RTTOV-9 Users Guide

Roger Saunders
Met Office, Exeter, UK
&
Marco Matricardi and Alan Geer
ECMWF

This documentation was developed within the context of the EUMETSAT Satellite Application Facility on Numerical Weather Prediction (NWP SAF), under the Cooperation Agreement dated 1 December, 2006, between EUMETSAT and the Met Office, UK, by one or more partners within the NWP SAF. The partners in the NWP SAF are the Met Office, ECMWF, KNMI and Météo France.

Change record			
Version	Date	Author / changed by	Remarks
1.0	14/08/07	R. Saunders	Initial draft
1.1	16/01/08	A. Lipton	Beta tester comments incorporated
1.2	19/02/08	R. Saunders	Other beta tester comments incorporated
1.3	22/02/08	R. Saunders	Added comments from Niels Bormann, Marco, Alan and Pascal
1.4	04/03/08	R. Saunders	Added more clarity on missing profiles
1.5	14/04/08	R. Saunders	Clarified 10m wind not 2m wind+ corrected Table 16 + eq 8
1.6	24/02/09	R. Saunders	More clarifications added to text after comments from Brett etc
1.7	04/02/10	R. Saunders	Changed header + removed ref to v9 + updated Intro.

Table of Contents

1. Introduction and Scope

This document gives an overview of the RTTOV9 fast radiative transfer model (in section 2), limitations in section 3, the differences from RTTOV-8 (in section 4), how to install the RTTOV9 fast radiative transfer model code on a UNIX/LINUX platform and run it (section 5) and how to apply it to the users particular application (section 6). The procedure for reporting bugs or learning about know bugs is given in section 7. Finally a frequently asked questions (FAQ) section is provided at section 8. This document relates to version 9 of the RTTOV code and all its sub versions (9.x). The document will not be systematically updated due to a change or new coefficient tables but users will be notified by email of these changes or can check on the RTTOV-9 web site (URL given below). Changes to this document are occasionally made to improve it and the document version is given in the header. If you want to request a copy of the RTTOV9 code go to http://www.metoffice.gov.uk/research/interproj/nwpsaf/request_forms/index.html and complete a licence agreement form on-line. You will then be given access to the code via FTP or sent a CD containing the code.

The old RTTOV-7 and 8 codes are still available in FORTRAN-90 but cannot be guaranteed to be upgraded for new instruments. Coefficient files for RTTOV-7/8 will continue to be made available from the NWP-SAF web site. RTTOV9 is a major rewrite of RTTOV-8 adding many more features.

The RTTOV9 scientific and validation report describes or gives links to the scientific basis of the model and also describes in more details any new scientific changes made. It also documents the test results carried out on the new code before delivery. The most up to date versions of these reports, including this users guide, can be viewed at the NWP-SAF web site: http://www.metoffice.gov.uk/research/interproj/nwpsaf/rtm/ in pdf format on the RTTOV-9 page. There is also a RTTOV-9 performance report which documents the run times of RTTOV-9 on a few platforms and compares these to the equivalent RTTOV-8 run times.

2. Overview of RTTOV9 and limitations

This section gives a brief overview of the RTTOV9 model and its limitations. More details can be found in the references given in this section. RTTOV9 is a development of the fast radiative transfer model for TOVS, RTTOV, originally developed at ECMWF in the early 90's (Eyre, 1991) for TOVS. Subsequently the original code has gone through several developments (e.g. Saunders et. al., 1999; Matricardi et. al., 2001), more recently within the EUMETSAT NWP Satellite Application Facility (SAF), of which RTTOV-8 and RTTOV-9 are the latest versions. RTTOV-9 also incorporates all the scientific features of RTIASI, the ECMWF fast radiative transfer model for the Infraed Atmospheric Sounding Interferometer (IASI) (Matricardi, 2003; Matricardi, 2005). The model allows rapid simulations (~1 ms for 40 channel ATOVS on a desktop PC) of radiances for satellite infrared or microwave nadir scanning radiometers given an atmospheric profile of temperature, variable gas concentrations, cloud and surface properties, referred to as the state vector. The only mandatory variable gas for RTTOV9 is water vapour. Optionally ozone, carbon dioxide, nitrous oxide, methane and carbon monoxide can be variable with all other constituents assumed to be constant. The state vector for RTTOV9 is given in Annex-J. Not all parameters have to be supplied as RTTOV can assume default profiles. A major change over previous versions of RTTOV is that RTTOV9 can accept input profiles on any set of pressure levels. The range of temperatures and water vapour concentrations over which the optical depth computations are valid depends on the training datasets which were used. This is defined in the coefficient file and for RTTOV9. 1 can be based on the 43L TIGR profile dataset or the 101L 52 profile ERA-40 dataset. The limits for the former are given in Table 1 and the limits for the latter can be found in the coefficient files supplied. For other gases and other profile datasets the limits are documented in the header section of the relevant coefficient file.

The spectral range of the RTTOV9 model is 3-20 microns (500 – 3000 cm^{-1}) in the infrared, governed by the range of the GENLN2 or kCarta or LBLRTM line-by-line datasets on which it is based. In the microwave the frequency range from 10 – 200 GHz is covered using the Liebe-89 MPM line-by-line model. The full list of currently supported platforms and sensors is given in Tables 2 and 3, although this list will be updated as new sensors are launched. New or updated coefficient files will be made available from the RTTOV pages on the NWP SAF web site.

An important feature of the RTTOV model is that it not only computes the forward (or direct) radiative transfer calculation but also the gradient of the radiances with respect to the state vector variables at the location in state space specified by the input state vector values. Given a state vector, **x,** a radiance vector, **y**, is computed:

$$\mathbf{y} = H(\mathbf{x}) \tag{1}$$

Level Number	Pressure (hPa)	Tmax deg K	Tmin degK	Qmax ppmv	Qmin ppmv	O_3max ppmv	O_3min ppmv	O_3Ref ppmv
1	0.1	304.9	174.6	12.8	1.4	1.732	0.359	0.640
2	0.3	314.8	196.2	11.5	2.6	2.755	0.794	1.398
3	0.7	330.9	194.2	12.2	3.3	5.612	1.103	2.504
4	1.4	339.5	179.9	9.6	2.9	8.601	1.927	4.346
5	2.6	338.0	174.6	9.2	3.0	9.892	3.179	6.496
6	4.4	319.6	166.2	8.5	2.5	11.330	3.953	7.652
7	7.0	306.5	159.9	8.1	1.9	12.440	3.801	7.807
8	10.4	299.3	160.9	8.0	1.5	12.740	2.073	7.349
9	14.8	293.9	163.2	7.8	1.5	12.540	1.643	6.605
10	20.4	289.7	162.1	7.6	1.4	11.680	1.585	5.657
11	27.3	282.7	158.4	7.3	0.8	9.272	1.084	4.577
12	35.5	276.0	158.9	7.2	0.7	7.008	0.224	3.443
13	45.3	271.8	162.2	7.0	1.1	6.497	0.094	2.428
14	56.7	269	165.3	7.2	1.3	5.804	0.012	1.645
15	70.0	269.5	165.7	7.8	1.2	4.720	0.010	1.105
16	85.2	267.2	164.5	10.6	1.0	3.756	0.008	0.733
17	102.1	264.7	162.8	20.8	0.6	2.955	0.009	0.500
18	122.0	264.7	162.0	46.5	0.4	2.510	0.008	0.346
19	143.8	263.5	162.2	101.5	0.3	2.151	0.006	0.245
20	168.0	262.5	163.9	210.7	0.4	1.711	0.006	0.174
21	194.4	263.7	167.1	408.7	0.7	1.274	0.005	0.126
22	222.9	265.7	170.9	831.6	1.2	0.893	0.005	0.097
23	253.7	268.5	176.1	1393.0	1.3	0.635	0.006	0.082
24	286.6	273.8	181.5	2351.0	1.3	0.390	0.006	0.075
25	321.5	279.8	184.9	3906.0	1.3	0.272	0.003	0.073
26	358.3	285.9	187.4	6192.0	1.3	0.178	0.001	0.073
27	396.8	291.1	190.3	8852.0	1.6	0.177	0.001	0.073
28	437.0	296.0	193.3	11670.0	1.5	0.186	0.001	0.073
29	478.5	301.4	195.7	15050.0	1.8	0.193	0.001	0.074
30	521.5	308.1	196.6	18170.0	2.3	0.198	0.001	0.074
31	565.5	311.4	192.2	20730.0	2.0	0.205	0.001	0.074
32	610.6	316.4	193.4	22530.0	3.4	0.213	0.001	0.074
33	656.4	318.9	189.5	25670.0	2.1	0.218	0.001	0.074
34	702.7	324.7	193.8	28850.0	8.1	0.220	0.001	0.073
35	749	330.7	196.4	34260.0	11.6	0.224	0.001	0.073
36	795.1	333.3	196.4	39300.0	15.4	0.226	0.001	0.072
37	840.0	337.8	196.4	40550.0	19	0.231	0.001	0.071
38	882.8	342.0	196.4	41730.0	20.3	0.238	0.001	0.070
39	922.5	343.6	196.4	45310.0	19.4	0.245	0.001	0.067
40	957.4	347.4	196.4	47690.0	18.7	0.249	0.001	0.062
41	985.9	349.0	196.4	52970.0	18.2	0.255	0.001	0.057
42	1005.4	346.8	196.4	46810.0	17.8	0.266	0.001	0.056
43	1013.3	346.8	194.8	44310.0	13.0	0.270	0.001	0.055

Table 1. Pressure levels adopted for RTTOV TIGR 43 level coefficient profile limits within which the transmittance calculations are valid. The default ozone profile is also given in the right hand column.

Platform	RTTOV id	Sat id range
NOAA[¶]	1	1 to 18
DMSP	2	8 to 17
Meteosat	3	5 to 7
GOES	4	8 to 13
GMS	5	5
FY-2	6	2 to 4
TRMM	7	1
ERS	8	1 to 2
EOS	9	1 to 2
METOP	10	2
ENVISAT	11	1
MSG	12	1 to 2
FY-1	13	3
ADEOS	14	1 to 2
MTSAT	15	1-2
CORIOLIS	16	1

¶ Includes TIROS-N

Table 2. Platforms supported by RTTOV_9 as at Feb 2008.

where *H* is the radiative transfer model (also referred to as the observation operator). The Jacobian matrix **H** gives the change in radiance **δy** for a change in any element of the state vector **δx** assuming a linear relationship about a given atmospheric state $\mathbf{x_0}$:

$$\mathbf{\delta y} = \mathbf{H}(\mathbf{x}_0)\mathbf{\delta x} \qquad (2)$$

The elements of **H** contain the partial derivatives $\partial \mathbf{y}_i/\partial \mathbf{x}_j$ where the subscript *i* refers to channel number and *j* to position in state vector. The Jacobian gives the top-of-atmosphere radiance change for each channel given unit perturbations at each respective level of the profile vectors and in each of the surface/cloud parameters. It shows clearly, for a given profile, which layers in the atmosphere are most sensitive to changes in temperature and variable gas concentrations for each channel. *RTTOV_K* (and its associated subroutines ending in *K*) compute the $\mathbf{H}(\mathbf{x}_0)$ matrix for each input profile.

It is not always necessary to store and access the full Jacobian matrix **H** and so the *RTTOV* package has routines to only output the *tangent linear* values ***δy***, the change in top of atmosphere radiances y_n for each channel *n*, for a given change in atmospheric profile, ***δx***, about an initial atmospheric state $\boldsymbol{x_0}$.

$$\delta y(x_0) = \left[\delta x \frac{\partial y_1}{\partial x}, \delta x \frac{\partial y_2}{\partial x}, \delta x \frac{\partial y_3}{\partial x}, ...\delta x \frac{\partial y_{nchan}}{\partial x} \right] \qquad (3)$$

Where the tangent linear routines all have *TL* as an ending. Conversely the adjoint routines (ending in *AD*) compute the change in any scalar quantity up to *nel* elements of the state vector (e.g. T, q, ozone, surface variables etc) ***δx*** for an assumed atmospheric state, $\mathbf{x_0}$, given a change in the radiances, *δy*.

$$\delta x(x_0) = \left[\delta y \frac{\partial x_1}{\partial y}, \delta y \frac{\partial x_2}{\partial y}, \delta y \frac{\partial x_3}{\partial y}, ...\delta y \frac{\partial x_{nel}}{\partial y} \right] \qquad (4)$$

These routines are normally used as part of the variational assimilation of radiances. Some more information on TL/AD and K codes is available at: http://cimss.ssec.wisc.edu/itwg/groups/rtwg/fastrt.html . For users who only want to compute radiances with the forward model the *TL/AD/K* routines are not required.

The model can simulate both clear sky radiances and cloudy radiances. It uses an approximate form of the atmospheric radiative transfer (RT) equation. If a black opaque cloud is assumed at a single level, the top of the atmosphere upwelling radiance, $L(v,\theta)$, at a frequency v and viewing angle θ from zenith at the surface, neglecting scattering effects, is written as:

$$L(v,\theta) = (1-N)L^{Clr}(v,\theta) + NL^{Cld}(v,\theta) \quad (5)$$

where $L^{Clr}(\text{v},\theta)$ and $L^{Cld}(\text{v},\theta)$ are the clear sky and fully cloudy top of atmosphere upwelling radiances and N is the fractional cloud cover.

Sensor	RTTOV id	Sensor Channel #	RTTOV-9 Channel #
HIRS	0	1 to 19	1 to 19
MSU	1	1 to 4	1 to 4
SSU	2	1 to 3	1 to 3
AMSU-A	3	1 to 15	1 to 15
AMSU-B	4	1 to 5	1 to 5
AVHRR	5	3b to 5	1 to 3
SSMI	6	1 to 7	1 to 7
VTPR1	7	1 to 8	1 to 8
VTPR2	8	1 to 8	1 to 8
TMI	9	1 to 9	1 to 9
SSMIS	10	1 to 24*	1 to 24*
AIRS	11	1 to 2378	1 to 2378
HSB	12	1 to 4	1 to 4
MODIS	13	1 to 16	1 to 16
ATSR	14	1 to 3	1 to 3
MHS	15	1 to 5	1 to 5
IASI	16	1 to 8461	1 to 8461
AMSR	17	1 to 14	1 to 14
MVIRI	20	1 to 2	1 to 2
SEVIRI	21	4 to 11	1 to 8
GOES-Imager	22	1 to 4	1 to 4
GOES-Sounder	23	1 to 18	1 to 18
GMS/MTSAT imager	24	1 to 4	1 to 4
FY2-VISSR	25	1 to 2/4	1 to 2/4
FY1-MVISR	26	1 to 3	1 to 3
CriS	27	TBD	N/A
VIIRS	29	TBD	N/A
WINDSAT	30	1 to 16	1 to 16

**channels 19-21 are not simulated accurately*

Table 3. Instruments supported by RTTOV9 as at Feb 2008.
Sensors in italics are not yet supported by RTTOV9 but soon will be.

2.1 Simulation of clear air radiances

If *N,* the cloud cover parameter, is set to zero and the liquid water concentration profile vector is set to zero both the infrared and microwave radiances computed are for clear air with the second right hand term of equation 5 being zero. $L^{Clr}(\nu,\theta)$ can be written as:

$$L^{Clr}(\nu,\theta)=\tau_s(\nu,\theta)\varepsilon_s(\nu,\theta)B(\nu,T_s)+\int_{\tau_s}^{1}B(\nu,T)d\tau+(1-\varepsilon_s(\nu,\theta))\tau_s^2(\nu,\theta)\int_{\tau_s}^{1}\frac{B(\nu,T)}{\tau^2}d\tau \qquad (6)$$

where τ_s is the surface to space transmittance, ε_s is the surface emissivity and $B(v,T)$ is the Planck function for a frequency v and temperature T.

The transmittances, τ, are computed by means of a linear regression in optical depth based on variables from the input profile vector as described in Matricardi et. al. (2001) for RTTOV-7 predictors, Matricardi (2003) for RTTOV-8 predictors and those given in Matricardi et. al. (2005) or the RTTOV-9 science plan for RTTOV-9 predictors (only used for AIRS and IASI). The code supports any of these predictor sets with the selection being made according to the coefficient file supplied to the program. More details on the performance of the different predictor sets are given in the RTTOV9 science and validation plan.

The integration of the radiative transfer up through the atmosphere (the integrals in eq. 6) has changed slightly with RTTOV9. In RTTOV_8_7 it was assumed the atmospheric layer was sufficiently optically thin that equal weight could be given to radiance emitted from all levels within the layer, i.e. the average value of the Planck function is used. In the case of optically thick layers only the upper regions of the layer give a significant contribution to the radiance. In this case the use of the average value of the Planck function gives too much weight to the radiance coming from the lower regions of the layer. To improve the accuracy of radiance calculations in RTTOV9, we have introduced a new parameterization of the Planck function based on a linear in optical depth assumption that the source function throughout the layer is linear with the optical depth of the layer:

$$B[T(\partial)] = B_0 + (B_1 - B_0)\frac{\partial}{\partial^*} \qquad (7)$$

where B_0 is the Planck function for the top of the layer, B_1 is the Planck function at the bottom of the layer and ∂ is the optical depth of the layer. The parameterization is exact at the top (∂=0) and the bottom ($\partial=\partial^*$) of the layer. Based on Eq. (5), the radiance emitted by a homogeneous layer can be written as:

$$L(\nu,\theta) = B_0(1-e^{-\frac{\partial^*}{\mu}}) + (B_1 - B_0)\left[\frac{1-e^{-\frac{\partial^*}{\mu}}}{\frac{\partial^*}{\mu}} - e^{-\frac{\partial^*}{\mu}}\right] \qquad (8)$$

where μ is the cosine of the local path angle and ∂^* is the total optical depth of the layer. The impact of this change on the radiance calculations is documented in the RTTOV9 science and validation plan. If users wish to retain the old layer mean approximation employed in earlier versions of RTTOV they can set the flag *rt8_mode* in the *RTTOV_CONST* routine to true. This allows the RTTOV9 code to be able to exactly reproduce the RTTOV_8_7 output.

There is also an optional correction to the local path angle in RTTOV9. In principle the satellite viewing angle (or the solar zenith angle) should be converted into a local path angle that decreases with altitude because of the curvature of the Earth and its surrounding atmosphere. This effect is largest at the maximum satellite viewing angle or at the maximum solar zenith angle and is currently ignored in RTTOV_8_7 where a constant local path angle is used throughout the atmosphere. The dependence of the local path angle on altitude has been explicitly introduced in RTTOV9 by considering the geometry and the bending of rays as they traverse the atmosphere. The details are outlined in the RTTOV9 science and validation plan. Note this is not invoked if the *rt8_mode* is set to true. In addition there is also an option controlled by the *profiles(j)%addrefrac* flag to include atmospheric refraction effects. Note that unless *rt8_mode* is set to true then the latitude *profiles(1)%latitude* and surface elevation *profiles(1)%elevation* associated with the profile are also required to be set.

If reflected solar radiation is required to be included in the SWIR channels (i.e. in the range 2000-2760 cm^{-1}) then the logical flag *profiles(1)%addsolar* must be set to true and a number of additional variables need to be specified which are, solar zenith and azimuth angles *profiles(1)%sunzenangle*, *profiles(1)%sunazangle*, the satellite azimuth angle, *profiles(1)%azangle* specification of fresh or salt water *profiles(1)%skin%watertype* and surface wind and wind fetch in *profiles(1)% s2m%u, profiles(1)% s2m%v, profiles(1)% s2m%wfetc*. The computation is only performed if the solar zenith angle is less than or equal to 84^o. The satellite azimuth angle is the azimuth angle of the direction formed by the projection on the X-Y plane of the vector pointing towards the receiver. The X-Y plane coincides with the mean sea level and the Z-axis points towards the zenith. The X-axis coincides with the direction of the local parallel on the Earth's surface. It is positive if directed eastward, negative if directed westward. The Y-axis coincides with the direction of the local meridian on the Earth's surface. It is positive if directed northward, negative if directed southward. The azimuth angle is counted counter-clockwise from the X-axis. Note that reflected solar radiation can only be included for the SWIR channels of IASI and AIRS since regression coefficients are not yet available for other sensors.

To compute ε_s over water there are fast surface emissivity routines for both the infrared, ISEM, (Sherlock, 1999) and for the microwave, FASTEM-2 (DeBlonde and English, 2001) or FASTEM-3 (see RTTOV_8_7 science and validation report). These models all compute a surface emissivity for the channel of interest at the given viewing angle θ. Note that using FASTEM requires the surface wind-speed to be provided in the state vector. Over the land and sea-ice surfaces only approximate default values are provided for the surface emissivity in both the infrared and microwave (see refs above for details and Table 4). The user also has the option of providing their own estimate of surface emissivity to the model if desired (see Table 4 for input options). Note that in contrast to RTTOV-7 the coefficient file supplied defines which version of FASTEM is used.

calcemiss	RTTOV coeff file version	Input ε	Forward Output ε	Tangent Linear Output ∂ε
			INFRARED CHANNELS	
true	7 or 8 or 9	0	Land=0.98/sea-ice=0.99/ sea= ε_{ISEM}	∂ε about 0.98/0.99/ε_{ISEM}
false	7 or 8 or 9	ε_{user}	ε_{user}	∂ε about ε_{user}
			MICROWAVE CHANNELS	
true	7	-1	Land/sea-ice computed from coefs in `prof % skin % fastem(1:5)` sea= $\varepsilon_{FASTEM2}$	Land/sea-ice ∂ε about $\varepsilon_{FASTEM2}$ sea ∂ε, computed from ∂u, ∂v, ∂sst about $\varepsilon_{FASTEM2}$
true	8 or 9	0	Land/sea-ice computed from coefs in `prof % skin % fastem(1:5)` sea= $\varepsilon_{FASTEM3}$	Land/sea-ice ∂ε about $\varepsilon_{FASTEM3}$ sea ∂ε, computed from ∂u, ∂v, ∂sst about $\varepsilon_{FASTEM3}$
false	7 or 8 or 9	ε_{user}	ε_{user}	∂ε about ε_{user}

Table 4. Input and output values of ε and ∂ε arrays for infrared and microwave channels for forward and gradient surface emissivity routines

2.2 Simulation of cloudy radiances

Assuming black, opaque clouds at a single level which fill the radiometer field of view the simulation of cloud affected radiances $L^{Cld}(\nu,\theta)$ is defined as:

$$L^{Cld}(\nu,\theta) = \tau_{Cld}(\nu,\theta)\,B(\nu,T_{Cld}) + \int_{\tau_{Cld}}^{1} B(\nu,T)\,d\tau \qquad (9)$$

where τ_{Cld} (ν,θ) is the cloud top to space transmittance and T_{Cld} the cloud top temperature specified by the cloud top pressure in the input state vector. The emissivity of the cloud top is assumed to be unity which is a tolerable assumption for optically thick water cloud at infrared radiances but not valid for optically thin cloud and all cloud at microwave frequencies. For partially cloud filled fields of view equation 5 is used to combine the clear sky radiance (equation 6) and cloudy radiance (equation 9) using the fractional cloud cover *N* provided as input.

This simple cloud calculation can be used for infrared channels and single layer optically thick water clouds at mid-infrared wavelengths but for more complex cloud types and/or multi-layer clouds a new multiple scattering radiance simulation code within RTTOV has been developed and is described in the RTTOV-9 science and validation plan. Note this is different to the multiple scattering code developed for microwave precipitation described in section 2.4 although there is no theoretical reason why this code couldn't be applied to the microwave, but for the moment the relevant parameters are not supplied with RTTOV for doing microwave calculations. Users should note this is an important difference between the RTTOV_CLD wrapper for RTTOV-8 where the cloudy code could be used for cloud affected microwave radiances. This new internal multiple scattering code for the infrared uses a different approach, in that scattering effects are parametrised rather than treated explicitly, and it is currently only intended for simulating cloud-affected radiances for infrared sensors such as SEVIRI, AIRS and IASI. RTTOV-SCATT is a slower, explicit approach, but often more accurate, particularly for smaller particles. It is appropriate for microwave sensors with fewer channels.

Invoking the multiple scattering scheme within RTTOV requires additional inputs to RTTOV9 as detailed in Table 5. If the user only wants to compute clear sky radiances or simple cloudy radiances as defined above the cloud and aerosol profile variables/flags need to be set to zero/.false. The cloud profile arrays *profiles(n)%clouds(i,j)* are 2 dimensional (index,levels) where the index refers to 6 different cloud types as defined in Table 6. The first 5 are water clouds and the sixth is ice cloud. The user must fill the required cloud type column with concentration values in units of g m^{-3}. Note that a non-zero concentration can be given for ***only one cloud type*** per level. In addition to the concentration array the fractional cloud cover array also must be provided *profiles(n)%cfrac(i,j)* where 0 is no cloud and 1 is overcast at level *j*. Example profiles are given in Tables 7 and 8. For water clouds optical parameters are available for five size distributions corresponding to five different cloud types whereas for ice clouds the optical parameters are parameterized as a function of the effective diameter of the size distribution. Consequently, for ice clouds the user must choose what assumption to use to parameterise the effective diameter and must also specify which shape is to be used for the ice crystals since optical parameters are available for hexagonal ice crystals and ice aggregates as defined in Table 5. As detailed in the scientific and validation report, the computation of cloud affected radiances is performed by dividing the

Options	**Set logical flags and fill profile arrays**	**Define options to convert IWC to effective diameter**	**Define ice crystal shape**
Cloudy simulation	*lclouds=.true.* *Profiles(1)%cld_data=.true.* *Profiles(1)%cloud(i,j)*=layer mean liquid or ice water content in units of g.m^{-3}. *Profiles(1)%cfrac(i,j)*= fractional cloud cover for each layer (0-1) where *i* is the index of the cloud type (see Table 6) and *j* is the level index.	Set *profiles(1)%idg* 1=Ou and Liou (1995) 2=Wyser (1998) 3=Boudala et al (2002) 4=McFarquar et al (2003)	Set *profiles(1)%ish* 1= Hexagonal 2= Aggregates
	Set logical flags and fill profile arrays	**Define climatological profile**	
Aerosol simulation	*laerosl=.true.* *profiles(1)%aer_data=.true.* *profiles(1)%aerosols(i,j)*=layer mean number density in units of cm^{-3} where *i* is the index of the aerosol component (see Table 9) and *j* is the level index.	0=User defined profile Profiles available in file *prof_aerosl_cl.dat* to read into *profiles(1)%aerosols(i,j)* 1=Continental clean 2=Continental average 3=Continental polluted 4=Urban 5=Desert 6=Maritime clean 7=Maritime polluted 8=Maritime tropical 9=Arctic 10=Antarctic	

Table 5. Inputs required for cloud and aerosol options

Column 1:	Stratus Continental	(STCO)
Column 2:	Stratus Maritime	(STMA)
Column 3:	Cumulus Continental Clean	(CUCC)
Column 4:	Cumulus Continental Polluted	(CUCP)
Column 5:	Cumulus Maritime	(CUMA)
Column 6:	Cirrus	(CIRR)

Table 6 Cloud types available in RTTOV-9.

STCO	STMA	CUCC	CUCP	CUMA	CIRR
0	0	0	0	0	0.026
0	0	0	0	0	0
0	0	0	0	0	0
0.	0	0.26	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0.28	0	0	0	0	0
0	0	0	0	0	0

Table 7. An example of the cloud liquid/ice water content input profile in g.m^{-3} for some atmospheric layers.

STCO	STMA	CUCC	CUCP	CUMA	CIRR
0	0	0	0	0	0.8
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0.5	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0.3	0	0	0	0	0
0	0	0	0	0	0

Table 8. An example of the cloud fractional coverage input profile for some atmospheric layers.

field of view into a number of streams. The number of streams used for the scattering calculation is computed internally in *rttov_cldstr*. It is possible to reduce the number of streams by considering only those streams whose weight is larger than the variable *cldstr_threshold* defined in the module *rttov_const*. By setting *cldstr_threshold* to a negative number, all the streams will be processed. This feature should be used with caution as the sum of the weights of all streams (including the clear one) must be equal to 1. If some streams are not considered the weight of the clear stream has to be adjusted. As a consequence, if the value used for *cldstr_threshold* is set too large, this can degrade the accuracy of the results and so only very small values should be used for *cldstr_threshold* in order to only remove the streams with a very small weight.

To compute cloudy IR radiances in addition to filling the input profile structure in cloud water concentration and fractional cover for each cloud type (cloud(:,:) in g.m^{-3}and cfrac(:,:) from 0-1) the user must ensure the IR scattering coefficient file is linked to the main directory. The naming convention for the file is :
sccldcoef_meteosat_5_mviri.dat – optical parameters for water and ice cloud types
where Meteosat-5 is the sensor in this case.

2.3 Simulation of aerosol affected radiances

Using the new multiple scattering code in RTTOV9 it is now possible to simulate the effects of aerosols at infrared wavelengths. The methodology is described in the RTTOV9 science and validation plan. Again additional inputs prescribed in Table 5 are required for aerosol simulations. The mixing of the various aerosol components can be defined by the user or climatological profiles with predefined mixing can be supplied. The input profile *profiles(n)%aerosols(i,j)* where the first index is the aerosol component (currently 11 as defined in Table 10) and the second index is the level number. To include an aerosol component in the radiative transfer the user must assign the layer mean density (in units of cm^{-3}) for that component. An example of an input profile for a few layers is given in Table 9. Alternatively if a climatological profile is chosen (see Table 5 for options and RTTOV-9 science plan for more details on profiles) the user can input the climtological profile file *prof_aerosl_cl.dat* and the layer mean number densities can be scaled by a factor if required. The aerosol profiles supplied are all on 101 levels. If the user requires these profiles to be on a different set of levels (e.g. 43) there is a program *aer_clim_prof* provided which will write aerosol climatological profiles on user defined levels.

INSO	WASO	SOOT	SSAM	SSCM	MINM	MIAM	MICM	MITR	SUSO	VOLA
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
1350	0.9	0	0	0	0	0	0	0	0	0
1600	0.11	0	0	0	0	0	0	0	0	0
1800	0.12	0	0	0	0	0	0	0	0	0
2000	0.13	0	0	0	0	0	0	0	0	0
2400	0.14	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Table 9. An example of part of the input profile for aerosol components for a few layers units are number density cm^{-3}

Column 1:	Insoluble	INSO
Column 2:	Water soluble	WASO
Column 3:	Soot	SOOT
Column 4:	Sea salt (acc mode)	SSAM
Column 5:	Sea salt (coa mode)	SSCM
Column 6:	Mineral (nuc mode)	MINM
Column 7:	Mineral (acc mode)	MIAM
Column 8:	Mineral (coa mode)	MICM
Column 9:	Mineral transported	MITR
Column 10:	Sulphated droplets	SUSO
Column 11:	Volcanic ash	VOLA

Table 10 Aerosol components available in RTTOV-9.

To compute cloudy IR radiances in addition to filling the input profile structure in aerosol number density in cm^{-3} for each aerosol type (aerosols(:,:) the user must ensure the IR aerosol scattering coefficient file is linked to the main directory. The naming convention for the file is :
scaercoef_meteosat_5_mviri.dat – optical parameters for aerosol types
where Meteosat-5 is the sensor in this case.

2.4 Simulation of microwave radiances scattered by precipitation

In common with RTTOV_8_7 the new RTTOV9 has a set of wrapper routines known as RTTOV-SCATT, used to compute scattering effects from hydrometeors at microwave frequencies using the delta-Eddington approximation. This should be used if needing to simulate rain and cloud affected microwave radiances. To make the difference clear, the internal RTTOV cloud parametrisation described in section 2.2 uses a completely separate approach, in that scattering effects are parametrised rather than treated explicitly, and it is for the moment only intended for simulating cloud-affected infrared radiances.

The RTTOV-SCATT code calls RTTOV-9 for the clear air part but adds the scattering effects from water/ice in the profile. RTTOV-SCATT uses a two-independent column approximation, summarised by:

$$T_B^{Total} = (1 - C_{\max})T_B^{Clear} + C_{\max} T_B^{Rainy} \qquad (10)$$

Here, $C_{\max}$ is the maximum cloud fraction in the vertical profile and T_B is brightness temperature. RTTOV-9 is called from within RTTOV-SCATT and returns the brightness temperature of the clear sky column, T_B^{Clear}, and the profile of clear sky transmittances. RTTOV-SCATT then computes the cloudy or rainy brightness temperature, T_B^{rainy}, using the clear sky transmittances provided by RTTOV-9, and lookup tables for Mie scattering properties. Finally, equation 10 is used to linearly combine the two independent columns, producing the total brightness temperature T_B^{total}.

The input profiles are the same as for RTTOV-9 (e.g. *profile_type*; see section 6.3) but there is additional information required, principally hydrometeor profiles, supplied in *profile_cloud_type* and listed in Table 11. The *use_totalice* logical variable in *profile_cloud_type* should be set to false for separate ice and snow and to true for total ice. The two options are mutually exclusive. RTTOV-SCATT uses a slightly different level definition (compared to RTTOV-9) for the cloudy/rainy column, in which constituent and hydrometeor amounts are given on 'full' pressure levels, and they apply to a domain bounded by 'half' pressure levels. Conventionally, the bottom half level is the surface and the top half level is the top of the atmosphere. Full pressure levels are those supplied in *profile_type*, but the half level pressures need to be supplied in *profile_cloud_type*.

Profile variable	Contents
nlevels	number of atmospheric levels, which should match that supplied in the other input profiles
use_totalice	logical flag to switch between using separate ice and snow, or total ice hydrometeor types.
ph(:)	nlevels+1 of half-level pressures (hPa)
cc(:)	nlevels of cloud cover (0-1)
clw(:)	nlevels of cloud liquid water (kg/kg)
ciw(:)	nlevels of cloud ice water (kg/kg)
totalice(:)	nlevels of total ice (kg/kg)
rain(:)	nlevels of rain flux (kg/(m^2)/s)
sp(:)	nlevels of solid precipitation flux (kg/(m^2)/s)

Table 11. RTTOV-SCATT profile variables for profile_cloud_type

More details on the methodology are provided in the RTTOV_8_7 science and validation report, and in Bauer et al. (2006). An example test program is supplied *rttovscatt_test_one.F90* which users should refer to for more details on the inputs required for RTTOV-SCATT. There are also several setup routines which need to be called as defined in *rttovscatt_test.F90* for additional coefficient files required when running RTTOV-SCATT as indicated in the script *rttovscatt_test.ksh* for the Mie table values (e.g. for SSM/I it is *mietable_dmsp_ssmi.dat*). Due to their large size Mie coefficient files for microwave sensors are not supplied in the tar file but will be provided on the RTTOV-9 web page for download. These are slightly different from the RTTOV_8_7 files.

3. Current limitations of RTTOV9

There are a number of limitations of RTTOV9 the user should be aware of. Some are fundamental and some are not. The main ones are listed here:

- RTTOV9 only simulates top of atmosphere radiances from a nadir or off-nadir view which intersects with the Earth's surface (i.e. no limb paths).
- RTTOV9 only allows for water vapour, ozone, carbon dioxide, nitrous oxide, methane and carbon monoxide to be variable gases with all others included in the mixed gases transmittance calculation.
- RTTOV9 can only simulate radiances for instruments for which a coefficient file has been generated. The instruments currently supported are listed in Table 3. Only sensors with channels at wavelengths greater than 3 microns can be simulated with RTTOV9.
- The accuracy of simulations for very broad channels (e.g. SEVIRI channel 4 at 3.9 microns) is poor with significant biases noted (~1-2K) (e.g. see Brunel and Turner, 2003). This is the case for all versions of RTTOV. A work around is to use planck weighted coefficient files which are now available for some sensors.
- RTTOV9 does not include the variation of the zeeman effect with magnetic field strength for the high peaking AMSU-A and SSMIS channels. Currently no correction factor is included in the coefficient files supplied with the RTTOV-9 package.

4. Changes from RTTOV-8

There are a number of important changes from RTTOV-8 which are listed below so the user is aware of them. Note that old RTTOV-7 and RTTOV-8 coefficient files can still work with the RTTOV9 code with the exception of RTTOV-8 SSM/I, TMI, SSMIS, WINDSAT and AMSR coefficient files where the number of channels were reduced to one per channel frequency. For these sensors RTTOV-7 or RTTOV-9 coefficient files must be used. Note old RTTOV-7 files may have FASTEM-2 coefficients and so it is worth checking if you want to run with FASTEM-3 that the correct version of FASTEM is included in the coefficient file you are using.

- The polarisation index arrays which were introduced with RTTOV-8 have been removed as users found them confusing. Channel and profile index arrays are still required in the same way as for RTTOV-7.
- The above change results in the channel numbering for sensors with dual polarisation channels (e.g. SSM/I, AMSR, WINDSAT, TMI, SSMI(S)) reverting to the RTTOV-7 scheme (e.g. SSM/I has 7 channels).
- The radiative transfer integration now includes a linear in optical depth parameterisation (see above). This is invoked by default but can be switched off (e.g. to provide the same values as RTTOV-8 code) by setting the variable *rt8_mode* in *rttov_const.F90* to .true.
- The curvature of the earth and refraction can now be optionally included in the optical path calculation.
- Solar reflected radiation can now be included in the shortwave IR channels.
- There are now potentially six variable gas profiles + cloud liquid water which can be supplied to RTTOV. The gases are H_2O, O_3, CO_2, N_2O, CO, CH_4. All but water vapour are optional and are activated with logical flags (e.g. *profiles%ozone_data=.true.*) See Annex J1 for more details on the variable gas options.
- The number of levels for the input profile can now be defined by the user and the radiances and transmittances output are on user defined levels. Internally the optical depth calculation however is still defined by the number of levels in the coefficient file. 43L profiles (as for RTTOV-7) for all sensors except AIRS and IASI will remain the default for the coefficient files supplied with AIRS and IASI on 100L. Users can still provide profiles on the same levels as the coefficient files and not invoke the profile interpolation to save computation time. One advantage of using the interpolation is the jacobians are properly mapped back on the user levels which was not the case for RTTOV-7/8. Note that there is a top 'hidden' layer in the RTTOV coefficient levels which is assumed to be isothermal. If the temperature of the user profile is changing rapidly at this point errors could be introduced into the simulations for high peaking channels. It is planned to fix this in an update version of RTTOV-9.
- The optical depth prediction can be either as in RTTOV-7, RTTOV-8 or RTTOV-9. In practice the latter 2 are only used for AIRS and IASI simulations as the results for IR and MW radiometers are best for the RTTOV-7 predictors. The coefficient file supplied provides the selection of predictors.
- If the internal emissivity model is invoked the microwave emissivity values returned are now as for RTTOV-7 not RTTOV-8 which means every channel has an emissivity assigned to it.
- The methodology used for the cloudy simulation and the code have been changed completely so that the cloud simulation is now part of the main RTTOV code instead of a wrapper code and so the specification of cloud profiles is different along with several options for specifying cloud optical properties (see section 2.2). This code cannot be used for simulating microwave cloudy radiances.
- Aerosol effects in the IR can now be simulated in the same way as cloudy radiances with an aerosol input profile (see section 2.3).
- The calling interface has changed significantly from RTTOV-8 and is documented below in the annexes.
- The 2m surface humidity variable can now be an active variable in the state vector if required by setting the variable *q2m* in *rttov_const.F90* to .true.

5. FORTRAN-90 UNIX/LINUX installation instructions

RTTOV9 is designed for UNIX/Linux systems. The software is now successfully tested on Sun, Intel, IBM, NEC, HP systems and for a range of Fortran 90 compilers listed in the Makefile supplied.

The following system components are needed before running RTTOV9:

* UNIX or Linux operating system
* Fortran 90 compiler
* make utilities
* gzip and gunzip
* about 100 MBytes of free disk space is the minimum required (3.5 GBytes if you require AIRS and IASI coefficient files)
* Typically 10 Mbytes of memory are required to run the code.

Some basic information on installing the RTTOV9 Fortran 90 code in a UNIX or LINUX environment follows. This assumes the code is obtained as a compressed unix tar file via ftp or on CD-ROM from ECMWF Data Services. The file name should be *rttov91.tar.gz* and be copied to your 'top' RTTOV directory (e.g. *~user/rttov91*) from which subdirectories will be created. Text in *italics* refers to specific commands to execute during the installation or file names.

5.1 Unpacking the code

First uncompress the tar file:
gunzip rttov91.tar.gz

and expand it:
tar -xvf rttov91.tar

The following subdirectories are created and contain:

- *src* Fortran source code + make files for a variety of platforms
- *scripts* Unix test scripts for running test programs
- *data* Associated input data files required for testing
- *rtcoef_rttov7* RTTOV-7 FASTEM-1/2 coefficient files for most sensors supported (see below)
- *rtcoef_rttov8* RTTOV-8 coefficient files for FASTEM-3, rttov-8 predictors, CO_2
- *rtcoef_rttov9* RTTOV-9 coefficient files for FASTEM-3, rttov-9 predictors + trace gases + cloud/aerosol
- *rtcoef_scatt* RTTOV-9 scattering coeffs
- *test_xxxx* Empty to contain output of tests run on user's machine to compare with reftest_xxx files
- *reftest* Output of test programs run by NWP SAF to compare your output with
- *reftest_rttov9* Output of RTTOV-9 tests run by NWP SAF
- *reftest_levels* Output of RTTOV-9 to test level interpolation
- *reftest_rttovscatt* Output of RTTOV-9 to test RTTOV-SCATT
- *docs* Documentation

There is also a *readme.txt* file in the main directory which defines the RTTOV version number, creation date and contents of the tar file.

Note that to reduce the size of the tar file the AIRS and IASI coefficient files and cloud and aerosol scattering coefficient files are not included but can be downloaded from the RTTOV-9 web pages if required. Also new coefficient files with RTTOV-9 predictors for more sensors will be made available on the RTTOV-9 web page in time. Bug fixes announced will also be on the web site along with corrected code to replace the module provided in the tar file.

5.2 Compiling the code

The code must be compiled with a Fortran-90 compiler. Note there are C-style pre-processor options for the compilation of the routines. First go to the source code directory:

cd src

The Fortran-90 code consists of subroutines, interfaces and modules and 3 top level test programs (*tstrad.F90, example_fwd.F90, rttovscatt_test.F90)* in *src* for complete testing of the RTTOV and RTTOV-SCATT subroutines. The first step is to compile the code and make an executable using the makefiles supplied. Edit the file called *Makefile* in *src* so that the F90 compiler options match those available on your machine. There are a selection already in the *Makefile* for commonly used compilers and flags and these can be uncommented by the user (i.e. remove the '#' symbol at the start of the line). Note that for *rttov_scatt* code there is a F77 routine *lapack.f* used so you also need to set a F77 compiler option if you require the scattering code to be compiled. Once this is done you can run the makefile. There are various options for running it:
You can either specifically give options in command line (e.g. *make FC=frt FFLAGS=' -Am -O3 -M .)* or remove the comment '#' in the file from the definitions you want to use on your machine. If the makefile is executed with options they will be passed along to the other make files. You can run make like "*make basic*" to just compile the part of the code for RTTOV (most users) or "*make all*" to compile all the code (includes RTTOV-SCATT). The full options are:

- *make direct: only compile forward model*
- *make basic (default): compile main RTTOV code (forward, TL, AD, K)*
- *make scat: compile microwave scattering code in RTTOV-SCATT*
- *make all: compile all the code*
- *make parallel: compile a parallel version of the code (for more information on this option contact nwpsaf@metoffice.gov.uk)*

With luck the code will compile and produce an executable *tstrad.out* for the basic RTTOV tests, and/or optionally *rttovscatt_test.out* for the RTTOV-SCATT tests. Note it is only worth compiling the RTTOV scattering code if there is a requirement to use it. The Makefile should move these executable files to the *scripts* subdirectory.

If the compilation was not successful then either edit the makefile again and for example change the compiler flags until it does or if all else fails compile the code manually as follows. Note you must first compile the modules then the subroutines and program:
Step 1: *f90 –c –your flags all modules (see Makefile_lib)*
Step 2: *f90 –c –your flags tstrad.F90 + subroutines (See Makefile_lib)*
Step 3: *f90 –o tstrad.out *.o*

This should produce an executable file *tstrad.out* in your *src* directory which you should then move to your *scripts* directory as *tstrad.out.*. If there were compiler errors reported when compiling the code please report these back to the NWP SAF (see section 7) so any bugs can be fixed. It is not possible for us to test the code with every compiler.

5.3 Running the test code

There are test scripts for running the executables (*tstrad.out* etc) which are in the *scripts* directory. The controlling script is *test_fwd.ksh* for testing the forward model for all sensors and *test_rttov9.ksh* for testing the tangent linear, adjoint and K codes for the RTTOV-9 options and finally *test_rttov9_hires.ksh* for testing the high resolution sounder options. Finally there is a script to test the newly introduced profile interpolation *test_levels_full.ksh.* If you only want to use the code in forward mode and/or for 1 instrument or clear air you may wish to reduce the number of test scripts called in *test_fwd.ksh* to just test for your particular application by commenting out calls to some of the other sensor tests. You don't need to run *test_rttov9.ksh* and other scripts if you are not interested in running the TL/AD/K codes. Note that the tests for the AIRS/IASI instruments in *test_rttov9_hires.ksh* are not invoked in the scripts as you will need to download the AIRS and/or IASI coefficient files first from the RTTOV-9 web site before running the test. The script *rttovscatt_test.ksh* runs the RTTOV-SCATT test program. If you want to run all the tests (including RTTOV-SCATT you can use the script *test_all.ksh.*

The rt coefficient files for all instruments supported as listed in Table 3 and input files for running *tstrad.out (*the test program) are all in the subdirectories *rtcoef_rttov7, rtcoef_rttov8, rtcoef_rttov9* and *data* respectively except for AIRS and IASI files which must be downloaded due to their size and placed in the relevant *rtcoef_rttovx* subdirectory where *x* is the RTTOV version number in the header of the coefficient file. Output files from the runs on the NWPSAF development computer are given in the *reftest* directories. The files in *reftest* can be compared with the output produced locally (the scripts write the output to a subdirectory *test* as **.lst* files) and difference files from those in *reftest* are also created *as *.diff* files in the *test* subdirectory. To check the installation has been successful you should check the *.diff* files are all of size zero or that the differences are just due to rounding errors in some of the values. Note however the TL/AD/K test outputs from running *test_rttov9.ksh* etc will differ slightly due to machine precision differences and use of a random number generator in the test code and so typical differences between machines are shown in the listing in Table 12 and sample files provided in *reftest*. These differences are normal. Note that in addition to the *test* directory there are *test_levels, test_rttov9* and *test_rttovscatt* directories (with corresponding *reftest_xxx* directories) to test the interpolation code, TL, AD and K code, the new RTTOV-9 capability and RTTOV-SCATT respectively. The test plan document included in the RTTOV-9 documentation lists all the tests for each script.

Once the code does reproduce the results in the sample files the code can then be linked into the users own particular applications. The subroutine interfaces and file structures are described in detail below and in the annexes. For the RTTOV-SCATT tests the Mie table files (e.g. *mietable_dmsp_ssmi.dat)* are required and the two output files (*.out) in the *test_rttovscatt* directory should be similar to those in the *reftest_rttovscatt* directory with only minor changes to the least significant figures.

```
<  BRUTE FORCE:      -0.2829711887E+02    0.1000000381E+01        6
<  BRUTE FORCE:      -0.2829710928E+02    0.1000000043E+01        7
<  BRUTE FORCE:      -0.2829711349E+02    0.1000000191E+01        8
<  BRUTE FORCE:      -0.2829712571E+02    0.1000000623E+01        9
<  BRUTE FORCE:      -0.2829696655E+02    0.9999949986E+00       10
<  BRUTE FORCE:      -0.2829438017E+02    0.9999035979E+00       11
<  BRUTE FORCE:      -0.2829665391E+02    0.9999839502E+00       12
<  BRUTE FORCE:      -0.2862066140E+02    0.1011434148E+01       13
<  BRUTE FORCE:      -0.3012701200E+02    0.1064667524E+01       14
<  BRUTE FORCE:      -0.2273736754E+02    0.8035226599E+00       15
---
>  BRUTE FORCE:      -0.2829711886E+02    0.1000000381E+01        6
>  BRUTE FORCE:      -0.2829710937E+02    0.1000000046E+01        7
>  BRUTE FORCE:      -0.2829711178E+02    0.1000000131E+01        8
>  BRUTE FORCE:      -0.2829712855E+02    0.1000000724E+01        9
>  BRUTE FORCE:      -0.2829702339E+02    0.9999970074E+00       10
>  BRUTE FORCE:      -0.2829381174E+02    0.9998835099E+00       11
>  BRUTE FORCE:      -0.2828812740E+02    0.9996826292E+00       12
>  BRUTE FORCE:      -0.2859223969E+02    0.1010429745E+01       13
>  BRUTE FORCE:      -0.3069544618E+02    0.1084755591E+01       14
>  BRUTE FORCE:      -0.1136868377E+02    0.4017613299E+00       15
872,873c872,873
<  PROFILE= 1 SUMRAD=   -0.7225097989E+01 SUMPROF=   -0.7225097989E+01
<  PROFILE= 2 SUMRAD=   -0.5618514327E+01 SUMPROF=   -0.5618514327E+01
---
>  PROFILE= 1 SUMRAD=   -0.7074762965E+01 SUMPROF=   -0.7074762965E+01
>  PROFILE= 2 SUMRAD=   -0.1075604858E+02 SUMPROF=   -0.1075604858E+02
At head of TL-AD and TL-K water vapour output small changes on a few rows:
<    3    0    0    0    0    0    0    0    0    0    0    0    3    0    0    0    0
---
>    3    0    0    0    0    0    0    0    0    0    0   -1    2    0    0    0    0
```

Table 12. Example of typical differences found between NWP SAF generated output and that from the users machine. The numbers can differ from run to run but are typically just in the least significant digits..

6. Running RTTOV9 for your applications

To run RTTOV9 for a user's application the test programs supplied **example_fwd.F90, tstrad.F90** can be used as a guide or template. Programs should be compiled with the C-style preprocessor options enabled **parkind1** so you can make use of the #include statements for the subroutines declarations. Note for most compilers this implies you need a *.F90* as the file extension which is what is provided to the users. For users with HP compilers it may be necessary to convert the *.F90* file extensions to *.f90* for all the routines. Use the modules **rttov_types** and **rttov_const** in your program for the definition of derived types (see annexes J and K). It is also important to allocate the various input and output arrays for **rttov_direct** to the correct dimensions (see **example_fwd or tstrad** for examples). Figure 1 gives a process diagram of what routines to call when running RTTOV9. Annex L gives an example program with comments to guide the user and this source code is provided in the tar file.

There are only 2 subroutines that must be called to run RTTOV9 which are **rttov_setup** which does all the general setup tasks, and the call to **rttov_direct** or **rttov_tl** or **rttov_ad** or **rttov_k** as required to compute the radiances or TL/AD/K outputs. Optionally for more flexibility **rttov_setup** can be replaced by the individual setup routines, **rttov_errorhandling** to set up the error message and level of verbosity, **rttov_readcoeffs** to read in the coefficients requested, **rttov_initcoeffs** to initialise coefficient arrays and to distribute on different processors for a MPP platform and **rttov_errorReport** to feedback information on any errors. There are 3 routines which do the allocation and deallocation of the various input/output arrays for RTTOV which are **rttov_alloc_prof, rttov_alloc_rad** and **rttov_dealloc_coef.**

It is recommended that users look at the header section of the coefficient file for the sensor they wish to simulate as there is useful information there such as the definition of channel numbers and the polarisation assumed for each channel for that instrument etc. The following steps are recommended in coding a program which calls RTTOV9.

6.1. Access to coefficients, initialisation

- Allocate the coefficient structure for the desired number of instruments you want to run inside your program.
- Initialise the logical unit for error/warning messages and the verbosity level. This is performed by **rttov_handling**, an optional routine which can be called at any time (see Annex A).
- The command **rttov_setup** is a general tool which includes a call to **rttov_errorhandling** and calls to **rttov_readcoeff** and **rttov_initcoeffs** for several coefficient files (see Annex E).

See the **test_coef** and **test_2_coef** main programs for an example of the different ways to read coefficients (ascii, binary, already opened or with a list of channels useful for AIRS/IASI to save reading in coefficients for all the channels). If fast performance is required for reading the coefficient files, it is better to access binary coefficient files. The user can use the **rttov_ascii2bin_coef** tool to convert the ASCII files provided to binary files on their local machine. The script converts all coefficient files which are present in one directory (edit the script to change directory). Take care of the compilation options because the user should always ensure that the compilation of the binary file creation program is consistent with the compilation for RTTOV. The **rttov_readcoeff** reads headers for checking the single/double precision and normally will give an error message if an incompatible binary coefficient file is being read, but this may not be fully failsafe. There are several options the user should be aware of in choosing a coefficient file for RTTOV9. These are defined in Tables 13 and 14 below. The infrared surface emissivity model ISEM is unchanged in RTTOV-9 and is invoked by setting *calcemiss* to .true. Similarly the microwave surface emissivity model has not changed as FASTEM-3 and is invoked in the same way.

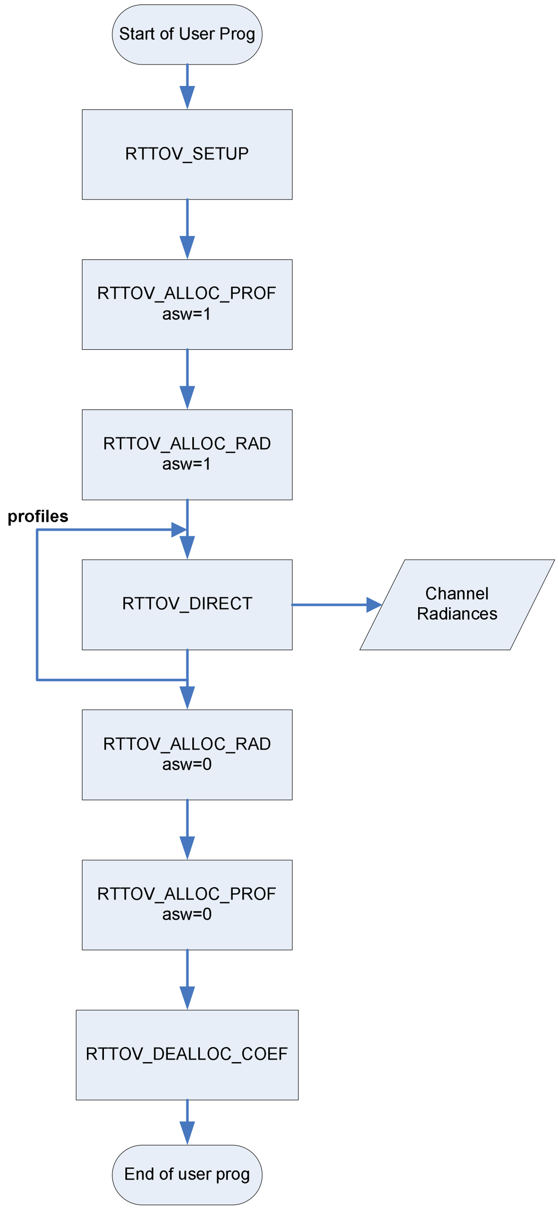

Figure 1. Process diagram of user program calling RTTOV9 forward model.

Surface emissivity model options	RTTOV-7 coefficients	RTTOV-8 coefficients	RTTOV-9 coefficients
Microwave emissivity			
FASTEM-2	set *calcemiss* to .true.	Not available unless modify coeff file to invoke FASTEM-2	Not available unless modify coeff file to invoke FASTEM-2
FASTEM-3	Not available	set *calcemiss* to .true.	set *calcemiss* to .true.
Infrared emissivity			
ISEM-6	set *calcemiss* to .true.	set *calcemiss* to .true.	set *calcemiss* to .true.

Table 13. Coefficient file options for surface emissivity in RTTOV9.

There are changes to the gaseous absorption options in RTTOV-9 as the user can now compute transmittances for more trace gases (see Table 14). There are 2 points for the user to be aware of in specifying what gases need to be included in the computation. The first is that the profile flag for the gas of interest must be set to .true. The second is that the coefficient file supplied must contain the coefficients for the gas of interest. Obviously the fewer gases simulated the faster the code will run. The options for the different versions of the coefficient files, all of which can be read by the RTTOV9 code are given in Table 14.

	RTTOV-7 Coefficients		RTTOV-8 coefficients		RTTOV-9 coefficients	
Gas	Profile	Coeffs	Profile	Coeffs	Profile	Coeffs
Mixed	Y	Y	Y	Y	Y	Y
H2O	Y	Y	Y	Y	Y	Y
O3	Y	O	O	O	O	O
CO2	N	N	O	O	O	O
N20	N	N	N	N	O	O
CO	N	N	N	N	O	O
CH4	N	N	N	N	O	O

Y=Input mandatory

O=Input optional depending on profile flag and contents of coeff file

N=No input possible

Table 14. Coefficient file options for gaseous absorption

It is important to be aware that if the coefficient file being used contains coefficients for a particular gas e.g. *coef(no_id)%nozone* > 0 then a profile must be supplied for that gas and the profile flag set to .true. This is necessary because the coefficients for the fixed gases will not include any gases which have specific variable gas coefficients and so the calculation will be in error if the variable gas calculation is not included. Note all the microwave coefficients do not include ozone as an active gas.

6.2. Setting up input arrays before each call to RTTOV

```
call rttov_direct(rttov_errorstatus, nprof, nchannels, channels,
lprofiles, addinterp, profiles, coef, coef_scatt_ir, optp,
lsun, laerosl, lclouds, calcemis, emissivity, transmission, radiance )
```

Setting up the input arrays should be simpler for RTTOV-9 than RTTOV-8 and follows closely the RTTOV-7 guidance. Table 15 gives examples of these arrays for three different sensors and for 2 profiles per RTTOV call. **nchannels** is the number of required channels for the sensor **multiplied** by **nprof** which is the number of profiles**.** The arrays **channels and lprofiles** contain the corresponding channel and profile indices for each computed radiance. All infrared sensors will be like HIRS or AVHRR. ***Note for SSM/I or other dual polarised sensor a RTTOV-8 coefficient file will not work with RTTOV-9. An RTTOV-7 coefficient file should work however.***

If the logical array **calcemiss** is set to *.false*. for a channel the emissivity array **emissivity(nchannels)** must contain the surface emissivity to be used for that channel on input and if set to *.true.* the ISEM model is used as defined in Table 13. For microwave channels again if **calcemiss** is set to *.false*. the emissivity array **emissivity(nchannels)** must contain the surface emissivity to be used for that 'channel' on input (and if set to *.true.* the FASTEM model is used). The **lclouds** and **laerosl** flags are logical switches which are set to false unless the user desires cloudy radiances or aerosol affected radiances to be computed.

Input arrays	HIRS (2 profiles/call)	SSM/I (2 profiles/call)	AMSU-B (2 profiles/call)
nchannels	38	14	10
nprof	2	2	2
channels(nchannels)	1,2,3 ..,19,1,2,3..,19	1,2,3,4,5,6,7,1,2,3,4,5,6,7	1,2,3,4,5,1,2,3,4,5
lprofiles(nchannels)	1,1,1,…,1,2,2,2…,2	1,1,1,1,1,1,1,2,2,2,….,2	1,1,1,1,1,2,2,2,2,2

Table 15. Examples of RTTOV9 input parameters

The **coef** array contains the RT optical depth coefficients read from the coefficient file. The **coef_scatt_ir,** and **optp** arrays contain the cloud and aerosol scattering coefficients and optical properties which are required if **lclouds** and **laerosl** flags are set to .true. These arrays are initialised during the call to **rttov_setup**.

6.3. Setting up input profile for RTTOV9

The users input profile is normally on different pressure levels to that required internally by RTTOV. Previous versions have required the user to interpolate/extrapolate their profile on to the fixed RTTOV pressure levels. However RTTOV9-1 does include an option to interpolate the input user profile on defined pressure levels **`profiles`** to the required RTTOV pressure levels defined in the coefficient file used. The method of interpolation is given in Rochon *et. al.* (2007) and also described in the RTTOV-9 science and validation report. The array **`profiles`** should contain the variables given in Table 16 some are mandatory as indicated and some are optional depending on what is required and also the coefficient file used. Note it is advisable to at least initialise all the profile variables to zero or .false. The example program in Annex L is also a useful guide on how to set up the profile arrays. If you use a coefficient file with trace gas coefficients (i.e. AIRS and IASI) and don't have the trace gas profile concentrations then you can set the variable *profiles(1) % co_data* for example for CO to false and the reference CO profile is then used for the calculation. If the variable is true however and the CO profile contains zeroes then the program will abort. This applies to all gases except water vapour which is always mandatory.

Note that profiles input can be on different levels but for one call to RTTOV-9 the number of levels must remain the same. If the user profile does not encompass all the pressure levels required in the coefficient profile levels the top level user profile is copied to all coefficient pressure levels above the user profile in order to fill the coefficient profile array and similarly for any coefficient levels required below the user profile. The non-mandatory variables are only required for certain options such as including the solar reflected term, allowing for refraction in the atmospheric path term, including aerosol and/or cloud scattering. The right hand column of Table 16 indicates when these optional parameters become mandatory.

There are several options on how to deal with input profiles that fall outside the regression limits for the clear-sky optical depth calculations. The clear-sky optical depth calculations are based on regressions derived from line-by-line calculations, and the validity limits for the regressions are given in the coefficient files. If *apply_reg_limits* in *rttov_const* is set to false, RTTOV-9 will return a warning *errorstatus* whenever the input profiles are outside these regression limits, but RTTOV-9 will nevertheless perform the full radiative transfer calculation, using the profile values outside the regression limits. If *apply_reg_limits* is set to true the profiles used for the optical depth calculations are reset to the limit values if some input values fall outside the regression limits and no warning code is returned. Note that the regression limits are applied to the clear-sky optical depth calculations only; the source function for the radiative transfer equation will still be based on unadjusted user input profiles. The user is advised to perform their own sensitivity studies to decide which option works best for their given application.

There are several points which need to be considered if the new internal profile interpolation is used (i.e. the input profile is on different levels to the coefficient file). Points to note are:

i. The user profile should cover the whole atmosphere covered with an adequate number of levels at least close to the coefficient levels or more. A coarse layering will reduce the accuracy of the calculations.
ii. The user profile lowest level should be equal or below the surface pressure
iii. If the user profile is below the top of the coefficient file the user profile is extrapolated assuming a constant value of the uppermost user value for all levels above the top.
iv. If the interpolation is used profiles_tl % p can be non-zero (and for sigma levels should be). If the interpolation is not used, input levels are assumed to be on fixed pressure levels, so profiles_tl % p should be zero. Also applies to _ad and _k codes.

6.4. Output arrays from RTTOV9

```
call rttov_direct(rttov_errorstatus, nprof, nchannels, channels,
lprofiles, addinterp, profiles, coef, coef_scatt_ir, optp,
lsun, laerosl, lclouds, calcemis, emissivity, transmission, radiance )
```

The **errorstatus** array contains an error code for each profile which if greater than 0 indicates a problem with that profile together with an error message output. Depending on the verbosity level set in **rttov_setup** (annex E) messages will be printed on the output logical unit to explain the error. Examples are:

- 0 = Computation OK
- 1 = FATAL error which mean that the profile should be aborted (e.g. unphysical profile input)
- 2 = WARNING an error which can allow the computation to continue but the results may be suspect (e.g. profile outside basis profile limits)

Input profile arrays	Description	Units	Mandatory?	When mandatory
profiles(i) % nlevels	Number of pressure levels		Y	
profiles(i) % p(:)	Pressure levels	hPa	Y	
profiles(1) % t(:)	Temperatures on levels	K	Y	
profiles(i) % q(:)	Water vapour conc on levels	ppmv	Y	
profiles(i) % o3(:)	Ozone conc on levels	ppmv	N	If flag .true.
profiles(i) % co2(:)	CO_2 conc on levels	ppmv	N	If flag .true
profiles(i) % n2o(:)	N_2O conc on levels	ppmv	N	If flag .true
profiles(i) % co(:)	CO conc on levels	ppmv	N	If flag .true
profiles(i) % ch4(:)	CH_4 conc on levels	ppmv	N	If flag .true
profiles(i) % aerosols(:,:) *(2nd array element is levels)*	Aerosol components conc on levels	cm^{-3}	N	Aerosol scatt
profiles(i) % cloud(:,:)	Cloud water/ice content on levels	$g.m^{-3}$	N	Cloud option
profiles(i) % cfrac(:,:)	Cloud fractional cover on levels	0-1	N	Cloud option
profiles(i) % clw(:)	Microwave cloud liquid water	Kg/kg	N	Microwave
profiles(i) % idg	IWC eff diam scheme	1-4	N	Cloud option
profiles(i) % ish	Ice crystal shape	1-2	N	Cloud option
profiles(i) % s2m	2m surface variables (see annex J)		Y	
profiles(i) % skin	Skin surface variable (see annex J)		Y	
profiles(i) % ctp	Cloud top pressure for simple cloud	hPa	N	Simple cloud
profiles(i) % cfraction	Cloud fraction for simple cloud	0-1	N	Simple cloud
profiles(i) % ozone_data	Flag to indicate profile present	logical	Y	
profiles(i) % co2_data	Flag to indicate profile present	logical	Y	
profiles(i) % n2o_data	Flag to indicate profile present	logical	Y	
profiles(i) % co_data	Flag to indicate profile present	logical	Y	
profiles(i) % ch4_data	Flag to indicate profile present	logical	Y	
profiles(i) % aer_data	Flag to indicate profile present	logical	Y	
profiles(i) % cld_data	Flag to indicate profile present	logical	Y	
profiles(i) % clw_data	Flag to indicate profile present	logical	Y	
profiles(i) % zenangle	Satellite zenith angle	deg	Y	
profiles(i) % azangle	Satellite azimuth angle	deg	N	Solar option
profiles(i) % sunzenangle	Solar zenith angle	deg	N	Solar option
profiles(i) % sunazangle	Solar azimuth angle	deg	N	Solar option
profiles(i) % latitude	Latitude	deg	Y	not mandatory if rt8_mode true
profiles(i) % elevation	Elevation	km	Y	not mandatory if rt8_mode true
profiles(i) % addsolar	Include solar reflection for IR	logical	Y	
profiles(i) % addrefrac	Include atmospheric refraction	logical	Y	

Table 16. Profile input parameters for user profile i

Annex J defines fully the output radiance, emissivity and transmittance type structures. Table 17 defines in more detail which arrays are used for output by users and their dimensions for **rttov_direct** and gradient routines. There can be confusion in the role of the surface pressure in the output radiance quantities on layers (e.g. *rad%overcast* etc). All values relate to the standard pressure levels defined by the user *except* for the layer containing the surface pressure where the level below the surface pressure contains the radiances from the surface level defined by the surface pressure not the profile level.

Radiance_Type Radiances in units of *mw/cm-1/ster/sq.m*		
Type	**Array name**	**Contents**
real	clear(nchannels)	Clear sky top of atmosphere radiance output for each channel
real	total(nchannels)	Clear+cloudy top of atmosphere radiance for given cloud top pressure and fraction for each channel
real	cloudy(nchannels)	Cloudy top of atmosphere radiance for 100% fraction for each channel at given cloud top pressure
real	upclear(nchannels)	clear sky upwelling radiance without reflection term
real	dnclear(nchannels)	clear sky downwelling radiance
real	overcast(nlev,nchannels)	Level to space overcast radiance given black cloud on each level (levels,channels)
real	up(nlev,nchannels)	Above cloud upwelling atmospheric radiance for each pressure level down to surface
real	down(nlev,nchannels)	Above cloud downwelling atmospheric radiance for each pressure level down to surface
real	surf(nlev,nchannels)	Radiance emitted by a black cloud at each pressure level except for the surface
real	downcld(nlev,nchannels)	Contribution to radiance of downward cloud emission at given cloud top
real	surf(nlev,nchannels)	Radiance emitted by a black cloud at each pressure level except for the surface
Radiance_Type Brightness Temperatures *degK*		
real	bt(nchannels)	BT equivalent to total (clear+cloudy) top of atmosphere radiance output for each channel
real	bt_clear(nchannels)	BT equivalent to clear top of atmosphere radiance output for each channel
Transmission_Type Transmittances 0-1		
real	tau_total(nchannels)	transmittance from surface for each channel
real	tau_layers(nlevels,nchannels)	Transmittance from each standard pressure level to top of atmosphere for each channel
Emissivity 0-1		
real	emissivity(nchannels)	Input surface emissivity values for *calcemis=.false.* Output emissivity vales for *calcemis=.true.*

Table 17. Main RTTOV9 output arrays. The green rows are those commonly used.

6.5. Running RTTOV9

You need to ensure the following in your program which calls RTTOV9.

- Before compilation check the values in the RTTOV constants file **rttov_const.F90** (see Annex K) are as required (normally this should only be the few logicals indicated in the code for user selection).
- Allocate the input/output structures to RTTOV with the number of channels, internally computed radiances, output radiances and profiles you want to run with and by the number of fixed pressure levels of the coefficients. See above and Annex F for a detailed description of the variables required for input to RTTOV9 and Annex L for example code.
- Initialise the profile variables, these are defined in the **rttov_types** module and listed in Annex J. Be careful that units for gases water vapour and ozone etc are volume mixing ratio (ppmv) and not specific concentration (kg/kg) as for RTTOV-7. You may give a surface emissivity value for each radiance calculation, but you may also let the code compute it by the use of the models ISEM (IR over ocean) and FASTEM (MW). In this case, you have to initialise the logical calculation of surface emissivity flag (*calcemiss*) to true for each channel. You can also specify

whether aerosol and/or cloudy calculations are to be performed by use of the logical flag *lcloud* and *laerosl* which should be set to false unless cloudy or aerosol scattered radiances are required.

- Ensure the variables *profiles(i)%zenangle* and *profiles(i)%azangle* contain the satellite zenith angle at the surface and satellite azimuth angle at the surface (from north – east is +90 and west is +270) for each profile. Note the latter can be set to zero unless FASTEM-3 is required or the reflected solar or cloudy/aerosol options are used.
- Ensure the variables *profiles(1) % ozone_data, profiles(1) % clw_data, profiles(1) % co2_data, profiles(1)% co_data, profiles(1) % n2o_data, profiles(1) % ch4_data* are set either 'true' or 'false' depending on whether you want to provide a concentration profile for each constituent or not.
- Make sure the coefficient file for the instrument you want to simulate is in the same directory as the executable (or better a symbolic link to the filename is made in the directory).
- Call RTTOV (**rttov_direct**) with the input/output variables and with the coefficient structure corresponding to the instrument you want to simulate.
- When all RTTOV calls are made, then you should free memory by de-allocating the coefficient structure with the **rttov_dealloc_coef** , **rttov_alloc_prof** and **rttov_alloc_rad** routines. See example in Annex L.
- All user's level RTTOV routines return an error status. This variable should be tested after each call and compared with the different error levels described in the module **rttov_const** or with 0 which is the "no error" value.
- If you want to allow for variation in the path length follow the guidance in section 2.1.
- If you want to include reflected solar radiation for the SWIR channels follow the guidance in section 2.1.
- If you want to run the cloud or aerosol options follow the guidance in sections 2.2/2.3.
- The **rttov_scatt** routines are a level up from **rttov_direct** but they have almost the same calling structure and arrays to fill. The test program supplied **rttovscatt_test** can be used as an example and similar rules apply to calling **rttov_direct**. Remember to compile using the *make scat* or *make all* option.

7. Reporting and known bugs for RTTOV9

The procedure to report bugs or make comments on the code to the NWP-SAF is as follows:

Send a bug report to nwpsaf@metoffice.gov.uk including the following information:

- RTTOV version number (i.e. 9_1)
- Platform and operating system you are running the code on (e.g. HP, Sun, LINUX PC)
- Compiler used (e.g. *g95, ifort, pgf90*, etc)
- Classification of report as: serious, cosmetic or improvement
- Report of problem including any input / output files the SAF can use to reproduce the problem

Once the problem has been analysed it will be posted on the RTTOV web site with a description of the fix if appropriate. There is also a RTTOV-9 (and v7 and v8) email list which you can subscribe to by sending an email to mailto:nwpsaf@metoffice.gov.uk where bugs are announced. If you request the code and sign a licence agreement you will be automatically included on this list.

There are several known bugs in the version ***rttov9*** of the code which are listed below. Corrections to these will be provided via the RTTOV-9 web page http://www.metoffice.gov.uk/research/interproj/nwpsaf/rtm/rtm_rttov9.html as they become available. They are:

i. The code is only partially optimised for vector machines. More work is underway by developers on NEC and IBM, platforms to further optimise the performance for all machines.

ii. Note that for compilation on the NEC several routines have to be compiled with the Cvsafe flag rather than Chopt. They are *rttov_tl, rttov_alloc_prof and rttov_alloc_predictor*. Also note that the rttov_integrate_xx routines require the flag -DRTTOV_ARCH_VECTOR. For RTTOV-SCATT *rttov_ad* also had to be compiled Cvsafe. The Makefile provides all compiler options.

iii. For polarimetric sensors (i.e. Windsat) the TL/AD/K code is not working. A fix will be released shortly.

iv. When the internal interpolation is activated RTTOV simulations close to and above the top user level may show slightly incorrect sensitivity to the top-most user input levels. This is because the internal interpolation keeps the total optical depth constant above the top-most coefficient level when interpolating optical depths to user levels. As the optical depth is non-zero for the top-most coefficient level due to the 'hidden' level, all user layers above the top-most coefficient level will have a layer optical depth of zero, except the top-most user layer. This feature will be fixed as part of the 'hidden' level update.

8. Frequently asked questions

This section will be updated on the web pages from time to time.

1. Do I need to bother to upgrade my version of RTTOV_8_7 to RTTOV9? If you want any of the following the answer is yes (if you have v7 there are more reasons) :
   - *Provide profiles on user defined levels and Jacobian returned takes account of adjacent levels so no 'blind' levels as for earlier version. This is important for assimilation where number of NWP pressure levels > 50.*
   - *Improved clear air simulations for water vapour, ozone and CO_2 due to improved predictors, linear in optical depth assumption for planck fn and inclusion of zenith angle dependence of path length due to refraction.*
   - *Additional trace gases can now be variable for AIRS and IASI i.e. CO, N_2O, CH_4,*
   - *Inclusion of reflected solar radiation for SW infrared channels*
   - *Improved simulations of IR radiances affected by multi-layer cloud and simpler interface using only RTTOV.*
   - *Simulations of IR radiances affected by aerosols*
   - *Improved microwave scattering code*

2. Can I compile the code in single precision to save space? *Yes for RTTOV9 the precision of the variables are defined by the parkind module. This file needs to be edited to change the precision using the JPRB variable.*

3. I don't have an ozone or CO_2 , CO, N_2O, CH_4 or cloud liquid water profile to include in the state vector. What can I do? *You can either make sure you have a coefficient file which includes the above in the mixed gases (apart from CLW) or you can still use a coefficient file with minor gas coefficients and the reference profile supplied with the coefficients is used. In both cases you must set the logical flags for example profiles(1) % co_data etc to false. If you don't the program assumes you are providing a valid profile and will fail if you don't.*

4. Why do the numbers in the ***test_rttov9.ksh*** output (see Table 12) change from run to run? *A random number generator is included in the code so different values can be expected. The important thing is SUMPROF=SUMRAD to machine precision.*

5. My profile top is below the top level required by RTTOV_9_1, how do I best extrapolate it for RTTOV? *For gas concentrations it is best to use the reference profile which is available by the coef structure e.g. for water vapour coef%ref_prfl_mr(:,coef%fmv_gas_pos(gas_id_watervapour)) . For temperature one can extrapolate from the top level of the NWP model using a representative lapse rate from standard atmospheres.*

Good Luck and please provide the NWPSAF with any feedback on your experiences. Remember do not pass this code on to anyone else without the permission of EUMETSAT by getting the new user to complete the on-line licence agreement at http://www.metoffice.gov.uk/research/interproj/nwpsaf/request_forms/request_rttov_9.html. The code is provided to you on an "as is" basis and there is no commitment to maintain it although we will try our best within the given resources.

9. References

Bauer P., E. Moreau, F. Chevallier, and U. O'Keeffe 2006 Multiple-scattering microwave radiative transfer for data assimilation applications. *Q.J.Royal. Meteorol. Soc.,* **132**, 1259-1281.

Boudala, F.S., Isaac, G.A., Fu, Q., and Cober, S.G., 2002: Parameterization of effective ice particle size for high latitude clouds. *Int. J. Climatol*., **22**, 1267-1284.

Brunel, P. and S. Turner 2003 On the use of Planck-weighted transmittances in RTTOV presented at the 13th International TOVS Study Conference, Ste Adele, Canada 29 Oct – 4 Nov 2003. http://cimss.ssec.wisc.edu/itwg/itsc/itsc13/thursday/brunel_poster.pdf

DeBlonde, G. and S.J. English 2001 Evaluation of the FASTEM-2 fast microwave oceanic surface emissivity model. Tech. Proc. ITSC-XI Budapest, 20-26 Sept 2000 67-78

Eyre J. R. 1991 A fast radiative transfer model for satellite sounding systems. ECMWF Research Dept. Tech. Memo. 176 (available from the librarian at ECMWF).

Matricardi, M., F. Chevallier and S. Tjemkes 2001 An improved general fast radiative transfer model for the assimilation of radiance observations. ECMWF Research Dept. Tech. Memo. 345. http://www.ecmwf.int/publications

Matricardi, M. 2003 RTIASI-4, a new version of the ECMWF fast radiative transfer model for the infrared atmospheric sounding interferometer. ECMWF Research Dept. Tech. Memo. 425. http://www.ecmwf.int/publications

Matricardi, M., 2005 The inclusion of aerosols and clouds in RTIASI, the ECMWF fast radiative transfer model for the Infrared Atmospheric Sounding Interferometer. *ECMWF Technical Memorandum 474* (available at: http:/www.ecmwf.int/publications/library/references/list/14)

McFarquhar, G.M., Iacobellis, S. & Somerville, R.C.J., 2003 : SCM simulations of tropical ice clouds using observationally based parameterizations of microphysics. *J. Clim.,* **16,** 1643-1664.

Ou, S. & Liou, K.-N., 1995: Ice microphysics and climatic temperature feedback. *Atmos. Res.,* **35,** *127-138*.

Rochon, Y., L. Garand, D.S. Turner and S. Polavarapu. 2007: Jacobian mapping between vertical co-ordinate systems in data assimilation. *Q.J.Royal Meteorol. .Soc.* **133** *1547-1558.*

Saunders R.W., M. Matricardi and P. Brunel 1999 An Improved Fast Radiative Transfer Model for Assimilation of Satellite Radiance Observations. *Q.J.Royal Meteorol. .Soc.* , **125**, *1407-1425*.

Sherlock, V. 1999 ISEM-6: Infrared Surface Emissivity Model for RTTOV-6. NWP SAF report. http://www.metoffice.gov.uk/research/interproj/nwpsaf/rtm/papers/isem6.pdf

Wyser, K., 1998: The effective radius in ice clouds. *J. Clim.,* **11,** 1793-1802.

Annex A: RTTOV_ERRORHANDLING interface

Call **rttov_errorhandling** (err_unit, verbosity_level, print_checkinput_warnings)

rttov_errorhandling may be called at any time. The purpose is to define the level of information messages are sent to which logical unit. The verbosity level allows the user to get various level of error messages or all the information. The logical error unit defines the fortran file unit number on which messages are written. The default value is the one given in the rttov_const file, on most computers the standard error is 0, but for HP it is 7. The user should set the value according to his system. If no call is made, it is the same as calling the routine with the default values. The number of error messages output is also controlled.

Subroutine Arguments:

Type	In/Out	Variable	Description
Integer	Intent (in)	err_unit	Logical error unit
Integer	Intent (in)	verbosity_level	0 = no error messages output 1 = FATAL errors only printed. these are errors which mean that profile should be aborted (e.g. unphysical profile input) 2 = WARNING errors only printed. Errors which can allow the computation to continue but the results may be suspect (e.g. profile outside basis profile limits) 3 = INFORMATION messages which inform the user about the computation Any other value is treated as 3
Logical (optional)	Intent (in)	print_checkinput_warnings	This can be used to ensure only a limited number number of warning messages are output. If set to .true. get all messages printed but if set to .false. the messages are not printed. The default is .true. See *rttov_checkinput.F90* for an example of this flag being used.

Annex B: RTTOV_ALLOC_PROF interface

```
call rttov_alloc_prof ( &
        errorstatus,   ! out
        nprof,         ! in
        profiles,      ! in/out
        nlevels,       ! in
        coef_scatt_ir, ! in
        asw,           ! in
        addaerosl,     ! in (optional)
        addclouds, ,   ! in (optional)
        init, ,        ! in (optional)
        blob) ,        ! in (optional)
```

rttov_alloc_prof is called in the users program to allocate or deallocate the memory for the profiles structure.

Subroutine Arguments:

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus	Error return code
Integer	Intent(in)	nprof	Number of profiles
Type(profile_type)	Intent (inout)	profiles(nprof)	Profiles structure to be allocated
Integer	Intent (in)	nlevels	Number of profile levels
Type(rttov_coef_scatt_ir)	Intent (in)	coef_scatt_ir	IR scattering coeff structure
Integer	Intent (in)	asw	Switch (1=allocate; 0=deallocate)
Logical	Intent (in) optional	addaerosol	Allocate aerosol structures
Logical	Intent (in) optional	addclouds	Allocate cloud structures
Logical	Intent (in) optional	init	Additionally initialise arrays
Logical	Intent (in) optional	blob	Allocate space via blob

Annex C: RTTOV_ALLOC_RAD interface

```
call rttov_alloc_rad ( &
      errorstatus,   ! out
      nchannels,     ! in
      radiance,      ! in/out
      nlevels,       ! in
      asw)           ! in
```

rttov_alloc_rad is called in the users program to allocate or deallocate the memory for the radiance structure.

Subroutine Arguments:

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus	Error return code
Integer	Intent(in)	nchannels	Number of channels
Type(profile_type)	Intent (inout)	profiles(nprof)	Profiles structure to be allocated
Integer	Intent (in)	nlevels	Number of levels
Integer	Intent (in)	asw	Switch (1=allocate; 0=deallocate)

Annex D: RTTOV_DEALLOC_COEF interface

Call **rttov_dealloc_coef** (errorstatus, coef, coef_scatt_ir,optp)

rttov_dealloc_coef is called in the users program to deallocate the memory for the coefficients structure.

Subroutine Arguments:

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus	Error return code
Type(rttov_coef)	Intent (in)	coef	Coefficient structure
Type(rttov_coef_scatt_ir)	Intent (in)	coef_scatt_ir	IR scattering coeff structure
Type(rttov_optpar_ir)	Intent (in)	optp	Optical parameters structure

Annex E: RTTOV_SETUP interface

```
Call rttov_setup (          &
    & errorstatus,          & ! out
    & err_unit,             & ! in
    & verbosity_level,      & ! in
    & ninst,                & ! in
    & laerosl,              & ! in
    & lcloud,               & ! in
    & coef,                 & ! in
    & coef_scatt_ir,        & ! out
    & optp,                 & ! out
    & instrument,           & ! in
    & channels              ) ! in Optional
```

Rttov_setup is called only once per main program. It defines the logical unit and verbosity level for information messages (see rttov_errorhandling) and it reads the coefficients for a set of instruments and an optional list of channels. The routine 'creates' the filename of the coefficient files.
If "channels" is present, only the corresponding list of channels (all >0 values) is extracted from the coefficient file.

Subroutine Arguments:

Type	In/Out	Variable	Description
Logical	Intent(in)	laerosl	Logical flag for aerosol ir calculations
Logical	Intent(in)	lcloud	Logical flag for cloudy ir calculations
Integer	Intent (in)	err_unit	Logical error unit
Integer	Intent (in)	verbosity_level	0 = no error messages output 1 = FATAL errors only printed. 2 = WARNING errors only printed. 3 = INFORMATION messages Any other value is treated as 3
Integer	Intent (in)	ninst	Number of RTTOV Ids or instrument requested
Integer	Intent (in)	instrument(3,ninst)	platform id; satellite id, instrument id for each sensor (see Tables 2/3).
Integer Optional	Intent (in)	channels(:,ninst)	list of channels to extract for each instrument
Integer	Intent (out)	errorstatus (ninst)	return code
Type(rttov_coef)	Intent (out)	coef (ninst)	coefficients
Type(rttov_coeff_scatt_ir)	Intent (out)	coef_scatt_ir(ninst)	Scattering coefficients
Type(rttov_optpar_ir)	Intent (out)	opt(ninst)	Optical parameters

Annex F: RTTOV_direct interface

Call **rttov_direct**(rttov_errorstatus, nprofiles, nchannels, channels,
lprofiles, addinterp, profiles, coef(no_id), coef_scatt_ir, optp,
lsun, laerosl, lclouds, calcemis, emissivity, transmission, radiance)

rttov_direct is called for every instrument required for *nprofiles* per call.

Subroutine arguments:

Type	In/Out	Variable	Description	Example for HIRS
Integer	Intent(out)	errorstatus(nprofiles)	Return flag (0=OK)	0 or >0
Integer	Intent(in)	nprofiles	Number of profiles	2
Integer	Intent(in)	nchannels	Number of radiance streams computed (nchannels * nprofiles)	38
Integer	Intent(in)	channels(nchannels)	Channel indices	1,2,3,…18,19
Integer	Intent(in)	lprofiles(nchannels)	Profile indices	1,1,1,1 …,2,2,2
Logical	Intent(in)	addinterp	Activate profile interpolation	True/False
Type(profile_type)	Intent(in)	profiles(nprofiles)	Profiles	N/A
Type(rttov_coef)	Intent(in)	coef	Optical depth coefficients	N/A
Type(rttov_coef_scatt_ir)	Intent(in)	coef_scatt_ir	IR scattering coefficients	N/A
Type(rttov_optpar_ir)	Intent(in)	optp	IR optical parameters	N/A
Logical	Intent(in)	lsun	switch for solar computations	Logical
Logical	Intent(in)	laerosl	switch for aerosol computations	Logical
Logical	Intent(in)	lclouds	switch for cloud computations	Logical
Logical	Intent(in)	calcemiss(nchannels)	switch for emissivity calc.	True/False
Real	Intent(inout)	emissivity(nchannels)	surface emissivity	0.98,0.98..
Type(transmission_Type	Intent(out)	transmission	Transmittances (0-1.)	N/A
Type(radiance_type)	Intent(inout)	radiance	radiances (mw/cm-1/ster/sq.m) & degK	

emissivity is calculated for channels when calcemiss for that channel is true. The model depends on the sensor and on the coefficient file, for IR the model is ISEM and for MW FASTEM 2 or 3. The version of the model inside the coefficient file defines the version of the emissivity algorithm (see Table 4).

Annex G: RTTOV K interface

Subroutine **rttov_k**(errorstatus, nprofiles, nchannels, channels, lprofiles, addinterp, profiles, coef, coef_scatt_ir, optp, lsun, laerosl, lclouds, switchrad, calcemiss, emissivity, profiles_k, emissivity_k, transmission, transmission_k, radiancedata, radiance_k)

rttov_k is called for every sensor required for nprofiles at a time. The number of calculated radiances is nchannels, the array channels and lprofiles contains the corresponding channel and profile number for each computed radiance.

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus(nprofiles)	return flag
Integer	Intent(in)	nprofiles	Number of profiles
Integer	Intent(in)	nchannels	Number of radiance streams computed internally (= nchan1 * profiles)
Integer	Intent(in)	channels(nchannels)	Channel indices (see Table 13)
Integer	Intent(in)	lprofiles(nchannels)	Profiles indices (see Table 13)
Logical	Intent(in)	addinterp	switch for interpolation
Type(profile_type)	Intent(in)	profiles(nprofiles)	Profiles
Type(profile_type)	Intent(inout)	profiles_k(nchannels)	K matrix on profile variables
Type(rttov_coef)	Intent(in)	coef	Coefficients
Type(rttov_coef_scatt_ir)	Intent(in)	coef_scatt_ir	IR scattering coefficients
Type(rttov_optpar_ir)	Intent(in)	optp	IR optical parameters
Logical	Intent(in)	lsun	switch for solar computations
Logical	Intent(in)	laerosl	switch for aerosol computations
Logical	Intent(in)	lclouds	switch for cloud computations
Logical	Intent(in)	switchrad	Switch for BT/Rad (true for BT)
Logical	Intent(in)	calcemiss(nchannels)	switch for emmissivity calc.
Real	Intent(inout)	emissivity(nchannels)	surface emmissivity
Real	Intent(inout)	emissivity_k(nchannels)	K matrix on surface emissivity
Type(transmission_Type	Intent(inout)	transmission	Transmittances (0-1.)
Type(transmission_Type	Intent(inout)	transmission_k	K of transmittances
Type(radiance_type)	Intent(inout)	radiance	Forward model output radiances (mw/cm-1/ster/sq.m) & degK
Type(radiance_type)	Intent(inout) Optional	radiance_k	Optional input if a perturbation radiance is already calculated

For normal use there is no need to provide the radiance_k argument, the routine makes the calculation of the K matrix for a perturbation of 1K (or 1 $mW/m^2/ster/cm^{-1}$ if switchrad is false).
For some applications when a perturbation is already calculated by the calling program, it is possible to call rttov_k with the radiance_k argument. In that case take care total (or BT), overcast and downcld arrays should be initialised and the others set to 0.
If calcemiss is .false. the emissivity_k array must be set to zero on input.

Annex H: RTTOV TL interface

Subroutine **rttov_tl**(errorstatus, nprofiles, nchannels, channels, lprofiles, addinterp, profiles, profiles_tl, coef, coef_scatt_ir, optp, lsun, laerosl, lclouds, calcemiss, emissivity, emissivity_tl, transmission, transmission_tl, radiancedata, radiancedata_tl)

rttov_tl is called for every sensor required for nprofiles at a time. The number of calculated radiances is nchannels, the array channels and lprofiles contain the corresponding channel and profile indices for each computed radiance.

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus(nprofiles)	Return flag (0=OK)
Integer	Intent(in)	nprofiles	Number of profiles
Integer	Intent(in)	nchannels	Number of radiance streams computed (nchannels * nprofiles)
Integer	Intent(in)	channels(nchannels)	Channel indices (see Table 13)
Integer	Intent(in)	lprofiles(nchannels)	Profiles indices (see Table 13)
Logical	Intent(in)	addinterp	switch for interpolation
Type(profile_type)	Intent(in)	profiles(nprofiles)	Profiles
Type(profile_type)	Intent(in)	profiles_tl(nprofiles)	Input profile variable increments
Type(rttov_coef)	Intent(in)	coef	Coefficients
Type(rttov_coef_scatt_ir)	Intent(in)	coef_scatt_ir	IR scattering coefficients
Type(rttov_optpar_ir)	Intent(in)	optp	IR optical parameters
Logical	Intent(in)	lsun	switch for solar computations
Logical	Intent(in)	laerosl	switch for aerosol computations
Logical	Intent(in)	lclouds	switch for cloud computations
Logical	Intent(in)	calcemiss(nchannels)	switch for emissivity calc.
Real	Intent(inout)	emissivity(nchannels)	surface emissivity
Real	Intent(inout)	emissivity_tl(nchannels)	TL on surface emissivity
Type(transmission_Type	Intent(inout)	transmission	Transmittances (0-1.)
Type(transmission_Type	Intent(inout)	transmission_tl	TL of transmittances
Type(radiance_type)	Intent(inout)	radiancedata	Forward model output radiances (mw/cm-1/ster/sq.m) & degK
Type(radiance_type)	Intent(inout)	radiancedata_tl	TL output radiances (mw/cm-1/ster/sq.m) & degK

If calcemiss is .false. the emissivity_tl array must be set to zero on input.

Annex I: RTTOV AD interface

Subroutine **rttov_ad**(errorstatus, nprofiles, nchannels, channels, lprofiles, addinterp, profiles, profiles_ad, coef, coef_scat_ir, optp, lsun, laerosl, lclouds, switchrad, calcemiss, emissivity, emissivity_ad, transmission, transmission_ad, radiancedata, radiancedata_ad)

rttov_ad is called for every sensor required for nprofiles at a time. The number of calculated radiances is nchannels, the array channels and lprofiles contains the corresponding channel and profile number for each computed radiance.

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus(nprofiles)	return flag
Integer	Intent(in)	nprofiles	Number of profiles
Integer	Intent(in)	nchannels	Number of radiance streams computed internally (= nfreq * npol * profiles) npol is required polarisation/channel.
Integer	Intent(in)	channels(nchannels)	Channel indices (see Table 13)
Integer	Intent(in)	lprofiles(nchannels)	Profiles indices (see Table 13)
Logical	Intent(in)	addinterp	switch for interpolation
Type(profile_type)	Intent(in)	profiles(nprofiles)	Profiles
Type(profile_type)	Intent(inout)	profiles_ad(nchannels)	AD of profile variables
Type(rttov_coef)	Intent(in)	coef	Coefficients
Type(rttov_coef_scatt_ir)	Intent(in)	coef_scatt_ir	IR scattering coefficients
Type(rttov_optpar_ir)	Intent(in)	optp	IR optical parameters
Logical	Intent(in)	lsun	switch for solar computations
Logical	Intent(in)	laerosl	switch for aerosol computations
Logical	Intent(in)	lclouds	switch for cloud computations
Logical	Intent(in)	switchrad	Switch for BT/Rad (true for BT)
Logical	Intent(in)	calcemiss(nchannels)	switch for emissivity calc.
Real	Intent(inout)	emissivity(nchannels)	surface emissivity
Real	Intent(inout)	emissivity_ad(nchannels)	AD on surface emissivity
Type(transmission_Type	Intent(inout)	transmission	Transmittances (0-1.)
Type(transmission_Type	Intent(inout)	transmission_ad	AD of transmittances
Type(radiance_type)	Intent(inout)	radiancedata	Forward model output radiances (mw/cm-1/ster/sq.m) & degK
Type(radiance_type)	Intent(inout)	radiancedata_ad	AD output radiances (mw/cm-1/ster/sq.m) & degK

switchrad determines the input perturbation array (and so unit) of radiancedata_ad. If switchrad is false the radiance array radiancedata_ad%total is the considered the input, if switchrad is true then the brightness temperature radiancedata_ad%bt is the input perturbation.
If calcemiss is .false. the emissivity_ad array must be set to zero on input.

Annex J: Definition of derived types (structures)

Only derived types which can be used at the user's level are presented, see rttov_types.F90 for the full description of all derived types used.

J.1 Profile Structure

The profile structure is composed of the atmospheric part and two other structures for 2 meters air and skin surface. If the user is not able to provide an ozone profile a CO_2 profile or a cloud liquid water, the flags ozone_data, co2_data and clw_data (unset flag) just need to be set to false.

Type	Variable	Description
Surface skin		
Type skin_type		
Integer	surftype	0=land, 1=sea, 2=sea-ice
Integer	watertype	0=fresh water, 1=ocean water
Real	t	radiative skin temperature (K)
Real	fastem(fastem_sp)	land/sea-ice surface parameters for fastem-2/3
Surface 2m		
Type s2m_type		
Real	t	temperature (K)
Real	q	water vapour (ppmv)
Real	o	ozone (ppmv) *never used*
Real	p	surface pressure (hPa)
Real	u	U 10m wind component (m/s)
Real	v	V 10m wind component (m/s)
Real	wfetc	Wind fetch (m) (typically 100000)
Atmospheric Profile		
Type profile_type		
Logical switches		
logical	addsolar	include solar radiance for NIR channels
logical	addrefrac	include variable path length calculation
logical	addaerosl	include aerosol calculation
logical	ozone_data	ozone profiles available
logical	co2_data	carbon dioxide profiles available
logical	clw_data	cloud liquid water profiles available (MW)
logical	n2o_data	nitrous oxide profiles available
logical	co_data	carbon monoxide profiles available
logical	ch4_data	methane profiles available
logical	cld_data	cloud liquid water profiles available (IR)
logical	aer_data	aerosol profiles available
Integer	nlevels	number of atmospheric levels
integer	idg	Scheme for IWC to eff diameter,Dg 1=Ou and Liuou; 2=Wyser et al.; 3=Boudala et al; 4=McFarquhar et al.

integer	ish	Choose the shape of ice crystals, 1=Hexagonal ice crystals; 2=Ice aggregates
Atmosphere defined on nlevels		
Real	p(:)	pressure (hPa)
Real	t(:)	temperature (K)
Real	q(:)	water vapour (ppmv)
Real	o3(:)	ozone (ppmv)
Real	co2(:)	carbon dioxide (ppmv)
Real	clw(:)	cloud liquid water (kg/kg) *Microwave only*
Real	n2o(:)	nitrous oxide (ppmv)
Real	ch4(:)	methane (ppmv)
Real	co(:)	carbon monoxide (ppmv)
Real	aerosols(:,:)	aerosols (cm^{-3})
Real	cloud(:,:)	cloud water/ice ($g.m^{-3}$) *IR only*
Real	cfrac(:,:)	cloud fractional cover (0-1) *IR only*
surface		
Type(sskin_type)	skin	
Type(s2m_type)	s2m	
angles		
Real	zenangle	local satellite zenith angle (deg)
Real	azangle	local sat azimuth angle (deg) (0-360; east=90)
Real	sunzenangle	local solar zenith angle (deg) (0-90)
Real	sunazangle	local solar azimuth angle (deg) (0-360; east=90)
Real	elevation	elevation (km)
Real	latitude	Latitude (deg)
Black body cloud		
Real	ctp	cloud top pressure (hPa)
Real	cfraction	cloud fraction (0 - 1) 1 for 100% cloud cover

J.2 Radiance Structure

The radiance structure is composed of the output radiances in units of $mw/cm^{-1}/ster/m^2$ and the output brightness temperatures in degK for each channel. Single element arrays are of size nchannels and arrays of 2 dimensions are of size (nlevels, nchannels). Both radiances and brightness temperatures are computed so the user can decide which quantity he wants to use in his program.

Type	Variable	Description
Radiance		*All in units of mw/cm-1/ster/sq.m*
Real(Kind=jprb)	clear(:) (nchannels)	clear sky radiance
Real(Kind=jprb)	cloudy(:)	100% cloudy radiance for given cloud
Real(Kind=jprb)	total(:)	Clear+cloudy radiance for given scene
Real(Kind=jprb)	upclear(:)	clear sky radiance without reflection term
Real(Kind=jprb)	dnclear(:)	clear sky downwelling radiance
Real(Kind=jprb)	reflclear(:)	reflected clear sky downwelling radiance
Real(Kind=jprb)	overcast(:,:)(nlevels,nchan)	overcast radiance at given cloud top
Real(Kind=jprb)	downcld(:,:)	contribution to radiance of downward cloud emission at given cloud top (levels,chan)
Real(Kind=jprb)	up(:,:)	sum(B * dT) above cloud upwelling radiance for

		each pressure level
Real(Kind=jprb)	down(:,:)	sum (B / T**2 dT) above cloud downwelling radiance for each pressure level
Real(Kind=jprb)	surf(:,:)	radiance at surface emitted from a black cloud
Brightness Temperature		*All in units of degK*
Real(Kind=jprb)	bt(:)	Brightness temp equivalent to total radiance
Real(Kind=jprb)	bt_clear(:)	Brightness temp equivalent to clear radiance

J.3 Transmittance Structure

Transmission and optical depths are both unitless. Single element arrays are of size nchannels and arrays of 2 or 3 dimensions are defined in the table below. For clear sky calculations the nstreams element is set to 0. This is just a subset of the variables available. See rttov_types.F90 for the complete list.

Type	Variable	Description
Transmission		*Unitless*
Real(Kind=jprb)	tau_surf(:,:)	transmittance from surface (array size is of size nstreams, nchannels)
Real(Kind=jprb)	tau_layer(:,:,:)	transmittance from each standard pressure level array (nlevels,nstreams,channels)
Real(Kind=jprb)	tau_total(:)	transmittance from surface (array size is of size nchannels)
Real(Kind=jprb)	tau_layers(:,:)	transmittance from each standard pressure level array (nlevels,nchannels)
Optical Depth		*Unitless*
Real(Kind=jprb)	od_layer(:,:,:)	op dep from each standard pressure level array (nlevels,nstreams,nchannels)
Real(Kind=jprb)	od_singlelayer(:,:,:)	single-layer optical depth (nlevels,nstreams,nchannels)

J.4 Profile Structure for RTTOV_SCATT cloud/precipitation

The *profile_cloud_type* defined in rttov_types.F90 is for the RTTOV_SCATT microwave scattering calculations.

Type	Variable	Description
Integer	nlevels	number of atmospheric levels, which should match that in the other input profiles
Logical	use_totalice	logical flag to switch between using separate ice and snow, or total ice hydrometeor types.
Real(Kind=jprb)	ph(:)	nlevels+1 of half-level pressures (hPa)
Real(Kind=jprb)	cc(:)	nlevels of cloud cover (0-1)
Real(Kind=jprb)	clw(:)	nlevels of cloud liquid water (kg/kg)
Real(Kind=jprb)	ciw(:)	nlevels of cloud ice water (kg/kg)
Real(Kind=jprb)	totalice(:)	nlevels of total ice (kg/kg)
Real(Kind=jprb)	rain(:)	nlevels of rain flux (kg/(m^2)/s)
Real(Kind=jprb)	sp(:)	nlevels of solid precipitation flux (kg/(m^2)/s)

Annex K: Contents of rttov_const.F90

```
!
Module rttov_const
  ! Description:
  ! Definition of all parameters (constants) for RTTOV
  !
  ! Copyright:
  !    This software was developed within the context of
  !    the EUMETSAT Satellite Application Facility on
  !    Numerical Weather Prediction (NWP SAF), under the
  !    Cooperation Agreement dated 25 November 1998, between
  !    EUMETSAT and the Met Office, UK, by one or more partners
  !    within the NWP SAF. The partners in the NWP SAF are
  !    the Met Office, ECMWF, KNMI and MeteoFrance.
  !
  !    Copyright 2002, EUMETSAT, All Rights Reserved.
  !
  ! History:
  ! Version   Date      Comment
  ! -------   ----      -------
  !  1.0   01/12/2002  New F90 code with structures (P Brunel A Smith)
  !  1.1   29/01/2003  New platforms and instruments (P Brunel)
  !                    Hard limits for input profiles
  !  1.2   19/02/2003  Some changes to limits and comments (R Saunders)
  !  1.3   06/05/2003  Change version number to 7.3.1
  !                    and add references for physical constants (P Brunel)
  !  1.4      08/2003  Added variables for MW scattering (F Chevallier)
  !  1.5   18/09/2003  Added coefficients for cloud absorption properties (P
Francis)
  !  1.6   15/10/2003  Added new sections in parameter files for scatt   (F
Chevallier)
  !  1.7   23/11/2003  Added new definitions of polarisations 2.1 (S English)
  !  1.8   25/08/2005  Made inst_name a parameter (R Saunders)
  !  1.9   11/01/2006  Added logical flag for surface humidity use (R Saunders)
  !  1.10  12/01/2006  Marco Matricardi (ECMWF):
  !           --       Added variables for CO2,CO,N2O and CH4 molecules.
  !           --       Added parameters for the computation of the refractive
index
  !           --       of air.
  !  1.11  06/02/2006  Added logical flag for linear in tau approx (R Saunders)
  !  1.12  06/04/2006  Added Meghatropiques (R. Saunders)
  !  1.13  14/03/2007  Added units conversion constants
  !  1.14  16/05/2007  Added polarimetric sensor type (R Saunders)
  !  1.15  25/09/2007  Added maximum number of warnings for checkinput (P
Brunel)
  !  1.16  11/10/2007  Remove zhusta* and zice* constants ( P.Marguinaud )
  !  1.17  07/12/2007  Remove maximum number of warnings for checkinput (P
Brunel)
  !  1.18  12/12/2007  Added hard limits for trace gases (R Saunders)
  !  1.19  13/12/2007  Renamed linear_tau (R Saunders)
  !  1.20  01/11/2007  Added parameters for section length and AD/K code (A.
Geer)
  !  1.21  16/01/2008  Facility to apply regression limits (N. Bormann)
  !
  !1.1 general
  !-----------
  ! Version number of the current code
```

```
  Use parkind1, Only : jpim     ,jprb
  Implicit None
  Integer(Kind=jpim), Parameter :: version = 9
  Integer(Kind=jpim), Parameter :: release = 0
  Integer(Kind=jpim), Parameter :: minor_version = 0

  Integer(Kind=jpim), Parameter :: version_compatible_min = 7 ! minimum version
number
!  Integer(Kind=jpim), Parameter :: version_compatible_max = 8 ! maximum version
number
  Integer(Kind=jpim), Parameter :: version_compatible_max = 9 ! maximum version
number
          ! compatible for coefficients.
          ! coef files with "id_comp_lvl" outside range will be rejected

  Character (len=16), Parameter :: rttov_magic_string = '%RTTOV_COEFF    '
  Real(Kind=jprb),                Parameter :: rttov_magic_number =
1.2345E+12_JPRB

  Integer(Kind=jpim), Parameter :: default_err_unit = 0  ! standard error unit
number
                              ! standard error unit number is 7 for HPUX
  Logical , Parameter :: use_q2m = .false.    ! set to true to activate use of
surface humidity
  Logical , Parameter :: rt8_mode = .false. ! set true to make RTTOV_9 compute
RTTOV-8 (non-linear tau etc) way
  Logical , Parameter :: apply_reg_limits = .false. ! set to true makes
rttov_checkinput reset the profiles
                                                     ! variables to the regression
limits if outside
  Logical , Parameter :: lgradp = .false.  ! Allow TL/AD of user pressure levels
                                           ! if the internal interpolation is
used

  !1.2 physical constants
  !----------------------
  ! Molecular weights  (g/mole) are calculated by adding NIST Standard Atomic
Weights
  ! Molecular weight of dry air refers to US standard atmosphere 1976
  ! NIST  Standard Atomic Weight are:
  ! H    1.00794   (7)
  ! C   12.0107    (8)
  ! N   14.0067    (2)
  ! O   15.9994    (3)
  Real(Kind=jprb), Parameter :: mair = 28.9644_JPRB
  Real(Kind=jprb), Parameter :: mh2o = 18.01528_JPRB
  Real(Kind=jprb), Parameter :: mo3  = 47.9982_JPRB
  Real(Kind=jprb), Parameter :: mco2 = 44.0095_JPRB
  Real(Kind=jprb), Parameter :: mch4 = 16.04246_JPRB
  Real(Kind=jprb), Parameter :: mn2o = 44.0128_JPRB
  Real(Kind=jprb), Parameter :: mco  = 28.0101_JPRB

  ! Gravity from NIST 9.80665 ms-1 (exact)
  Real(Kind=jprb), Parameter :: gravity = 9.80665_JPRB

  !
  ! Kaye & Laby latest library edition is 16e 1995, and gives
  ! * standard value  g = 9.80665 ms-1 exactly (p.191)
  ! * earth mean radius r= 6371.00 km (p191)
```

```
!    [defined as [(r_equator)^2 (r_pole)]^1/3]
Real(Kind=jprb), Parameter :: pi      = 3.1415926535_JPRB
Real(Kind=jprb), Parameter :: deg2rad = pi/180.0_JPRB
Real(Kind=jprb), Parameter :: earthradius = 6371.00_JPRB
Real(Kind=jprb), Parameter :: flatt       = 3.3528107E-3_JPRB
Real(Kind=jprb), Parameter :: omega       = 7292115E-11_JPRB
Real(Kind=jprb), Parameter :: eqrad       = 6378.137_JPRB
Real(Kind=jprb), Parameter :: grave       = 9.7803267715_JPRB

! The Cosmic Microwave Background Spectrum from the Full COBE FIRAS Data Set
! Fixsen D.J. et all
! Astrophysical Journal v.473, p.576 December 1996
! CMBR = 2.728 +- 0.004K
Real(Kind=jprb), Parameter :: tcosmic     = 2.728_JPRB
!  Real(Kind=jprb), Parameter :: tcosmic     = 0.1_JPRB !used for ECMWF tests

! Universal gas constant R = 8.314510 J/mol/K
Real(Kind=jprb), Parameter :: rgp = 8.314510_JPRB
Real(Kind=jprb), Parameter :: rgc = 8.314472_JPRB

! mean molar mass of dry air rm = 0.0289644 kg.mol^-1
Real(Kind=jprb), Parameter :: rm = 0.0289644_JPRB

! units conversion from  mixing ratio to ppmv
Real(Kind=jprb), Parameter :: q_mixratio_to_ppmv  = 1.60771704e+6_JPRB
Real(Kind=jprb), Parameter :: o3_mixratio_to_ppmv = 6.03504e+5_JPRB
Real(Kind=jprb), Parameter :: co2_mixratio_to_ppmv= 6.58114e+5_JPRB
Real(Kind=jprb), Parameter :: co_mixratio_to_ppmv = 9.67053e+5_JPRB
Real(Kind=jprb), Parameter :: n2o_mixratio_to_ppmv= 6.58090e+5_JPRB
Real(Kind=jprb), Parameter :: ch4_mixratio_to_ppmv= 1.80548e+6_JPRB

! zero temperature(K)
Real(Kind=jprb), Parameter :: t0 =273.15

!1.3 satellite and instrument information
!----------------------------------------

!platform id codes
Integer(Kind=jpim), Parameter :: nplatforms = 20
Integer(Kind=jpim), Parameter :: &
     & platform_id_noaa     = 1, &
     & platform_id_dmsp     = 2, &
     & platform_id_meteosat = 3, &
     & platform_id_goes     = 4, &
     & platform_id_gms      = 5, &
     & platform_id_fy2      = 6, &
     & platform_id_trmm     = 7, &
     & platform_id_ers      = 8, &
     & platform_id_eos      = 9, &
     & platform_id_metop    = 10, &
     & platform_id_envisat  = 11, &
     & platform_id_msg      = 12, &
     & platform_id_fy1      = 13, &
     & platform_id_adeos    = 14, &
     & platform_id_mtsat    = 15, &
     & platform_id_coriolis = 16, &
     & platform_id_npoess   = 17, &
     & platform_id_gifts    = 18, &
     & platform_id_xxxxx    = 19, &
```

```
      & platform_id_meghatr  = 20

 !platform names
 Character (len=8), Parameter :: platform_name(nplatforms) = &
      & (/ 'noaa    ', 'dmsp    ', 'meteosat', 'goes    ', 'gms     ', &
         & 'fy2     ', 'trmm    ', 'ers     ', 'eos     ', 'metop   ', &
         & 'envisat ', 'msg     ', 'fy1     ', 'adeos   ', 'mtsat   ', &
         & 'coriolis', 'npoess  ', 'gifts   ', 'xxxxxxxx', 'meghatr '/)

 !instrument id codes
 Integer(Kind=jpim), Parameter :: &
      & inst_id_hirs    =  0, &
      & inst_id_msu     =  1, &
      & inst_id_ssu     =  2, &
      & inst_id_amsua   =  3, &
      & inst_id_amsub   =  4, &
      & inst_id_avhrr   =  5, &
      & inst_id_ssmi    =  6, &
      & inst_id_vtpr1   =  7, &
      & inst_id_vtpr2   =  8, &
      & inst_id_tmi     =  9, &
      & inst_id_ssmis   = 10, &
      & inst_id_airs    = 11, &
      & inst_id_hsb     = 12, &
      & inst_id_modis   = 13, &
      & inst_id_atsr    = 14, &
      & inst_id_mhs     = 15, &
      & inst_id_iasi    = 16, &
      & inst_id_amsr    = 17, &
      & inst_id_mtsatim= 18, &
      & inst_id_atms    = 19, &
      & inst_id_mviri   = 20, &
      & inst_id_seviri  = 21, &
      & inst_id_goesim  = 22, &
      & inst_id_goessd  = 23, &
      & inst_id_gmsim   = 24, &
      & inst_id_vissr   = 25, &
      & inst_id_mvisr   = 26, &
      & inst_id_cris    = 27, &
      & inst_id_cmis    = 28, &
      & inst_id_viirs   = 29, &
      & inst_id_windsat= 30, &
      & inst_id_gifts   = 31, &
      & inst_id_xxxx1   = 32, &
      & inst_id_xxxx2   = 33, &
      & inst_id_saphir  = 34, &
      & inst_id_madras  = 35


 Integer(Kind=jpim), Parameter :: ninst = 36
 ! List of instruments  !!!! HIRS is number 0
 Character (len=8), Dimension(0:ninst-1),parameter :: inst_name =       &
       & (/ 'hirs    ', 'msu     ', 'ssu     ', 'amsua   ', 'amsub   ',  &
          & 'avhrr   ', 'ssmi    ', 'vtpr1   ', 'vtpr2   ', 'tmi     ',  &
          & 'ssmis   ', 'airs    ', 'hsb     ', 'modis   ', 'atsr    ',  &
          & 'mhs     ', 'iasi    ', 'amsr    ', 'imager  ', 'atms    ',  &
          & 'mviri   ', 'seviri  ', 'imager  ', 'sounder ', 'imager  ',  &
          & 'vissr   ', 'mvisr   ', 'cris    ', 'cmis    ', 'viirs   ',  &
          & 'windsat ', 'gifts   ', 'xxxxxxxx', 'xxxxxxxx', 'saphir  ',  &
```

```
        & 'madras  '   /)


!1.4 Coefficient file Section names
!----------------------------------
Integer(Kind=jpim), Parameter :: nsections = 31
Integer(Kind=jpim), Parameter :: lensection = 22
Character(len=lensection), Parameter :: section_types(nsections) = &
  & (/ 'IDENTIFICATION        ', 'LINE-BY-LINE          ', &
     & 'FAST_MODEL_VARIABLES  ', 'FILTER_FUNCTIONS      ', &
     & 'FUNDAMENTAL_CONSTANTS ', 'SSIREM                ', &
     & 'FASTEM                ', 'REFERENCE_PROFILE     ', &
     & 'PROFILE_LIMITS        ', 'FAST_COEFFICIENTS     ', &
     & 'COEF_SUB_FILES        ', 'GAZ_UNITS             ', &
     & 'DIMENSIONS            ', 'FREQUENCIES           ', &
     & 'HYDROMETEOR           ', 'CONVERSIONS           ', &
     & 'EXTINCTION            ', 'ALBEDO                ', &
     & 'ASYMMETRY             ', 'GAS_SPECTRAL_INTERVAL ', &
     & 'TRANSMITTANCE_TRESHOLD', 'SOLAR_SPECTRUM        ', &
     & 'WATER_OPTICAL_CONSTANT', 'WAVE_SPECTRUM         ', &
     & 'AEROSOLS_PARAMETERS   ', 'AEROSOLS_COMPONENTS   ', &
     & 'WATERCLOUD_TYPES      ', 'WATERCLOUD_PARAMETERS ', &
     & 'ICECLOUD_TYPES        ', 'HEXAGONAL_PARAMETERS  ', &
     & 'AGGREGATE_PARAMETERS  '/)

!sensors id codes
Integer(Kind=jpim), Parameter :: nsensors = 4
Integer(Kind=jpim), Parameter :: &
     & sensor_id_ir     = 1, &
     & sensor_id_mw     = 2, &
     & sensor_id_hi     = 3, &
     & sensor_id_po     = 4

!sensors names
Character (len=2), Parameter :: sensor_name(nsensors) = &
     & (/ 'ir', 'mw', 'hi', 'po' /)

!gas id codes
Integer(Kind=jpim), Parameter :: ngases_max = 8
Integer(Kind=jpim), Parameter :: &
      & gas_id_mixed       = 1, &
      & gas_id_watervapour = 2, &
      & gas_id_ozone       = 3, &
      & gas_id_wvcont      = 4, &
      & gas_id_co2         = 5, &
      & gas_id_n2o         = 6, &
      & gas_id_co          = 7, &
      & gas_id_ch4         = 8

!gas names
Character (len=12), Parameter :: gas_name(ngases_max) = &
      & (/ 'Mixed_gases ', &
         & 'Water_vapour', &
         & 'Ozone       ', &
         & 'WV_Continuum', &
         & 'CO2         ', &
         & 'N2O         ', &
         & 'CO          ', &
         & 'CH4         ' /)
```

```
  !gas units
  Integer(Kind=jpim), Parameter :: ngases_unit = 2
  Integer(Kind=jpim), Parameter :: &
        & gas_unit_specconc  = 1, &
        & gas_unit_ppmv      = 2
  Character (len=12), Parameter :: gas_unit_name(ngases_unit) = &
        & (/ 'spec. concen', &
           & 'ppmv        '  /)


  !1.5 error reporting
  !--------------------
  !error status values
  Integer(Kind=jpim), Parameter :: nerrorstatus = 3
  Integer(Kind=jpim), Parameter :: errorstatus_success = 0
  Integer(Kind=jpim), Parameter :: errorstatus_warning = 1
  Integer(Kind=jpim), Parameter :: errorstatus_fatal   = 2
  Integer(Kind=jpim), Parameter :: errorstatus_info    = 3
  Character(len=*), Parameter :: errorstatus_text(0:nerrorstatus) = &
       & (/ 'success', &
       & 'warning', &
       & 'fatal  ', &
       & 'info   '  /)


  !1.6 surface types
  !-----------------
  Integer(Kind=jpim), Parameter :: nsurftype = 2
  Integer(Kind=jpim), Parameter :: surftype_land = 0
  Integer(Kind=jpim), Parameter :: surftype_sea = 1
  Integer(Kind=jpim), Parameter :: surftype_seaice = 2

  !1.7 water types
  !---------------
  Integer(Kind=jpim), Parameter :: nwatertype = 1
  Integer(Kind=jpim), Parameter :: watertype_fresh_water = 0
  Integer(Kind=jpim), Parameter :: watertype_ocean_water = 1

  !1.8 cloud emissivity
  !---------------------
  Integer(Kind=jpim), Parameter :: overlap_scheme = 2    ! overlap scheme
  ! 1 => Geleyn and Hollingsworth (1979)
  ! 2 => Raisanen (1998)

  !
  !1.9 Hard limits for control of input profile
  !--------------------------------------------
  ! Temperature
  Real(Kind=jprb), Parameter :: tmax   = 400.0_JPRB       ! degK
  Real(Kind=jprb), Parameter :: tmin   = 90.0_JPRB        ! degK
  ! Water Vapour
  Real(Kind=jprb), Parameter :: qmax   = 0.60E+06_JPRB    ! ppmv 0.373_JPRB
kg/kg
  Real(Kind=jprb), Parameter :: qmin   = 0.00_JPRB        ! ppmv
  ! Ozone
  Real(Kind=jprb), Parameter :: o3max  = 1000.0_JPRB      ! ppmv  1.657E-3_JPRB
kg/kg
  Real(Kind=jprb), Parameter :: o3min  = 0.0_JPRB         ! ppmv
```

```
  ! CO2
  Real(Kind=jprb), Parameter :: co2max = 1000.0_JPRB       ! ppmv
  Real(Kind=jprb), Parameter :: co2min = 0.0_JPRB          ! ppmv
  ! CO
  Real(Kind=jprb), Parameter :: comax  = 10.0_JPRB         ! ppmv
  Real(Kind=jprb), Parameter :: comin  = 0.0_JPRB          ! ppmv
  ! N2O
  Real(Kind=jprb), Parameter :: n2omax = 10.0_JPRB         ! ppmv
  Real(Kind=jprb), Parameter :: n2omin = 0.0_JPRB          ! ppmv
  ! CH4
  Real(Kind=jprb), Parameter :: ch4max = 50.0_JPRB         ! ppmv
  Real(Kind=jprb), Parameter :: ch4min = 0.0_JPRB          ! ppmv
  ! Cloud Liquid Water
  Real(Kind=jprb), Parameter :: clwmax = 1.0_JPRB          ! kg/kg
  Real(Kind=jprb), Parameter :: clwmin = 0.0_JPRB          ! kg/kg
  ! Surface Pressure
  Real(Kind=jprb), Parameter :: pmax   = 1100.0_JPRB       ! surface pressure hPa
  Real(Kind=jprb), Parameter :: pmin   = 400.0_JPRB        ! hPa
  ! Surface Wind
  Real(Kind=jprb), Parameter :: wmax   =  100.0_JPRB       ! surface wind speed
(m/s)
  ! Zenith Angle
  Real(Kind=jprb), Parameter :: zenmax = 75.0_JPRB         ! zenith angle (Deg) =
secant 3.86_JPRB
  ! Cloud Top Pressure
  Real(Kind=jprb), Parameter :: ctpmax = 1100.0_JPRB       ! (hPa)
  Real(Kind=jprb), Parameter :: ctpmin =   50.0_JPRB       ! (hPa)


  !1.10  Maximum Optical Depth
  !--------------------------
  ! maximum value of optical depth for transmittance calculation
  ! e(-30) -> 10**-14
  ! e(-50) -> 10**-22
  Real(Kind=jprb), Parameter  :: max_optical_depth = 50._JPRB

  !2 RTTOV7 aux parameters
  !-------------------------
  Integer(Kind=jpim), Parameter :: fastem_sp = 5  ! max. number of fastem
surface parameters
  Integer(Kind=jpim), Parameter :: mwcldtp = 322.0_JPRB  ! Upper pressure level
(HPa) for lwp calcs
  Real(Kind=jprb), Parameter    :: pressure_top = 0.004985_JPRB ! Pressure of
top level for
                                                ! Line/Line calculations (hPa)
  Real(Kind=jprb) , Dimension(8), Parameter :: dcoeff =       &! Debye coefs
        & (/ 17.1252_JPRB, 134.2450_JPRB, 310.2125_JPRB,  5.667_JPRB,    &
          & 188.7979_JPRB,  80.5419_JPRB,   0.1157_JPRB,  4.8417_JPRB/)

  !2.1 Polarisation definitions
  !----------------------------
  ! 1 = 0.5 (V+H)
  ! 2 = QV
  ! 3 = QH
  ! 4 = V
  ! 5 = H
  ! 6 = V , H
  ! 7 = Stokes (i.e. V , H , U, RHC)
  Integer(Kind=jpim), Dimension(7), Parameter :: npolar_compute = &
```

```
 & (/ 2, 2, 2, 1, 1, 2, 4/)
Integer(Kind=jpim), Dimension(7), Parameter :: npolar_return = &
 & (/ 1, 1, 1, 1, 1, 2, 4/)
Real(Kind=jprb), Parameter :: pol_v(3,5) = Reshape( &
  & (/ 0.5_JPRB, 0.0_JPRB, 0.0_JPRB, &
     & 0.0_JPRB, 0.0_JPRB, 1.0_JPRB, &
     & 0.0_JPRB, 1.0_JPRB, 0.0_JPRB, &
     & 1.0_JPRB, 0.0_JPRB, 0.0_JPRB, &
     & 0.0_JPRB, 0.0_JPRB, 0.0_JPRB  /), (/3,5/) )
Real(Kind=jprb), Parameter :: pol_h(3,5) = Reshape( &
  & (/ 0.5_JPRB, 0.0_JPRB, 0.0_JPRB, &
     & 0.0_JPRB, 1.0_JPRB, 0.0_JPRB, &
     & 0.0_JPRB, 0.0_JPRB, 1.0_JPRB, &
     & 0.0_JPRB, 0.0_JPRB, 0.0_JPRB, &
     & 1.0_JPRB, 0.0_JPRB, 0.0_JPRB  /), (/3,5/) )

!3 RTTOVSCATT aux parameters
!---------------------------
! Minimum cloud cover processed by rttov_scatt
Real(Kind=jprb),    Parameter  :: ccthres = 0.05_JPRB
! Rain density (g.cm-1)
Real(Kind=jprb), Parameter :: rho_rain = 1.0_JPRB
! Snow density (g.cm-1)
Real(Kind=jprb), Parameter :: rho_snow = 0.1_JPRB

! Flags to identify function in shared K/Adjoint routines
Integer(Kind=jpim), Parameter :: adk_adjoint = 0
Integer(Kind=jpim), Parameter :: adk_k       = 1

!4 Parameters to compute refractive index of air
!--------------------------------------------------------
Real(Kind=jprb), Parameter :: D1   =8341.87_JPRB
Real(Kind=jprb), Parameter :: D2   =2405955.0_JPRB
Real(Kind=jprb), Parameter :: D3   =130.0_JPRB
Real(Kind=jprb), Parameter :: D4   =15996.0_JPRB
Real(Kind=jprb), Parameter :: D5   =38.9_JPRB
Real(Kind=jprb), Parameter :: DCO2 =0.540_JPRB
Real(Kind=jprb), Parameter :: ED1  =96095.43_JPRB
Real(Kind=jprb), Parameter :: ED2  =0.601_JPRB
Real(Kind=jprb), Parameter :: ED3  =0.00972_JPRB
Real(Kind=jprb), Parameter :: ED4  =0.003661_JPRB
Real(Kind=jprb), Parameter :: EW1  =3.7345_JPRB
Real(Kind=jprb), Parameter :: EW2  =0.0401_JPRB
Real(Kind=jprb), Parameter :: HTOP =100.0_JPRB
Real(Kind=jprb), Parameter :: CTOM =1.0E-4_JPRB
Real(Kind=jprb), Parameter :: WAVER=1700.0_JPRB

!5 RTTOV8_M_SCATT
!--------------------------------------------------------
Integer(Kind=jpim), Parameter :: naer_max = 11
Integer(Kind=jpim), Parameter :: naer_cl  = 10
Integer(Kind=jpim), Parameter :: nhumaer(naer_max)=                    &
     & (/1,8,1,8,8,1,1,1,1,8,1/)

Integer(Kind=jpim), Parameter :: &
      & aer_id_inso       = 1, &
      & aer_id_waso       = 2, &
      & aer_id_soot       = 3, &
      & aer_id_ssam       = 4, &
```

```
       & aer_id_sscm        = 5, &
       & aer_id_minm        = 6, &
       & aer_id_miam        = 7, &
       & aer_id_micm        = 8, &
       & aer_id_mitr        = 9, &
       & aer_id_suso        =10, &
       & aer_id_vola        =11

  Character (len=4), Parameter :: aer_name(naer_max) = &
       & (/ 'inso', &
          & 'waso', &
          & 'soot', &
          & 'ssam', &
          & 'sscm', &
          & 'minm', &
          & 'miam', &
          & 'micm', &
          & 'mitr', &
          & 'suso', &
          & 'vola' /)

  Integer(Kind=jpim), Parameter :: nwcl_max = 5
  Integer(Kind=jpim), Parameter :: nhumwcl(nwcl_max)=                     &
      & (/1,1,1,1,1/)

  Integer(Kind=jpim), Parameter :: &
       & wcl_id_stco        = 1, &
       & wcl_id_stma        = 2, &
       & wcl_id_cucc        = 3, &
       & wcl_id_cucp        = 4, &
       & wcl_id_cuma        = 5

  Character (len=4), Parameter :: wcl_name(nwcl_max) = &
       & (/ 'stco', &
          & 'stma', &
          & 'cucc', &
          & 'cucp', &
          & 'cuma' /)

  Real(Kind=jprb), Parameter :: cldstr_threshold=-999.0_JPRB

  Integer(Kind=jpim), Parameter:: ncldtyp=6

  Integer(Kind=jpim), Parameter:: jpazn=11

  Real(Kind=jprb), Parameter :: E00        = 611.21_JPRB
  Real(Kind=jprb), Parameter :: T00        = 273.16_JPRB
  Real(Kind=jprb), Parameter :: TI         = T00 - 23.0_JPRB

  Real(Kind=jprb), Parameter :: min_tau = 1.0e-8_JPRB
  Real(Kind=jprb), Parameter :: min_od  = 1.0e-5_JPRB

End Module rttov_const
```

Annex L: Example of user interface program to run RTTOV9

```
Program example_fwd
  !
  !    This software was developed within the context of
  !    the EUMETSAT Satellite Application Facility on
  !    Numerical Weather Prediction (NWP SAF), under the
  !    Cooperation Agreement dated 25 November 1998, between
  !    EUMETSAT and the Met Office, UK, by one or more partners
  !    within the NWP SAF. The partners in the NWP SAF are
  !    the Met Office, ECMWF, KNMI and MeteoFrance.
  !
  !    Copyright 2007, EUMETSAT, All Rights Reserved.
  !
  !     *************************************************************
  !
  !     TEST PROGRAM FOR RTTOV SUITE FORWARD MODEL ONLY
  !          RTTOV VERSION 9
  ! To run this program you must have the following files
  ! either resident in the same directory or set up as a
  ! symbolic link:
  !   prof.dat       --    input profile
  !   rtcoef_platform_id_sensor.dat --  coefficient file to match
  !   the sensor you request in the input dialogue
  ! There are scripts available to set up the files above and
  ! run this program (e.g. test_rttov9.ksh)
  ! The output is generated in a file called print.dat.
  !
  !
  ! If the user wants to use this example to create his own
  ! program he will have to modify the code between
  ! comment lines of that kind:
  !     !================================
  !     !======Read =====start===========
  !          code to be modified
  !     !======Read ===== end ===========
  !     !================================
  !
  ! Current Code Owner: SAF NWP
  !
  ! History:
  ! Version   Date        Comment
  ! -------   ----        -------
  !  1.0    27/04/2004   orginal (based on tstrad) P. Brunel
  !  1.1    09/08/2004   modified to allow for variable no. channels/per profile
  !                       R. Saunders
  !  1.2    13/04/2007   Modified for RTTOV-90
  !  1.3    31/07/2007   Modified for RTTOV-91 R Saunders
  !  1.4    11/10/2007   Parallel version P.Marguinaud
  !
  ! Code Description:
  !   Language:          Fortran 90.
  !   Software Standards: "European Standards for Writing and
  !     Documenting Exchangeable Fortran 90 Code".
  !
  Use rttov_const, Only :  &
       & errorstatus_success,&
       & errorstatus_warning,&
```

```
       & errorstatus_fatal  ,&
       & q_mixratio_to_ppmv

  Use rttov_types, Only : &
       & rttov_coef     ,&
       & profile_Type   ,&
       & transmission_Type ,&
       & radiance_Type  ,&
       & rttov_coef_scatt_ir ,&
       & rttov_optpar_ir

  Use parkind1, Only : jpim     ,jprb
  !
  Implicit None
  !
#ifdef _RTTOV_EXAMPLE_FWD_PARALLEL
#include "rttov_parallel_direct.interface"
#define rttov_direct rttov_parallel_direct
#else
#include "rttov_direct.interface"
#endif
#include "rttov_setup.interface"
#include "rttov_errorhandling.interface"
#include "rttov_dealloc_coef.interface"
#include "rttov_alloc_rad.interface"
#include "rttov_errorreport.interface"
#include "rttov_alloc_prof.interface"

  !--------------------------
  !
  Integer(Kind=jpim) :: iup=20      ! unit for profile file
  Integer(Kind=jpim) :: ioout=21    ! unit for output

  ! One profile per call and one sensor only for this simple example
  Integer (Kind=jpim) :: nprof = 1           ! Number of profiles per call
  Integer(Kind=jpim)  :: nrttovid = 1        ! Number of sensor coeff files to
read

  ! RTTOV_errorhandling interface
  !====================
  Integer :: Err_Unit        ! Logical error unit (<0 for default)
  Integer :: verbosity_level ! (<0 for default)

  ! RTTOV_setup interface
  !====================
  Integer(Kind=jpim), Allocatable :: setup_errorstatus(:)  ! setup return code
  Integer(Kind=jpim), Allocatable :: instrument(:,:) !to contain platform id,
sat id and sensor id
  Type( rttov_coef ), allocatable      :: coef(:)          ! optical depth
coefficients
  Type(rttov_coef_scatt_ir),allocatable:: coef_scatt_ir(:)! scattering coeffs
  Type(rttov_optpar_ir),allocatable    :: optp(:)          ! optical properties

  ! RTTOV interface
  !====================
  Integer(Kind=jpim), Allocatable :: rttov_errorstatus(:)  ! rttov error return
code
  Integer(Kind=jpim) :: nchannels
  Integer(Kind=jpim), Allocatable :: channels   (:)
```

```
  Integer(Kind=jpim), Allocatable :: lprofiles  (:)
  Type(profile_Type), Allocatable :: profiles(:)
  Logical, Allocatable          :: calcemis(:)
  Real(Kind=jprb), Allocatable :: emissivity (:)
  Type(transmission_Type)  :: transmission ! transmittances and layer optical
depths
  Type(radiance_Type)      :: radiance

  Integer(Kind=jpim) :: alloc_status(20)
  Integer(Kind=jpim) :: errorstatus
  Real(Kind=jprb),    Allocatable :: input_emissivity (:)
  Character (len=80) :: errMessage
  Character (len=6)  :: NameOfRoutine = 'exampl'

  ! variables for input
  !====================
  Integer(Kind=jpim), Parameter :: mxchn = 9000 ! max number of channels
  Integer(Kind=jpim) :: input_chan(mxchn)
  Real(Kind=jprb)    :: input_ems(mxchn)
  Real(Kind=jprb)    :: zenith
  Real(Kind=jprb)    :: azimut
  Real(Kind=jprb)    :: lat
  Real(Kind=jprb)    :: zerht
  Real(Kind=jprb)    :: sunzang
  Real(Kind=jprb)    :: sunazang
  Integer(Kind=jpim) :: nlevels
  Integer(Kind=jpim) :: ivch, ich
  Integer(Kind=jpim) :: asw          ! allocate or deallocate switch
  Real(Kind=jprb)    :: ems_val
  Integer(Kind=jpim), Allocatable :: nchan(:) ! number of channels per profile
  Integer(Kind=jpim) :: isurf
  Integer(Kind=jpim) :: nwater
  logical :: refrac
  logical :: solrad
  logical :: laerosl
  logical :: lclouds
  logical :: all_channels
  Logical :: addinterp  ! switch for the interpolator
  ! printing arrays
  Real(Kind=jprb), Allocatable :: pr_radcld(:)
  Real(Kind=jprb), Allocatable :: pr_trans(:)
  Real(Kind=jprb), Allocatable :: pr_emis(:)
  Real(Kind=jprb), Allocatable :: pr_trans_lev(:,:)
  Character (len=3)  :: cref
  Character (len=3)  :: caer
  Character (len=3)  :: ccld
  Character (len=3)  :: csun
  ! loop variables
  Integer :: j, jch
  Integer :: np, nch
  Integer :: ilev, nprint
  Integer :: ios

  !- End of header --------------------------------------------------------

  errorstatus     = 0_jpim
  alloc_status(:) = 0_jpim
  allocate (instrument(3,nrttovid),stat= alloc_status(1))
```

```
  !=====================================================
  !========== Interactive inputs == start ==============
  Write(0,*) 'enter platform number'
  Read(*,*) instrument(1,nrttovid)
  Write(0,*) 'enter satellite number '
  Read(*,*) instrument(2,nrttovid)
  Write(0,*) 'enter instrument number'
  Read(*,*) instrument(3,nrttovid)
  Write(0,*) 'enter surface type (0=land, 1=sea, 2=ice/snow)'
  Read(*,*) isurf
  Write(0,*) ' Water type (0=fresh water, 1=ocean water)'
  Read(*,*) nwater
  Write(0,*) 'enter number of profile levels'
  Read(*,*) nlevels
  Write(0,*) 'enter zenith angle in degrees'
  Read(*,*) zenith
  !
  ! Prescribe other inputs
  azimut = 0._jprb   ! Satellite azimuth angle
  sunzang = 0._jprb  ! solar zenith angle
  sunazang = 0._jprb ! solar azimuth angle
  lat = 0._jprb      ! profile latitude
  zerht = 0._jprb    ! elevation of surface
  !
  ! Set flags
  refrac=.True.      ! include refraction in path calc
  solrad=.False.     ! Do not include reflected solar
  laerosl=.False.    ! Don't include aerosol effects
  lclouds=.False.    ! Don;t include cloud effects
  all_channels=.True.! Read all channels into memory from coef file
  addinterp = .True. ! Allow interpolation of input profile
  cref = 'YES'
  caer = ' NO'
  ccld = ' NO'
  csun = ' NO'
  !
  Allocate (nchan(nprof))
  nchan(:) = 0_jpim
  Read(*,*,iostat=ios) ich, ivch, ems_val ! channel number, validity, emissivity
  Do While (ios == 0 )
     If( ivch /= 0 ) Then
        nchan(nprof) = nchan(nprof) +1
        input_chan(nchan(nprof)) = ich
        input_ems(nchan(nprof)) = ems_val
     Endif
     Read(*,*,iostat=ios) ich, ivch, ems_val
  End Do
  nchannels = nchan(nprof)    ! Number of valid channels to compute radiances

  !Pack channels and emmissivity arrays
  Allocate(channels(nchan(nprof)))   ! Note these array sizes nchan can vary per
profile
  Allocate(emissivity(nchan(nprof))) ! but for this example assume 1
profile/call with same channels
  Allocate(lprofiles(nchan(nprof)))

  channels(:)      = input_chan(1:nchannels)
  emissivity(:)    = input_ems(1:nchannels)
  ! Build the list of profile indices
```

```
nch = 0_jpim
Do j = 1 , nprof
  DO  jch = 1,nchan(j)
    nch = nch +1_jpim
    lprofiles( nch ) = j
  End Do
End Do
!
!========== Interactive inputs == end ==============
!===================================================

!Initialise error management with default value for
! the error unit number and
! Fatal error message output
Err_unit = -1
!verbosity_level = 1
! All error message output
verbosity_level = 3
Call rttov_errorhandling(Err_unit, verbosity_level)
!
allocate ( setup_errorstatus(nrttovid),stat= alloc_status(1))
allocate (coef(nrttovid),stat= alloc_status(2))
allocate (coef_scatt_ir(nrttovid),stat= alloc_status(3))
allocate (optp(nrttovid),stat= alloc_status(4))

If( any(alloc_status /= 0) ) then
   errorstatus = errorstatus_fatal
   Write( errMessage, '( "mem deallocation error for setup_errorstatus")' )
   Call Rttov_ErrorReport (errorstatus, errMessage, NameOfRoutine)
   Stop
End If
!
!Read and initialise coefficients
    Call rttov_setup (&
    & setup_errorstatus,   &! out
    & Err_unit,            &! in
    & verbosity_level,     &! in
    & nrttovid,            &! in
    & laerosl,             &! in
    & lclouds,             &! in
    & coef,                &! out
    & coef_scatt_ir,       &! out
    & optp,                &
    & instrument)          ! in

if(any(setup_errorstatus(:) /= errorstatus_success ) ) then
   write ( *,* ) 'rttov_setup fatal error'
   stop
endif
deallocate( setup_errorstatus ,stat=alloc_status(1))
If( any(alloc_status /= 0) ) then
   errorstatus = errorstatus_fatal
   Write( errMessage, '( "mem deallocation error for setup_errorstatus")' )
   Call Rttov_ErrorReport (errorstatus, errMessage, NameOfRoutine)
   Stop
End If

! security if input number of channels is higher than number
! stored in coeffs
```

```
  If( nchannels > coef(nrttovid) % fmv_chn ) Then
     nchannels = coef(nrttovid) % fmv_chn
     nchan(nprof) = coef(nrttovid) % fmv_chn
  Endif

  !Open output file which prints results
  Open(IOOUT,file='print.dat',status='unknown',form='formatted',iostat=ios)
  If( ios /= 0 ) Then
     Write(*,*) 'error opening the output file ios= ',ios
     Stop
  Endif

  !===============================================
  !========== Read profile == start ==============
  Open(iup, file='prof.dat',status='old',iostat=ios)
  If( ios /= 0 ) Then
     Write(*,*) 'error opening profile file ios= ',ios
     Stop
  Endif

  ! Do allocation of input profile arrays with the number of levels.
  asw = 1 ! allocate
  allocate( profiles(nprof),stat= alloc_status(1))
  profiles(1) % nlevels = nlevels
  call rttov_alloc_prof
(errorstatus,nprof,profiles,nlevels,coef_scatt_ir(nrttovid),asw, &
    & addaerosl = laerosl, addclouds = lclouds, init = .true. )

  ! Presures are from reference profile
  profiles(1) % p(:) = coef(nrttovid) % ref_prfl_p(:)

  ! read pressure, temp (K), WV (lnq), O3 (ppmv)
  ! take care of doing the unit conversions to
  ! hPa, K and ppmv
  Read(iup,*) profiles(1) % t(:)
  Read(iup,*) profiles(1) % q(:)
  Read(iup,*) profiles(1) % o3(:)
  Read(iup,*) profiles(1) % clw(:)
  ! 2 meter air variables
  Read(iup,*) profiles(1) % s2m % t ,&
        & profiles(1) % s2m % q ,&
        & profiles(1) % s2m % p ,&
        & profiles(1) % s2m % u ,&
        & profiles(1) % s2m % v

  !  Convert lnq to q in ppmv for profile
  profiles(1) %    q(:) = (Exp(profiles(1) %     q(:)) / 1000._JPRB) *
q_mixratio_to_ppmv
  profiles(1) % s2m % q = (Exp(profiles(1) %  s2m % q) / 1000._JPRB) *
q_mixratio_to_ppmv

  ! Skin variables
  Read(iup,*) profiles(1) % skin % t ,&
        & profiles(1) % skin % fastem

  ! Cloud variables
  Read(iup,*) profiles(1) % ctp,&
        & profiles(1) % cfraction
  Close(iup)
```

```
  ! we have an ozone profile
  profiles(1) % ozone_Data =.True.
  ! we do not have CO2 profile
  profiles(1) % co2_Data   =.False.
  profiles(1) %  co2(:) =0._jprb
  ! we do not have n2o profile
  profiles(1) % n2o_data = .false.
  profiles(1) %  n2o(:) =0._jprb
  ! we do not have CH4 profile
  profiles(1) % ch4_Data   =.False.
  profiles(1) %  ch4(:) =0._jprb
  ! we do not have CO profile
  profiles(1) % co_Data   =.False.
  profiles(1) %  co(:) =0._jprb
  ! check Cloud liquid water profile
  profiles(1) % clw_Data   = profiles(1) % clw(1) >= 0.0_JPRB
  ! Other variables from interactive inputs
  profiles(1) % zenangle   = zenith
  profiles(1) % azangle    = azimut
  profiles(1)  % latitude      = LAT
  profiles(1)  % elevation     = ZERHT
  profiles(1)  % sunzenangle   = SUNZANG
  profiles(1)  % sunazangle    = SUNAZANG
  profiles(1)  % addsolar      = solrad
  profiles(1)  % addrefrac     = refrac
  ! surface type
  profiles(1) % skin % surftype  = isurf
  profiles(1) % skin % watertype = nwater
  profiles(1) % aer_data   = laerosl
  profiles(1) % cld_data   = lclouds
  profiles(1) %idg         = 0._jprb
  profiles(1) %ish         = 0._jprb
  if( lclouds ) then
    profiles(1) %cloud(:,:)  = 0._jprb
    profiles(1) %cfrac(:,:)  = 0._jprb
  endif
  !========== Read profile == end ==============
  !============================================

  ! allocate radiance results arrays with number of channels
  asw = 1 ! allocate
  call rttov_alloc_rad (errorstatus,nchannels,radiance,nlevels,asw)
  If( errorstatus /= 0) Then
     errorstatus = errorstatus_fatal
     Write( errMessage, '( "mem allocation error for radiance arrays")' )
     Call Rttov_ErrorReport (errorstatus, errMessage, NameOfRoutine)
     Stop
  Endif
  Allocate( calcemis ( nchannels ) ,stat= alloc_status(2))
  Allocate( input_emissivity ( nchannels ) ,stat= alloc_status(3))

  ! allocate transmittance structure
  Allocate( transmission % tau_layers(nlevels,nchannels ) ,stat=
alloc_status(4))
  Allocate( transmission % tau_total (nchannels ) ,stat= alloc_status(5))

  ! Allocate Error flag
  Allocate( rttov_errorstatus(1)           ,stat= alloc_status(7))
```

```
  If( Any(alloc_status /= 0) ) Then
     errorstatus = errorstatus_fatal
     Write( errMessage, '( "mem allocation error prior to rttov_direct")' )
     Call Rttov_ErrorReport (errorstatus, errMessage, NameOfRoutine)
     Stop
  End If

  ! save input values of emissivities for all calculations
  ! calculate emissivity where the input emissivity value is less than 0.01
  input_emissivity(:) = emissivity(:)
  calcemis(:) = emissivity(:) < 0.01_JPRB
  ! Call RTTOV forward model
    call rttov_direct(  &
          & rttov_errorstatus, &! out error flag
          & nprof,             &! in  number of profiles computed
          & nchannels,         &! in  total number of channels computed for all
profiles
          & channels,          &! in  array of channel indices
          & lprofiles,         &! in  array of profile indices
          & addinterp,         &! in  switch for profile interpolation
          & profiles,          &! in  profile array
          & coef(nrttovid),    &! in  coeff array
          & coef_scatt_ir(nrttovid), &! in  scatterinf coeff array
          & optp(nrttovid),    &! in  optical props array
          & solrad,            &! in  solar calc flag
          & laerosl,           &! in  aerosol radiance flag
          & lclouds,           &! in  cloudy radiance flag
          & calcemis,          &! in  flag for internal emissivity calc
          & emissivity,        &! inout input emissivities per channel
          & transmission,      &! out array of transmittances
          & radiance  )         ! inout computed radiance array

  If ( Any( rttov_errorstatus(:) == errorstatus_warning ) ) Then
     Write ( ioout, * ) 'rttov_direct warning'
  End If

  If ( Any( rttov_errorstatus(:) == errorstatus_fatal ) ) Then
     Write ( 0, * ) 'rttov_direct error'
     Stop
  End If

  ! transfer data to printing arrays
  Allocate(pr_radcld(nchannels)   ,stat= alloc_status(1))
  Allocate(pr_trans(nchannels)    ,stat= alloc_status(2))
  Allocate(pr_emis(nchannels)     ,stat= alloc_status(3))
  Allocate(pr_trans_lev(nlevels,nchannels) ,stat= alloc_status(4))
  If( Any(alloc_status /= 0) ) Then
     errorstatus = errorstatus_fatal
     Write( errMessage, '( "mem allocation error for printing arrays")' )
     Call Rttov_ErrorReport (errorstatus, errMessage, NameOfRoutine)
     Stop
  End If

  pr_radcld(:) = 0.0_JPRB
  pr_trans(:)  = 0.0_JPRB
  pr_emis(:)   = 0.0_JPRB
  pr_trans_lev(:,:) = 0.0_JPRB
  !
  Do j = 1 , nchannels
```

```
      pr_radcld(j) = radiance % cloudy(j)
      pr_trans(j)  = Transmission % tau_total(j)
      pr_emis(j)   = emissivity(j)
      Do ilev = 1 ,  nlevels
         pr_trans_lev(ilev,j) = Transmission % tau_layers(ilev,J)
      Enddo
   Enddo
   !
   !     OUTPUT RESULTS
   !
   NPRINT = 1+ Int((nchannels-1)/10)
   Write(IOOUT,*)' -----------------'
   Write(IOOUT,*)' Instrument ', instrument(3,nrttovid)
   Write(IOOUT,*)' -----------------'
   Write(IOOUT,*)' '
   WRITE(IOOUT,777) profiles(nprof)%zenangle,profiles(nprof)%azangle, &
       &
csun,profiles(nprof)%sunzenangle,profiles(nprof)%sunazangle,profiles(nprof)%skin
%surftype,&
       &
profiles(nprof)%skin%watertype,profiles(nprof)%latitude,profiles(nprof)%elevatio
n,cref,caer,ccld,&
       & instrument(2,nrttovid)

   WRITE(IOOUT,*)'CHANNELS PROCESSED:'
   WRITE(IOOUT,111) (channels(J), J = 1,nchannels)
   WRITE (IOOUT,*)' '
   Write(IOOUT,222) radiance % bt(:)
   Write(IOOUT,*)' '
   Write(IOOUT,*)'CALCULATED RADIANCES: SAT =', instrument(2,nrttovid)
   Write(IOOUT,222) radiance % total(:)
   Write(IOOUT,*)' '
   Write(IOOUT,*)'CALCULATED OVERCAST RADIANCES: SAT =', instrument(2,nrttovid)
   Write(IOOUT,222) pr_radcld(:)
   Write (IOOUT,*)' '
   Write(IOOUT,*)'CALCULATED SURFACE TO SPACE TRANSMITTANCE: S'&
              & ,'AT =',instrument(2,nrttovid)
   Write(IOOUT,4444) pr_trans(:)
   Write (IOOUT,*)' '
   Write(IOOUT,*)'CALCULATED SURFACE EMISSIVITIES '&
              & ,'SAT =',instrument(2,nrttovid)
   Write(IOOUT,444) pr_emis(:)
   !
   !
   If(nchan(nprof) <= 20)Then
      Do  NP = 1 , NPRINT
         Write (IOOUT,*)' '
         Write (IOOUT,*)'Level to space transmittances for channels'
         Write(IOOUT,1115) (channels(j),&
                 & J = 1+(NP-1)*10,Min(10+(NP-1)*10,nchannels))
         Do  ILEV = 1 , NLEVELS
            Write(IOOUT,4445)ILEV,(pr_trans_lev(ilev,J),&
                    & J = 1+(NP-1)*10,Min(10+(NP-1)*10,nchannels))
         End Do
         Write(IOOUT,1115) (CHANNELS(J),&
                 & J = 1+(NP-1)*10,Min(10+(NP-1)*10,nchannels))
      End Do
   Endif
   !
```

```
  ! Deallocate arrays
     deallocate( channels   ,stat=alloc_status(1))
     deallocate( lprofiles  ,stat=alloc_status(2))
     deallocate( emissivity ,stat=alloc_status(3))
     deallocate( input_emissivity ,stat=alloc_status(4))
     deallocate( calcemis   ,stat=alloc_status(5))

    ! deallocate transmittances
     Deallocate( transmission % tau_total   ,stat= alloc_status(7))
     Deallocate( transmission % tau_layers  ,stat= alloc_status(8))
    ! dealloc printing arrays
     deallocate(pr_radcld ,stat= alloc_status(9))
     deallocate(pr_trans ,stat= alloc_status(10))
     deallocate(pr_emis ,stat= alloc_status(11))
     deallocate(pr_trans_lev ,stat= alloc_status(12))
     If( any(alloc_status /= 0) ) then
        errorstatus = errorstatus_fatal
        Write( errMessage, '( "mem deallocation error")' )
        Call Rttov_ErrorReport (errorstatus, errMessage, NameOfRoutine)
        Stop
     End If

     asw = 0 ! deallocate radiance arrays
     call rttov_alloc_rad (errorstatus,nchannels,radiance,nlevels,asw)
     If(errorstatus /= errorstatus_success) Then
        Write( errMessage, '( "radiance deallocation error")' )
        Call Rttov_ErrorReport (errorstatus, errMessage, NameOfRoutine)
     Endif

     asw = 0 ! deallocate profile arrays
     call rttov_alloc_prof
(errorstatus,nprof,profiles,nlevels,coef_scatt_ir(nrttovid),asw, &
    & addaerosl = laerosl, addclouds = lclouds )
     deallocate( profiles,stat=alloc_status(1))
     If(errorstatus /= errorstatus_success .or. alloc_status(1) /= 0) Then
        Write( errMessage, '( "profile deallocation error")' )
        Call Rttov_ErrorReport (errorstatus, errMessage, NameOfRoutine)
     Endif

     Call rttov_dealloc_coef (errorstatus,
coef(nrttovid),coef_scatt_ir(nrttovid),optp(nrttovid))
     If(errorstatus /= errorstatus_success) Then
        Write( errMessage, '( "coef deallocation error")' )
        Call Rttov_ErrorReport (errorstatus, errMessage, NameOfRoutine)
     Endif

   !Close output file
    Close(IOOUT,iostat=ios)
    If( ios /= 0 ) Then
      Write(*,*) 'error closing the output file ios= ',ios
      Stop
    Endif

111  FORMAT(1X,10I8)
1115 Format(3X,10I8)
222  Format(1X,10F8.2)
444  Format(1X,10F8.3)
4444 Format(1X,10F8.4)
4445 Format(1X,I2,10F8.4)
```

```
777 FORMAT( &
       & ' ZENITH ANGLE        =',F7.2,/ &
       & ' AZIMUTH ANGLE       =',F7.2,/&
       & ' SOLAR RADIATION     =',A7,/&
       & ' SOLAR ZENITH ANGLE =',F7.2,/&
       & ' SOLAR AZIMUTH ANGLE=',F7.2,/ &
       & ' SURFACE TYPE        =',I7,/&
       & ' WATER TYPE          =',I7,/ &
       & ' LATITUDE            =',F7.2,/&
       & ' ELEVATION           =',F7.2/&
       & ' REFRACTION          =',A7,/&
       & ' AEROSOLS            =',A7,/&
       & ' CLOUDS              =',A7,//,&
       &'CALCULATED BRIGHTNESS TEMPERATURES: SAT =',I2 )

End Program example_fwd
```

End of User Guide