable documents

F Acronyms and abbreviations

AC	Analysis Correction (NWP assimilation technique)
API	Application Programming Interface
BG	Background
CASE	Computer Aided Software Engineering
CF	Climate and Forecasts (CF) Metadata Convention
CGS	Core Ground Segment
CHAMP	Challenging Mini–Satellite Payload
CLIMAP	Climate and Environment Monitoring with GPS–based Atmospheric Profiling (EU)
CODE	Centre for Orbit Determination in Europe
COSMIC	Constellation Observing System for Meteorology, Ionosphere & Climate
DMI	Danish Meteorological Institute
DoD	US Department of Defense
EC	European Community
ECF	Earth–centred, Fixed coordinate system
ECI	Earth–centred, Inertial coordinate system
ECMWF	The European Centre for Medium-Range Weather Forecasts
EGM–96	Earth Gravity Model, 1996. (US DoD)
EOP	Earth Orientation Parameters
EPS	EUMETSAT Polar System
ESA	European Space Agency
ESTEC	European Space Research and Technology Centre (ESA)
EU	European Union
EUMETSAT	European Organisation for the Exploitation of Meteorological Satellites
GALILEO	European GNSS constellation project (EU)
GCM	General Circulation Model
GFZ	GFZ Helmholtz Centre (Germany)
GLONASS	Global Navigation Satellite System (Russia)
GNSS	Global Navigation Satellite Systems (generic name for GPS, GLONASS and the future GALILEO)
GPL	General Public Licence (GNU)
GPS	Global Positioning System (US)
GPS/MET	GPS Meteorology experiment, onboard Microlab-1 (US)
GPSOS	Global Positioning System Occultation Sensor (NPOESS)
GRAS	GNSS Receiver for Atmospheric Sounding (onboard Metop)
GUI	Graphical User Interface
GTS	Global Telecommunications System

HIRLAM	High Resolution Limited Area Model
IERS	International Earth Rotation Service
ITRF	International Terrestrial Reference Frame
ITRS	International Terrestrial Reference System
IGS	International GPS Service
JPL	Jet Propulsion Laboratory (NASA)
LAM	Local Area Model (NWP concept)
LEO	Low Earth Orbited
LGPL	Lesser GPL (*q.v.*)
LOS	Line Of Sight
METOP	Meteorological Operational polar satellites (EUMETSAT)
MKS	Meter, Kilogram, Second
MPEF	Meteorological Products Extraction Facility (EUMETSAT)
MSL	Mean Sea Level
N/A	Not Applicable or Not Available
NASA	National Aeronautics and Space Administration (US)
NMS	National Meteorological Service
NOAA	National Oceanic and Atmospheric Administration (US)
NPOESS	National Polar-orbiting Operational Environmental Satellite System (US)
NRT	Near Real Time
NWP	Numerical Weather Prediction
OI	Optimal Interpolation (NWP assimilation technique)
Operational ROM SAF	Team responsible for the handling of GRAS data and the delivery of meteorological products during the operational life of the instrument
PAZ	Spanish Earth Observation Satellite, carrying a Radio Occultation Sounder
PFS	Product Format Specifications
PMSL	Pressure at Mean Sea Level
POD	Precise Orbit Determination
Q/C	Quality Control
RO	Radio Occultation
ROC	Radius Of Curvature
ROM SAF	The EUMETSAT Satellite Application Facility responsible for operational processing of radio occultation data from the Metop satellites. Members are DMI (leader), UKMO, ECMWF and IEEC.
ROPP	Radio Occultation Processing Package
ROSA	Radio Occultation Sounder for Atmosphere (on OceanSat-2 and Megha-Tropiques)
RMDCN	Regional Meteorological Data Communication Network
SAC–C	Satelite de Applicaciones Cientificas – C
SAF	Satellite Application Facility (EUMETSAT)
SAG	Scientific Advisory Group
SI	Système International (The MKS units system)
TAI	Temps Atomique International (International Atomic Time)

TanDEM–X	German Earth Observation Satellite, carrying a Radio Occultation Sounder
TBC	To Be Confirmed
TBD	To Be Determined
TDB	Temps Dynamique Baricéntrique (Barycentric Dynamical Time)
TDT	Temps Dynamique Terrestre (Terrestrial Dynamical Time)
TDS	True–of–date coordinate system
TerraSAR–X	German Earth Observation Satellite, carrying a Radio Occultation Sounder
TP	Tangent Point
UKMO	United Kingdom Meteorological Office
UML	Unified Modelling Language
UT1	Universal Time-1 (proportional to the rotation angle of the Earth)
UTC	Universal Time Coordinated
VAR	Variational analysis; 1D, 2D, 3D or 4D versions (NWP data assimilation technique)
VT	Valid or Verification Time
WGS–84	World Geodetic System, 1984. (US DoD)
WMO	World Meteorological Organization
WWW	World Weather Watch (WMO)

G Definitions

RO data products from the GRAS instrument onboard Metop and RO data from other data providers are grouped in levels and are either NRT or offline products.

Data levels:

- Level 0: Raw sounding, tracking and ancillary data, and other GNSS data before clock correction and reconstruction;
- Level 1a: Reconstructed full resolution excess phases, SNRs, amplitudes, orbit information, I, Q, and NCO values, and navigation bits;
- Level 1b: Bending angles and impact parameters, Earth location, metadata and quality information;
- Level 2: Refractivity profiles (level 2a), and pressure, temperature, and specific humidity profiles (level 2b and 2c), Earth location, metadata, and quality information;
- Level 3: Gridded level 1 and 2 offline profile products in the form of, e.g., monthly and seasonal zonal means, metadata, and quality information.

Product types:

- NRT product: data product delivered less than 3 hours after measurement;
- Offline product: data product delivered less than 30 days after measurement (the timeliness for some offline level 3 products may be up to 6 months).

H Copyrights

The majority of ROPP code is

Some parts of the source code within this distribution were developed within the Met Office outside the context of the ROM SAF and represents pre-existing software (PES); this portion is

This ROPP package also contains open source code libraries available through its author, Christian Marquardt. This is also PES, and is

IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

This ROPP package may also contain open source code libraries available through its author, Michael Gorbunov. This is also PES, and is

© Copyright 1998-2010 Michael Gorbunov <michael.gorbunov@zmaw.de>

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software with the rights to use, copy, modify, merge copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software as well as in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. HOWEVER, ALL EFFORTS ARE BEING MADE BY THE AUTHOR IN ORDER TO FIND AND ELIMINATE ALL POSSIBLE ERRORS AND PROBLEMS. IN THIS CASE THE AUTHOR MAY PROVIDE AN UPDATE.

This ROPP package may also contain open source code libraries available through its author, Stig Syndergaard. This is also PES, and is

© Copyright 1998 Stig Syndergaard <ssy@dmi.dk>

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sublicense the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software as well as in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.

IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.