RTTOV_8_7 Users Guide

Roger Saunders
Met Office, Exeter, UK
&
Pascal Brunel
MétéoFrance, CMS, Lannion, France

This documentation 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 Météo France.

Change record			
Version	Date	Author / changed by	Remarks
1.0	17/12/02	P. Brunel	Initial draft
1.1	19/12/02	R. Saunders	Comments/changes added
1.2	12/02/03	P. Brunel	For RTTOV7.2 release
1.3	09/02/04	R. Saunders	Update for RTTOV-8 release to beta testers
1.4	06/05/04	R. Saunders	Updated again after beta testers comments
1.5	02/07/04	R. Saunders	Updated after DRI and RT development meeting
1.6	17/08/04	R.Saunders	Final update before release DRI actions in yellow
1.7	02/11/04	R. Saunders	Version for release included known bugs section+ ECMWF changes+refer to model release version
1.8	14/06/05	R. Saunders	Corrected Table 3.
1.9	17/11/05	R. Saunders	Updated for version 8-7 release

1. Introduction and Scope

This document gives an overview of the RTTOV_8_7 fast radiative transfer model (in sec 2), the differences from RTTOV-7 (in sec 3), how to install the RTTOV_8_7 fast radiative transfer model code on a UNIX/LINUX platform and run it (sec 4) and how to apply it to the users particular application (sec 5). The procedure for reporting bugs or making comments to the NWP SAF is given in sec 6. Finally a frequently asked questions (FAQ) section is provided at section 7. If you want to order a copy of the RTTOV_8_7 code send an email to mailto:nwpsaf@metoffice.gov.uk requesting a copy of the code. You will need to sign a RTTOV_8_7 licence form before you are given access to the code.

The old RTTOV-6 code is still available in FORTRAN-90 or FORTRAN-77 but will no longer be upgraded for new instruments. The old RTTOV-7 code is still available (but not in FORTRAN-77) and will not provide all the capability of RTTOV_8_7. Bugs reported with RTTOV-6/7 will continue to be announced and users informed of fixes on the RTTOV web pages and email lists. Coefficient files for RTTOV-6/7 will continue to be made available from the NWP-SAF web site. The RTTOV-7 code took part in the Garand fast model intercomparison (see Garand et. al. 2001 for details) and has been distributed to over 50 users worldwide. RTTOV_8_7 is a complete rewrite of RTTOV-7 using the FORTRAN-90 features.

The RTTOV_8_7 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 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_8_7 page.

2. Overview of RTTOV_8_7 and limitations

This section gives a brief overview of the RTTOV_8_7 model and its limitations. More details can be found in the references given in this section. RTTOV_8_7 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_7 is the latest version. The model allows rapid simulations (~1 ms for 40 channel ATOVS on a HP workstation) 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 RTTOV_8_7 is water vapour, and optionally ozone and carbon dioxide can be variable with all other constituents assumed to be constant. The state vector for RTTOV_8_7 is given in Annex L. Not all parameters have to be supplied as actual values. RTTOV_8_7 can accept state vectors on any set of pressure levels *but* the coefficients in the initial release are supplied for the 43 pressure levels defined in Table 1. To work on other pressure levels coefficients will be generated with their own transmittances on the required levels.

Currently the spectral range of the RTTOV_8_7 model is 3-20μm (500 – 3000 cm^{-1}) in the infrared governed by the range of the GENLN2 line-by-line dataset 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 or as improved line-by-line model data are generated. 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 for the input state vector values. Given a state vector, **x,** a radiance vector, **y**, is computed:

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

where *H* is the radiative transfer model (also referred to as the observation operator).

Level number	Pressure (hPa)	Tmax deg K	Tmin degK	Qmax Kg/Kg	Qmin Kg/Kg	O_3max Kg/Kg	O_3min Kg/Kg	O_3 Ref Kg/Kg
1	0.1	335.5	162.0	4.38E-05	1.20E-06	1.63E-05	7.00E-07	9.69E-06
2	0.3	335.8	173.1	4.65E-05	1.20E-06	1.69E-05	1.00E-06	1.00E-05
3	0.7	352.8	168.9	4.61E-05	1.20E-06	1.70E-05	2.10E-06	1.01E-05
4	1.4	354.4	160.9	4.51E-05	1.20E-06	1.71E-05	2.11E-06	1.02E-05
5	2.6	349.4	160.5	4.29E-05	1.20E-06	1.71E-05	2.11E-06	1.02E-05
6	4.4	328.8	160.3	4.26E-05	1.20E-06	1.71E-05	2.11E-06	1.02E-05
7	7.0	321.4	158.5	4.36E-05	1.20E-06	1.71E-05	2.11E-06	1.02E-05
8	10.4	300.3	154.7	4.35E-05	1.20E-06	1.71E-05	2.11E-06	1.02E-05
9	14.8	295.0	154.9	4.01E-05	1.20E-06	1.72E-05	2.11E-06	1.01E-05
10	20.4	289.0	151.1	4.03E-05	1.20E-06	1.61E-05	2.11E-06	9.36E-06
11	27.3	286.5	151.2	4.18E-05	1.20E-06	1.60E-05	2.03E-06	8.10E-06
12	35.5	285.3	151.6	3.62E-05	1.20E-06	1.14E-05	8.33E-07	6.72E-06
13	45.3	284.2	152.5	3.43E-05	1.20E-06	1.11E-05	5.49E-07	5.19E-06
14	56.7	283.8	154.2	3.33E-05	1.20E-06	9.82E-06	2.85E-07	3.72E-06
15	70.0	282.7	155.7	3.23E-05	1.20E-06	6.46E-06	2.13E-07	2.58E-06
16	85.2	282.7	153.9	3.01E-05	1.20E-06	5.31E-06	1.71E-07	1.72E-06
17	102.1	281.5	151.5	2.90E-05	1.20E-06	4.10E-06	6.96E-08	1.19E-06
18	122.0	280.1	156.7	3.58E-05	1.20E-06	3.63E-06	1.18E-08	8.45E-07
19	143.8	278.6	157.4	8.61E-05	1.20E-06	3.06E-06	1.03E-08	6.50E-07
20	168.0	278.8	159.7	1.64E-03	1.20E-06	2.24E-06	8.72E-09	5.27E-07
21	194.4	280.1	163.2	2.79E-03	1.20E-06	1.64E-06	7.43E-09	4.13E-07
22	222.9	282.3	165.3	4.44E-03	1.20E-06	1.47E-06	7.14E-09	3.03E-07
23	253.7	285.3	166.7	7.64E-03	1.20E-06	1.09E-06	1.20E-08	2.11E-07
24	286.6	288.7	167.6	1.12E-02	1.20E-06	7.60E-07	1.16E-08	1.56E-07
25	321.5	294.0	170.6	1.68E-02	1.20E-06	5.90E-07	1.59E-08	1.23E-07
26	358.3	300.5	174.2	2.54E-02	1.20E-06	3.83E-07	7.94E-09	1.08E-07
27	396.8	306.4	175.3	3.62E-02	1.20E-06	3.13E-07	1.13E-08	1.01E-07
28	437.0	312.2	178.8	5.00E-02	1.20E-06	2.45E-07	6.58E-09	9.60E-08
29	478.5	317.2	182.1	6.50E-02	1.95E-06	2.34E-07	6.24E-09	9.17E-08
30	521.5	321.1	185.0	7.75E-02	4.51E-06	2.31E-07	4.95E-09	8.91E-08
31	565.5	325.2	187.8	8.95E-02	1.04E-05	2.13E-07	2.76E-09	8.47E-08
32	610.6	328.2	190.3	1.05E-01	1.29E-05	2.01E-07	2.41E-09	8.12E-08
33	656.4	333.0	192.8	1.24E-01	1.42E-05	2.07E-07	2.27E-09	7.78E-08
34	702.7	336.8	195.4	1.41E-01	1.71E-05	2.21E-07	2.07E-09	7.57E-08
35	749.1	340.7	197.1	1.59E-01	3.63E-05	1.87E-07	7.24E-10	7.12E-08
36	795.1	344.4	198.4	1.78E-01	5.34E-05	1.91E-07	7.24E-10	6.63E-08
37	840.0	348.0	199.0	2.00E-01	6.42E-05	1.81E-07	7.24E-10	6.16E-08
38	882.8	350.3	197.5	2.14E-01	6.68E-05	1.73E-07	8.10E-10	5.68E-08
39	922.5	352.2	195.5	2.40E-01	6.57E-05	1.68E-07	8.10E-10	5.21E-08
40	957.4	354.7	188.2	2.70E-01	6.57E-05	1.65E-07	8.10E-10	4.79E-08
41	985.9	356.6	155.0	2.79E-01	6.57E-05	1.63E-07	8.10E-10	4.44E-08
42	1005.4	357.9	135.0	2.82E-01	6.57E-05	1.62E-07	8.10E-10	4.20E-08
43	1013.3	385.9	135.0	2.84E-01	6.57E-05	1.62E-07	8.10E-10	4.10E-08

Table 1 Pressure levels adopted for RTTOV-7 coefficient profile limits within which the transmittance calculations are valid. The default ozone profile is also given in the right hand column. The levels and limits will change when the newer RTTOV_8_7 coefficient files are released.

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

[¶] Includes TIROS-N

Table 2. Platforms supported by RTTOV_8_7 as at 17 Nov 2005 in normal text. Platforms in italics are not yet supported by RTTOV_8_7 but soon will be.

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 $\mathbf{x_0}$:

$$\boldsymbol{\delta}\mathbf{y} = \mathbf{H}(\mathbf{x_0})\boldsymbol{\delta}\mathbf{x} \qquad (2)$$

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 from *each level* in the profile given a unit perturbation at any level of the profile vectors or in any of the surface/cloud parameters. It shows clearly, for a given profile, which levels 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 $\mathbf{H(x_0)}$ 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, for a given change in atmospheric profile, **δx,** about an initial atmospheric state $\mathbf{x_0}$. The tangent linear routines all have *TL* as an ending. Conversely the adjoint routines (ending in *AD*) compute the change in the gradient of any scalar quantity with respect to the atmospheric state, $\mathbf{x_0}$**,** given a change in the gradient of that quantity with respect to the radiances, **y.** These routines are normally used as part of the variational assimilation of radiances. For users only interested in the forward model the *TL/AD/K* routines are not required.

Sensor	RTTOV id	Sensor Channel #	RTTOV-7 Channel #	RTTOV-8 Channel #
HIRS	0	1 to 19	1 to 19	1 to 19
MSU	1	1 to 4	1 to 4	1 to 4
SSU	2	1 to 3	1 to 3	1 to 3
AMSU-A	3	1 to 15	1 to 15	1 to 15
AMSU-B	4	1 to 5	1 to 5	1 to 5
AVHRR	5	3b to 5	1 to 3	1 to 3
SSMI	6	1 to 7	1 to 7	1 to 4
VTPR1	7	1 to 8	1 to 8	1 to 8
VTPR2	8	1 to 8	1 to 8	1 to 8
TMI	9	1 to 9	1 to 5	1 to 9
SSMIS	10	1 to 24*	1 to 24*	1 to 21
AIRS	11	1 to 2378	1 to 2378	1 to 2378
HSB	12	1 to 4	1 to 4	1 to 4
MODIS	13	1 to 17	1 to 17	1 to 17
ATSR	14	1 to 3	1 to 3	1 to 3
MHS	15	1 to 5	1 to 5	1 to 5
IASI	16	1 to 8461	N/A	1 to 8461
AMSR	17	1 to 14	1 to 14	1 to 7
MVIRI	20	1 to 2	1 to 2	1 to 2
SEVIRI	21	4 to 11	1 to 8	1 to 8
GOES-Imager	22	1 to 4	1 to 4	1 to 4
GOES-Sounder	23	1 to 18	1 to 18	1 to 18
GMS/MTSAT imager	24	1 to 4	1 to 4	1 to 4
FY2-VISSR	25	1 to 2	1 to 2	1 to 2
FY1-MVISR	26	1 to 3	1 to 3	1 to 3
CriS	27	TBD	N/A	TBD
CMISS	28	TBD	N/A	TBD
VIIRS	29	TBD	N/A	TBD
WINDSAT	30	1 to 10	N/A	1 to 5

**channels 19-21 are not simulated accurately*

Table 3. Instruments supported by RTTOV_8_7 as at 17 Nov 2005. Sensors in italics are not yet supported by RTTOV_8_7 but soon will be.

The model can simulate both clear sky radiances and cloudy radiances. It uses an approximate form of the atmospheric radiative transfer (RT) equation. 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:

$$L(v,\theta) = (1-N)L^{Clr}(v,\theta) + NL^{Cld}(v,\theta) \qquad (3)$$

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

2.1 Simulation of clear air radiances

If *N*, the cloud cover parameter is set to zero and the LWP path 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 3 being zero. $L^{Clr}(\nu,\theta)$ can be written as:

$$L^{Clr}(\nu,\theta) = \tau_s(\nu,\theta)\varepsilon_s(\nu,\theta)B(\nu,T_s) + \int_{\tau_s}^{1} B(\nu,T)d\tau + (1-\varepsilon_s(\nu,\theta))\tau_s^2(\nu,\theta)\int_{\tau_s}^{1}\frac{B(\nu,T)}{\tau^2}d\tau \qquad (4)$$

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 and Matricardi et. al. (2003) for RTTOV_8_7 predictors. The code supports either set of predictors with the selection being made according to the coefficient file supplied. More details are given in the RTTOV_8_7 science and validation plan.

To compute ε_s over water there are fast surface emissivity routines for both the infrared, ISEM, (Sherlock, 1999) and for the microwave, FASTEM-1 (English and Hewison, 1998) which is no longer supported for RTTOV_8_7 (but still works) or 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 whether FASTEM-2 or 3 is used.

calcemiss	RTTOV coeff file version	Input ε	Forward Output ε	Tangent Linear Output $\partial\varepsilon$
			INFRARED CHANNELS	

true	7 or 8	0	Land=0.98/sea-ice=0.99/
			sea= ε_{ISEM}
				$\partial\varepsilon$ about 0.98/0.99/ε_{ISEM}

false	7 or 8	ε_{user}	ε_{user}	$\partial\varepsilon$ about ε_{user}
			MICROWAVE CHANNELS	

true	7	0	Land/sea-ice computed from coeffs in
			`prof % skin % fastem(1:5)`
			sea= $\varepsilon_{FASTEM1}$
				Land/sea-ice $\partial\varepsilon$, about $\varepsilon_{FASTEM1}$
				sea $\partial\varepsilon$, computed from ∂u, ∂v, ∂sst about $\varepsilon_{FASTEM1}$

true	7	-1	Land/sea-ice computed from coefs in
			`prof % skin % fastem(1:5)`
			sea= $\varepsilon_{FASTEM2}$
				Land/sea-ice $\partial\varepsilon$ about $\varepsilon_{FASTEM2}$
				sea $\partial\varepsilon$, computed from ∂u, ∂v, ∂sst about $\varepsilon_{FASTEM2}$

true	8	0	Land/sea-ice computed from coefs in
			`prof % skin % fastem(1:5)`
			sea= $\varepsilon_{FASTEM3}$
				Land/sea-ice $\partial\varepsilon$ about $\varepsilon_{FASTEM3}$
				sea $\partial\varepsilon$, computed from ∂u, ∂v, ∂sst about $\varepsilon_{FASTEM3}$

false	7 or 8	ε_{user}	ε_{user}	$\partial\varepsilon$ 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 the simulation of cloud affected radiances $L^{Cld}(\nu,\theta)$ is defined as:

$$L^{Cld}(\nu,\theta) = \tau_{Cld}(\nu,\theta)\, B(\nu,T_{Cld}) + \int_{\tau_{Cld}}^{1} B(\nu,T)d\tau \qquad (5)$$

where τ_{Cld} (ν,θ) is the cloud top to space transmittance and T_{Cld} the cloud top temperature, 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 microwave frequencies the liquid water profile can be supplied in the profile array as cloud liquid water concentration in units of kg/kg. Only layers from the surface to the level *mwcldtop* set in the *rttov_const.F90* file (see below) are taken into account in the computation. The default value set is for a level at 321 hPa. For cloud water drops scattering is assumed to be negligible below 200 GHz and so it follows that the extinction per unit mass is independent of radius and thus the sensitivity of changes in optical depth to changes in liquid water mass is independent of the drop-size distribution. This allows a calculation of the optical depth if an assumed dependence of the permittivity of the liquid water with temperature is assumed. Ice extinction is assumed to be zero so the input cloud water profile is all assumed to be liquid. Scattering becomes important for ice crystals above 100GHz. If the variable *profiles(1) % clw_data* is set to false the liquid water path transmittance calculation is not performed regardless of the input profile which reduces execution time of the model.

For the standard RTTOV model at infrared frequencies clouds are assumed to be at one level, have unit emissivity and a top at a fixed cloud top pressure with a fractional coverage for each input profile. The outputs of *RTTOV* can be used however to simulate a more realistic multilevel infrared and microwave cloudy radiance and the *RTTOV_CLD* routines supplied with RTTOV_8_7 provide this capability. *RTTOVCLD* and its associated TL/K/AD routines take a profile input on 43 levels for the normal state variables in and the gaseous transmittances are computed on the 43 levels. In addition *RTTOVCLD* also takes a profile of temperature, cloud cover, cloud liquid water (kg/kg) and cloud ice water (kg/kg) on user defined model pressure levels and computes infrared and/or cloudy radiances for multilevel and multiphase cloud fields. The clear and cloudy radiative transfer computation is done on the user defined model levels in *RTTOV_CLD*. The advantage of using this method for computing cloudy microwave radiances is there is no interpolation to the RTTOV levels for the cloudy radiance computations and there is a consistent random-overlap scheme with the infrared. More details are given in Chevallier *et. al*. (2001) and the RTTOV_8_7 science and validation plan for this enhancement of *RTTOV*. Note that the values assumed for the infrared cloud optical properties have changed in RTTOV_8_7 from RTTOV-7 to be more realistic and the user can select various assumptions for the ice cloud microphysics as given in Table 5.

RTTOV_8_7 variable	Parameter	Reference
kradip=0	Ice crystal type	Hexagonal columns
kradip=1	Ice crystal type	Crystal Aggregates
kice=0	Ice effective diameter scheme	Ou and Liou (1995)
kice=1	Ice effective diameter scheme	Wyser (1998)
kice=2	Ice effective diameter scheme	Boudala *et al.* (2002)
kice=3	Ice effective diameter scheme	McFarquhar (2003)

Table 5. Options for ice crystal parametrisations in RTTOV_8_7

In the same way as *RTTOV_CLD* provides additional routines to compute cloudy radiances RTTOV_8_7 also has a set of routines to compute scattering effects from hydrometeors at microwave frequencies using the delta-Eddington approximation. This is the *RTTOV_SCATT* code which calls RTTOV_8_7 for the clear air part but adds the scattering effects from water droplets in the profile. This can be used for simulating rain affected microwave radiances. The input profiles are the same as for *RTTOV_CLD*. More details on the methodology are provided in the RTTOV_8_7 science and validation report. Note that an additional coefficient file is required when running *RTTOV_SCATT* as indicated in the script *test_rttovscatt.ksh* for the mie table values (e.g. for SSM/I it is *mietable_dmsp_ssmi.dat*). Coefficient files for microwave sensors not supplied in the tar file will be provided on the RTTOV_8_7 web page for download.

2.3 Current limitations of RTTOV_8_7

There are a number of limitations of RTTOV_8_7 the user should be aware of. Some are fundamental and some are not. The main ones are listed here:

- RTTOV_8_7 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).
- RTTOV_8_7 does not include any reflected solar component (e.g. to simulate sunglint).
- RTTOV_8_7 only allows for water vapour, ozone and carbon dioxide to be variable gases with all others included in the mixed gases transmittance calculation.
- RTTOV_8_7 with coefficients as supplied can only provide simulations with a 43 level profile as input on the defined pressure levels in Table 1. However it is planned to make available coefficient files for the advanced IR sounders on more levels a few months after the initial release of RTTOV_8_7.
- RTTOV_8_7 can only simulate radiances for instruments for which a coefficient file has been generated. The instruments currently supported are listed in Table 3.
- 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 available from the RTTOV-8 web site.
- RTTOV_8_7 does not include the variation of the zeeman effect with magnetic field strength for the high peaking AMSU-A and SSMIS channels. Only a constant correction factor is included.

3. Changes from RTTOV-7

There are a number of important changes from RTTOV-7 which are listed below so the user is aware of them. Note that old RTTOV-7 coefficient files are still valid to work with RTTOV_8_7 code.

- The code is completely rewritten so that passing of variables is now achieved through structures and all the arrays are allocatable to be more efficient in terms of memory usage. See annex L for the definitions of the structures.
- There are more index arrays which need to be passed into the RTTOV routines to take account of the polarisation of the microwave channels. These need care in setting up, although routines are provided to help the user. Section 5.2 describes this aspect in more detail.
- The required units for water vapour and ozone profile concentrations for input to RTTOV_8_7 are changed from kg/kg to ppmv. Variables in the code are defined to allow the user to convert (see *rttov_const.F90*) and routines *rttov_q2v.F90* and *rttov_v2q.F90* are supplied with the code.
- The ozone and cloud liquid water profiles are now optional and are activated with logical flags (e.g. *profiles%ozone_data*)
- Optionally a CO_2 profile can now be provided.
- The number of levels for the input profile is now variable and defined by the coefficient file. Initially 43L profiles (as for RTTOV-7) will remain the standard for the coefficient files supplied but later coefficients on more levels are planned.
- The optical depth prediction can be either as in RTTOV-7 (for backward compatibility) or using new predictors which separates out the water vapour line and continuum absorption. As for the levels the coefficient file supplied provides the selection of predictors.
- The selection of whether the surface emissivity is computed internally or not is now via a logical *calcemiss* and this can be selected for each input channel. If *calcemiss* is .false. the value input is used.
- There is an improved version of the microwave sea surface emissivity model FASTEM-3 but FASTEM-2 and FASTEM-1 can still be invoked (the latter is not supported).

- The selection of FASTEM-2 or FASTEM-3 is now driven by the coefficient file supplied not the input emissivity value as for RTTOV-7. However the selection of FASTEM-1 or FASTEM-2 is still as for RTTOV-7 using an old FASTEM-2 coefficient file and assigning the input to 0 or -1.
- If the internal emissivity model is invoked the microwave emissivity values returned are different from RTTOV-7.
- The code now allows simulations of polarimetric radiometers such as WINDSAT.
- There are changes to the infrared cloud optical properties in the RTTOV_CLD programme to improve the cloudy IR simulations and the option of choosing different parameterisations.
- A new wrapper programme RTTOV_SCATT has been developed to allow the simulation of rain affected microwave radiances.

It should be noted that for those RTTOV-7 users who want to use the new RTTOV_8_7 code but do not want to change their code interfaces to RTTOV some subroutines are provided to allow RTTOV_8_7 to be called from an RTTOV-7 like interface. However this is only for a limited set of options for the forward model only. More details in Annex K.

4. FORTRAN-90 UNIX/LINUX installation instructions

RTTOV_8_7 is designed for UNIX/Linux systems. The software is now successfully tested on SUN, HP, SGI, Linux PC, Cray T3E and Fujitsu VPP systems and for a range of compilers listed in the Makefile supplied.

The following system components are needed before running RTTOV_8_7:

* UNIX or Linux operating system

* Fortran 90 compiler

* make utilities

* gzip and gunzip

* about 100 MByte of free disk space (160 MByte if you require AIRS coefficient files)

Some basic information on installing the RTTOV_8_7 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 *rttov85.tar.gz* and be copied to your ‘top’ RTTOV directory (e.g. *~user/rttov85*) from which subdirectories will be created. Text in *italics* refers to specific commands to execute during the installation or file names.

4.1 Unpacking the code

First uncompress the tar file:
gunzip rttov87.tar.gz

and expand it:
tar -xvf rttov87.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
-	*rtcoef_scatt*	RTTOV-8 scattering coeffs
-	*test*	Output of test programs run on user's machine
-	*reftest*	Output of test programs run by NWP SAF
-	*reftest_rttov8*	Output of RTTOV-8 predictor tests run by NWP SAF
-	*docs*	Documentation

There is also *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 coefficient files and scattering coefficient files are not included but can be downloaded from the RTTOV-7/8 web pages if required. Also new coefficient files with RTTOV-8 predictors for more sensors and more FASTEM-3 coefficients will be made available on the RTTOV-8 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.

4.2 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, rttovcld_test.F90, rttovscatt_test.F90)* in *src* for complete testing of the RTTOV, RTTOV_CLD 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. Note that for *rttovscatt_test.F90* there is a F77 routine *lapack.f* used in the test program so you also need to set a F77 compiler option if you require the scattering code to be compiled. A selection of compiler flags for different platforms are included so if you are running using one of these compilers you should be able to just uncomment the relevant section. Once this is done you can run the makefile. Note there is no longer a need to compile the code as double precision as the *parkind* module controls this. In that module single precision is achieved by defining *JPRB* to be the same as *JPRM.* There are various options for running the makefile:
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 clear air RTTOV (most users) or "*make all*" to compile all the code. The full options are:

- *make basic (default): compile classical RTTOV code*
- *make cld: compile cloud code with input cloud profile*
- *make scat: compile scattering code*
- *make all: compile all the code*

With luck the code will compile and produce an executable *tstrad.out* for the basic RTTOV tests, and/or optionally *rttovcld_test.out* for the RTTOVCLD tests and *rttovscatt_test.out* for the RTTOVSCATT tests. The Makefile should move these executable files to the *scripts* subdirectory.

If the compilation was not successful then either edit the makefile again 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.* This only provides code to test the RTTOV routines and not the RTTOV_CLD or RTTOV_SCATT routines. If you want to test the cloudy/scatt routines also restart from step 2 and *rm -f tstrad.o* and recompile *rttovcld_test.F90* or *rttovscatt_test.F90* in place of *tstrad.F90.* If on the other hand there were compiler errors reported when compiling the code please report these back to the NWP SAF (see section 7).

4.3 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_full.ksh* for testing the tangent linear, adjoint and K codes. 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 sensor tests. You don't need to run *test_full.ksh* if you are not interested in running the TL/AD/K codes. Note that there are tests for the AIRS instrument included but not invoked in the scripts as you will need to download the AIRS coefficient file first. If you want to run AIRS simulations it is necessary to activate the AIRS tests in the test scripts by uncommenting the line in the script.

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* and *data* respectively. Output files from the runs on the

NWPSAF development computer are given in *reftest.* 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. Note however the TL/AD/K test outputs from running *test_full.ksh* 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 6 and sample difference files provided in *reftest*. These differences are normal. 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 and the RTTOV-8 design document.

```
<  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
```

Table 6. Example of typical differences found between NWPSAF generated output and that from the users machine. The numbers can differ from run to run.

5. Running RTTOV_8_7 for your applications

To run RTTOV_8_7 for a user's application the test programs supplied **tstrad.F90**, **example_fwd.F90** can be used as a rough 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. A script **move_F90_to_f90.ksh** is provided for this purpose. Use the modules **rttov_types** and **rttov_const** in your program for the definition of derived types (see annex L). It is also important to allocate the various input and output arrays for **rttov_direct** to the correct dimensions (see **tstrad** for example). Annex M gives an example program with comments to guide the user and this source code is provided in the tar file.

There are only 4 subroutines that must be called which are **rttov_setup** which does all the general setup tasks in one go, **rttov_setupchan** which given a list of valid channel numbers computes the various index array sizes required and **rttov_setupindex** which sets up the various indices and emissivity arrays (see below) and finally the call to **rttov_direct** itself which actually computes the radiances. 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.

Users requiring the TL/AD/K routines may also call these (e.g. **rttov_k**) as required. 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 such as the definition of channel numbers for that instrument etc. The following steps are recommended in coding a program which calls RTTOV_8_7.

5.1. 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_errorhandling**, an optional routine which can be called at any time.
- Read the coefficient file by calling **rttov_readcoeffs** with the instrument triplet (platform id, satellite id, sensor id) and the optional arguments if required (see annex B). One call for each coefficient file.
- Initialise the coefficient file by calling **rttov_initcoeffs**
- The command **rttov_setup** is a more general tool which includes a call to **rttov_errorhandling** and calls to **rttov_readcoeff** and **rttov_initcoeffs** for several coefficient files and so may be simpler to use.

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 RTTOV_8_7. These are defined in Table 7 below.

Table 7 Coefficient file options for RTTOV-7 and 8.

Surface emissivity model options	RTTOV-7 Old optical depth predictors	RTTOV-8 Old optical depth predictors used in RTTOV-7	RTTOV-8 New optical depth predictors+
FASTEM-1/2	Yes (0/-1 in input)	Yes (0/-1 in input)	Yes (0/-1 in input)
FASTEM-3	No	Yes defined in new coeff file	Yes defined in new coeff file
ISEM-6	Yes (0 in input)	Yes (0 in input)	Yes (0 in input)
RTTOVSCATT	RTTOV-7	RTTOV-8 old optical depth predictors	RTTOV-8 new optical depth predictors
RTTOVSCATT+	No	Yes	Yes

+coefficient files are available from web site.

5.2. Setting up input arrays before each call to RTTOV

```
Call rttov_direct(errorstatus,nfrequencies,nchannels,nbtout,nprofiles,
channels,polarisations,lprofiles,profiles,coef,addcloud,calcemiss,
emissivity,transmission,radiancedata)
```

Setting up the input arrays is one aspect that is more complicated for RTTOV_8_7 than RTTOV-7 due to the inclusion of the polarimetry option and time saving option for "SSM/I like" instruments for RTTOV_8_7 and requires some careful explanation. Table 8 gives examples of these arrays for three different sensors and for 2 profiles per RTTOV call. **nfrequencies** is the number of separate frequencies at which a sensor measures radiance e.g. for 1 profile for HIRS this is 19 (the visible channel is excluded), for SSM/I 4 (but note it will be 7 for old RTTOV-7 coeff files) for AMSU-A it is 15 and for AVHRR 3. Inside RTTOV_8_7 separate radiance calculations can be performed for each polarisation at a single frequency resulting in **nchannels** radiance streams in the code:

$$nchannels = \sum_{i=1}^{i=nfrequencies} npol(i)$$

Note **npol** (number of polarisations per channel) is 1 for all infrared channels but can vary with channel for microwave sensors (e.g. SSM/I, WINDSAT) from 1 to 4. These radiance streams can then be combined on output (e.g. for AMSU) to provide a single brightness temperature/radiance or not combined to give separate brightness temp/radiance for each polarisation and frequency (e.g. for SSM/I). The **nbtout** parameter defines the number of computed brightness temperatures/radiances output from RTTOV_8_7. The arrays **channels, polarisations and lprofiles** contain the corresponding channel, polarisation and profile indices for each computed radiance. Figures 1, 2 and 3 illustrate this structure for AVHRR, SSM/I and AMSU-B. All infrared sensors will be like HIRS or AVHRR.

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 4. 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 (note this is different for each polarisation), and if set to *.true.* the FASTEM model is used. The FASTEM model used in the model now depends on the coefficient file provided. Microwave coefficient files for both FASTEM-1/2 and FASTEM-3 are available for most sensors. Note if a FASTEM-1/2 coefficient file is used input values of 0 and -1 in the *emissivity* array can still be used to select FASTEM-1 or 2 as for RTTOV-7.

Table 8. Examples of RTTOV_8_7 input parameters

Input arrays	HIRS (2 profiles/call)	SSM/I (2 profiles/call)	AMSU-B (2 profiles/call)
nfrequencies	38	8	10
nchannels	38	14	20
nbtout	38	14	10
nprofiles	2	2	2
channels(nfrequencies)	1,2,3 ..,19,1,2,3..,19	1,2,3,4,1,2,3,4	1,2,3,4,5,1,2,3,4,5
polarisations(nchannels,1)	1,2,3 ..,19,20,21,..38	1,3,4,6,8,10,11,13	1,3,5,7,9,11,13,15,17,19
polarisations(nchannels,2)	1,2,3 ..,19,20,21,..38	1,1,2,3,3,4,4,5,5…8,8	1,1,2,2,3,3,4,4,5,5,6,6…
polarisations(nchannels,3)	1,1,1…1,1,1,1,…,1	2,1,2,2,2,1,2,2	2,2,2,2,2,2,2,2,2,2
lprofiles(nfrequencies)	1,1,1,…,1,2,2,2…,2	1,1,1,1,2,2,2,2	1,1,1,1,1,2,2,2,2,2

Note the SSM/I numbers assume an RTTOV-8 not an RTTOV-7 coeff file.

The **coef** array contains the RT coefficients read from the coefficient file. The **addcloud** parameter is a logical switch which is normally set to false unless the user desires some of the additional downwelling cloudy radiance streams to be computed which increases computation time. This option is needed for RTTOV_CLD.

5.3. Output arrays from RTTOV_8_7

```
Call rttov_direct(errorstatus,nfrequencies,nchannels,nbtout,nprofiles,
channels,polarisations,lprofiles,profiles,coef,addcloud,calcemiss,
emissivity,transmission,radiancedata)
```

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 D) 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)

Annex L defines fully the output radiance, emissivity and transmittance type structures. Table 9 defines in more detail which arrays are used for output by users and their dimensions for **rttov_direct** and gradient routines.

Table 9 RTTOV_8_7 output arrays

Radiance_Type Radiances in *mw/cm-1/ster/sq.m*		
Type	**Array name**	**Contents**
real	clear_out(nbtout)	Clear sky top of atmosphere radiance output for each channel
real	total_out(nbtout)	Clear+cloudy top of atmosphere radiance for given cloud top pressure and fraction for each channel
real	upclear(nchannels)	clear sky radiance without reflection term

real	dnclear(nchannels)	clear sky downwelling radiance
real	reflclear(nchannels)	clear sky reflected downwelling radiance
real	overcast(nlev,nchannels)	overcast radiance at given cloud top and above (levels,channels)
real	downcld(nlev,nchannels)	contribution to radiance of downward cloud emission at given cloud top (levels,channels)
Radiance_Type Brightness Temperatures *degK*		
real	out(nbtout)	BT equivalent to total (clear+cloudy) top of atmosphere radiance output for each channel
real	out_clear(nbtout)	BT equivalent to clear top of atmosphere radiance output for each channel
Transmission_Type Transmittances 0-1		
real	tau_surf(nfrequencies)	transmittance from surface for each channel
real	tau_layer(nlevels,nfrequencies)	Transmittance from each standard pressure level to top of atmosphere for each channel
real	od_singlelayer(nlevels,nfrequencies)	single-layer optical depth for each level and channel
Emissivity 0-1		
real	emissivity(nchannels)	Input surface emissivity values for *calcemis=.false*. Output emissivity vales for *calcemis=.true.* Note for microwave channels these are emissivity values for each polarisation not a combined value as in RTTOV-7

5.4. Running RTTOV_8_7

You need to ensure the following in your program which calls RTTOV_8_7.

- 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 G for a detailed description of the variables required for input to RTTOV_8_7 and Annex M for example code.
- Initialise the variables, these are defined in the **rttov_types** module and listed in Annex L. Be careful that units for gases water vapour and ozone are now 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 cloudy calculations are to be performed by use of the logical flag *addcloud* which should be set to false unless cloudy 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.
- Ensure the variables *profiles(1) % ozone_data*, *profiles(1) % clw_data* and *profiles(1) % co2_data* are both set either 'true' or 'false' depending on whether you want to provide a concentration profile for each constituent or not respectively.
- *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 may free memory by de-allocating the coefficient structure with the **rttov_dealloc_coef** routine.
- 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.
- The **rttov_cld** and **rttov_scatt** routines are a level up from **rttov_direct** but they have almost the same calling structure and arrays to fill. Again the test programs supplied **rttovcld_test** and **rttovscatt_test** can be used as examples and similar rules apply to calling **rttov_direct**. Note however the cloud/hydrometeor profile arrays are input on user defined model levels not on the coefficient file levels and an interpolation is provided to the internal

RTTOV_8_7 levels. Remember to compile using the *make cloud* or *make scat* options. To invoke the various cloud physics options in Table 5 the values for kice and kradip must be set in *cld_profiles(j) % kice* and *cld_profiles(j) % kradip)* as defined in Annex L.

6. Known bugs for RTTOV_8_7

There are several known bugs in the version ***rttov_8_7*** of the code which are listed below. Corrections to these will be provided via the RTTOV-8 web page http://www.metoffice.gov.uk/research/interproj/nwpsaf/rtm/rtm_rttov8.html as they become available. They are:

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

7. Reporting bugs to the NWP SAF

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. 6, 7, 71 or 8_5 or 8_7)
- Platform and operating system you are running the code on (e.g. HP, UNIX)
- Compiler used (e.g. HP FORTRAN-90)
- Classification of report as: serious, cosmetic or improvement
- Copy of file rttov_const.f90 (for RTTOV_8_7) or mod_cparam.f90 (for RTTOV-7)
- 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 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.

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 to RTTOV_8_7? *If you want any of the following the answer is yes:*
   - *Cleaner F90 code for integration in your applications*
   - *Simulations for IASI or WINDSAT*
   - *Better IR cloudy radiance simulations*
   - *Improved microwave window channel radiances over the ocean*
   - *Possibilities to upgrade to newer spectroscopy and hence better accuracy with new coeff files available soon*
   - *Ability to change number of levels with same executable (i.e. driven by coeff file)*
   - *Slightly faster execution times, much faster for SSM/I or TMI*
   - *Much reduced use of memory for AIRS simulations*
   - *Ability to run scattering calculations for microwave radiances to simulate precipitation effects*

2. Can I compile the code in single precision? *Yes for RTTOV_8_7 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 or CLW profile to include in the state vector. What can I do? *You should set the logical flags* ***ozone_data*** *or* ***co2_data*** *or* ***clw_data*** *to false and the code will assign a reference profile.*

4. Why do the numbers in the ***tstrad_full.ksh*** output (see Table 6) 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_8_7, how do I best extrapolate it for RTTOV? *For water vapour it is best to use the reference profile which is available by the coef structure coef%ref_prfl_mr(:,coef%fmv_gas_pos(gas_id_watervapour)) . For temperature one can extrapolate from the top level of the model.*

6. Can I keep the same interface in my application as for RTTOV-7? *Yes see Annex K but note that not all the new capabilities are available*

7. More to be added here……

Good Luck and please provide me with any feedback on your experiences. Remember do not pass this code on to anyone else without the permission of EUMETSAT. The code is provided to you on an "as is" basis and there is no commitment to maintain it.

9. References

Boudala, F.S., Isaac, G.A., Fu, Q. & 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

Chevallier, F., P. Bauer, G. A. Kelly, C. Jakob, and T. McNally, 2001 Model clouds over oceans as seen from space: comparison with HIRS/2 and MSU radiances. J. Climate 14 4216-4229.

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

English S.J. and T.J. Hewison 1998 A fast generic millimetre wave emissivity model. Microwave Remote Sensing of the Atmosphere and Environment Proc. SPIE 3503 22-30

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).

Garand, L., Turner, D.S., Larocque, M., Bates, J., Boukabara, S., Brunel, P., Chevallier, F., Deblonde, G., Engelen, R., Hollingshead, M., Jackson, D., Jedlovec, G., Joiner, J., Kleespies, T., McKague, D.S., McMillin, L., Moncet, J. L., Pardo, J. R., Rayer, P. J., Salathe, E., Saunders, R., Scott, N. A., Van Delst, P., Woolf, R. 2001 Radiance and Jacobian intercomparison of radiative transfer models applied to HIRS and AMSU channels. J. Geophys. Res., 106, D20, 24,017

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 transfer model for the infrared atmospheric sounding interferometer. ECMWF Research Dept. Tech. Memo. 425. 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

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.

Saunders R.W., M. Matricardi and P. Brunel 1999 An Improved Fast Radiative Transfer Model for Assimilation of Satellite Radiance Observations. QJRMS, 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 cice clouds. *J. Clim.,* **11,** 1793-1802.

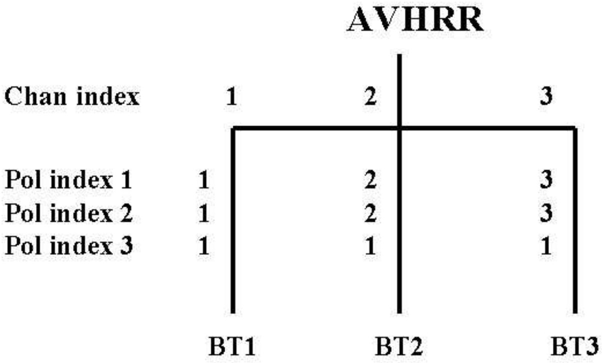

Figure 1. AVHRR channel definitions nfrequency=3, nchannels=3, nbtout=3

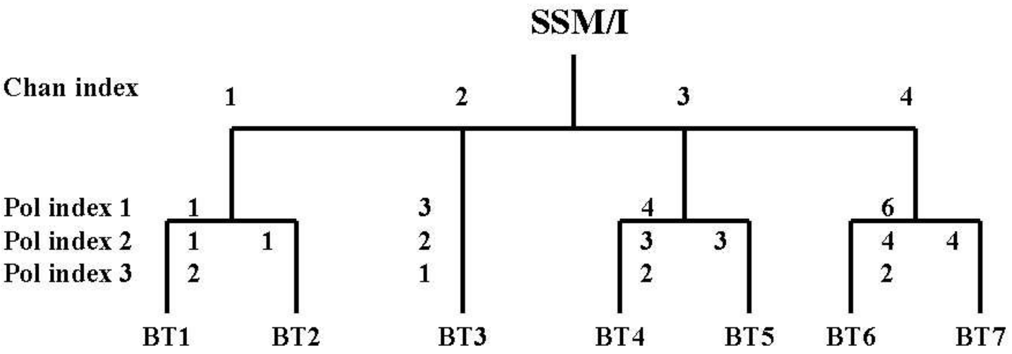

Figure 2 SSM/I channel definitions nfrequency=4, nchannels=7,nbtout=7 (rttov-8 coef file)

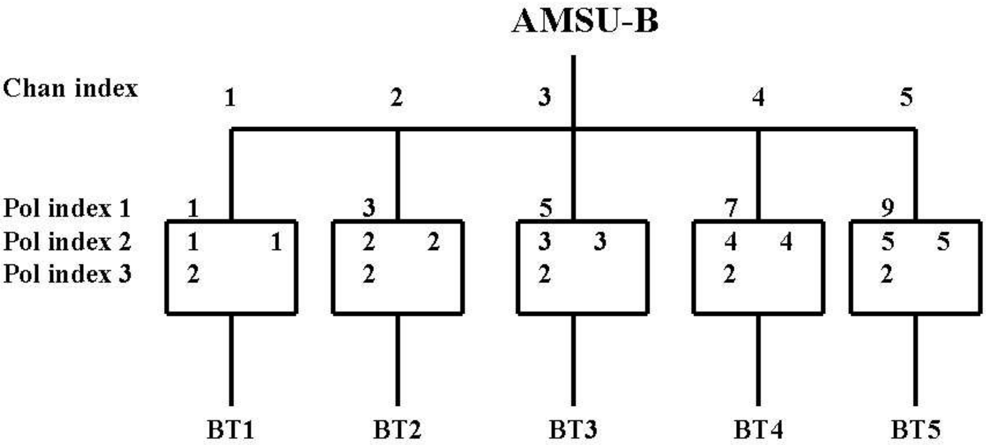

Figure 3 AMSU-B channel definitions nfrequency=5, nchannels=10, nbtout=5

Annex A: RTTOV_ERRORHANDLING interface

Call **rttov_errorhandling** (err_unit, verbosity_level)

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.

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

Annex B: RTTOV_READCOEFFS interface

```
Call rttov_readcoeffs  (    &
            & errorstatus,    & ! out
            & coef,           & ! out
            & instrument,     & ! in Optional
            & kmyproc,        & ! in Optional
            & kioproc,        & ! in Optional
            & file_id,        & ! in Optional
            & channels        ) ! in Optional
```

rttov_readcoeff is called once per rttov identification (platform, satellite, instrument). The routine is able to create the filename of the coefficient if *instrument(3)* is provided. If *file_id* is present the coefficient file is opened with this number as logical unit, if not, the first free logical unit (>9) is used. Note that *instrument* or *file_id* must be present. If *channels* is present, only the corresponding list of channels is extracted from the coefficient file

Subroutine Arguments:

Type	In/Out	Variable	Description
Integer Optional	Intent (in)	instrument(3)	(platform id, satellite id, instrument id)
Integer Optional	Intent (in)	file_id	file logical unit number
Integer Optional	Intent (in)	channels(:)	list of channels to extract
Integer Optional	Intent (in)	kmyproc	Logical processor id
Integer Optional	Intent (in)	kioproc	Processors dedicated to i/o
Integer	Intent (out)	errorstatus	return code
Type(rttov_coef)	Intent (out)	coef	coefficients

Optional Arguments:

Argument	Action if present	Action if not present
Instrument	Filename = rtcoef_*platform_satellite_instrument*.bin or extension “.dat” for ASCII	File should be opened by user. File_id must be present
File_id	Open(unit=file_id)	File_id = first free logical unit
Channels	Read only selected channels	Read all channels
kmyproc	Assumes logical processor id is kmyproc	Assumes kmyproc=1
kioproc	Assumes number of i/o processors is kioproc	Assumes kioproc=1

The optional arguments instrument and file_id determines whether the file is already opened or not
If "instrument" is present the routine will try to open the corresponding binary file (extension .bin) in read only mode. If it fails then it tries to open the ASCII file (extension .dat). File is closed before return.

If "instrument" is not present but file_id is present the routine will access to the coefficient file already opened with the logical unit file_id.

The ASCII/binary test is performed by reading the first characters, binary files will always start by %RTTOV_COEFF characters. An ASCII file cannot contain such a string at the beginning of the file because it will be considered as a section name which will not be recognised.

The optional variables kmyproc and kioproc relate to processing on a MPP machine. For users on a single processor platform these variables should not be provided.

Annex C: RTTOV_INITCOEFFS interface

```
Call rttov_initcoeffs (   &
       & errorstatus,     & ! out
       & coef,            & ! out
       & knproc,          & ! in Optional
       & kmyproc,         & ! in Optional
       & kioproc          ) ! in Optional
```

rttov_initcoeffs is called once per rttov identification (platform, satellite, instrument) immediately after **rttov_readcoeffs**.

Subroutine Arguments:

Type	In/Out	Variable	Description
Integer Optional	Intent (in)	knproc	Number of processors
Integer Optional	Intent (in)	kmyproc	Logical processor
Integer Optional	Intent (in)	kioproc	Number of processors dedicated to i/o
Integer	Intent (out)	errorstatus	return code
Type(rttov_coef)	Intent (out)	coef	coefficients

Optional Arguments:

Argument	Action if present	Action if not present
knproc	Assumes number of processors is knproc	Assumes knproc=1
kmyproc	Assumes logical processor id is kmyproc	Assumes kmyproc=1
kioproc	Assumes number of i/o processors is kioproc	Assumes kioproc=1

The optional variables knproc, kmyproc and kioproc relate to processing on a MPP machine. For users on a single processor platform these variables should not be provided.

Annex D: RTTOV_SETUP interface

```
Call rttov_setup  (          &
          & errorstatus,      & ! out
          & err_unit,         & ! in
          & verbosity_level,& ! in
          & ninst,            & ! in
          & coef,             & ! 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
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

Annex E: RTTOV_SETUPCHAN

Call **rttov_setupchan** (nprofiles, nchan, coef, nfrequencies, nchannels, nbtout)

rttov_setupchan is called for every instrument to compute the default values for *nfrequencies, nchannels and nbtout.*

Subroutine arguments:

Type	In/Out	Variable	Description	Example for AMSU-A
Integer	Intent (in)	nprofiles	Number of profiles per RTTOV call	3
Integer	Intent (in)	nchan(nprofiles)	Number of channels required for each profile	15,15,15
Integer	Intent (in)	coef	Coefficient file structure	N/A
Integer	Intent (out)	nfrequencies	Number of channels* nprofiles	45
Integer	Intent (out)	nchannels	Number of radiance streams required	90
Integer	Intent (out)	nbtout	Number of output BTs or rads per call.	45

Annex F: RTTOV_SETUPINDEX

Call subroutine **rttov_setupindex** (nchan, nprofiles, nfrequencies, nchannels, nbtout, coef, surfem, lprofiles, channels, polarisations, emissivity)

rttov_setupindex is called for every instrument to fill the arrays for *lprofiles, channels, polarisations and emissivity*. This routine assumes ALL channels for which coefficients are read in will be computed.
Subroutine arguments:

Type	In/Out	Variable	Description	Example for SSM/I
Integer	Intent (in)	nchan(nprofiles)	Number of channels/profile required	4,4
Integer	Intent (in)	nprofiles	Number of profiles required	2
Integer	Intent (in)	nfrequencies	Number of channels* nprofiles	8
Integer	Intent (in)	nchannels	Number of radiance streams required	14
Integer	Intent (in)	nbtout	Number of output BTs or rads per call.	14
Real	Intent (in)	coef	Coefficient structure	N/A
Real	Intent (in)	surfem(nprofiles,nchan)	Input surface emissivities	0.5,0.5,0.5,0.5, 0.5,0.5,0.5,0.5
Integer	Intent (out)	lprofiles(nfrequencies)	Profile indices	1,1,1,1,2,2,2,2
Integer	Intent (out)	channels(nfrequencies)	Channel indices	1,2,3,4,1,2,3,4
Integer	Intent (out)	polarisations(nchannels,3)	Polarisation indices	(see Table 7)
Real	Intent (out)	emissivity(nchannels)	Emissivities for input to rttov	0.5,0.5,0.5,0.5 ……

Annex G: RTTOV interface

Call **rttov_direct**(errorstatus, nfrequencies, nchannels, nbtout, nprofiles, channels, polarisations, lprofiles, profiles, & coef, addcloud, calcemiss, emissivity, transmission, radiancedata)

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)	nfrequencies	Number of channels used * profiles (=nfreq * profiles)	38
Integer	Intent(in)	nchannels	Number of radiance streams computed internally (= nfreq * npol * profiles) npol is required polarisation/channel.	38
Integer	Intent(in)	nbtout	Number of output BTs or radiances returned per call	38
Integer	Intent(in)	nprofiles	Number of profiles	2
Integer	Intent(in)	channels(nfrequencies)	Channel indices	1,2,3,…18,19
Integer	Intent(in)	polarisations(nchannels,3)	Polarisation indices	see Table 8
Integer	Intent(in)	lprofiles(nfrequencies)	Profiles indices	1,1,1,1 …,2,2,2
Type(profile_Type)	Intent(in)	profiles(nprofiles)	Profiles	N/A
Logical	Intent(in)	addcloud	switch for cloud computations	True/False
Type(rttov_coef)	Intent(in)	coef	Coefficients	N/A
Logical	Intent(in)	calcemiss(nchannels)	switch for emissivity calc.	True/False
Real	Intent(inout)	emissivity(nchannels)	surface emissivity	0.98,0.98..
Type(radiance_Type)	Intent(inout)	radiancedata	radiances (mw/cm-1/ster/sq.m) & degK	See Table 9
Type(transmission_Type	Intent(out)	transmission	Transmittances (0-1.)	N/A

nfreq is the number of separate frequencies per profile at which a sensor measures radiance e.g. for SSM/I this is 4 (but 7 for RTTOV-7 coeff files) for AMSU-A it is 15 and for HIRS 19. *nfrequencies* is the number of separate frequencies for all profiles input per RTTOV call. Inside RTTOV_8_7 separate radiance calculations can be performed for each polarisation at a single frequency resulting in *nfrequencies*npol = nchannels* radiance streams in the code. *npol* can vary with channel. These radiance streams can be combined on output (e.g. for AMSU) to provide a single brightness temperature/radiance or not combined to give separate brightness temp/radiance for each polarisation and frequency (e.g. for SSM/I). The *nbtout* parameter defines the number of computed brightness temperatures/radiances output from RTTOV_8_7. The arrays *channels*, *polarisations* and *lprofiles* contain the corresponding channel, polarisation and profile indices for each computed radiance. Figures 1, 2, 3 illustrate this structure for AVHRR, SSM/I and AMSU-B. All infrared sensors will be like AVHRR.

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 1, 2 or 3. The version of the model inside the coefficient file defines the version of the emissivity algorithm (see Table 4).

addcloud is a flag that allows the user to compute additional radiances for further cloudy calculations. The flag is copied to the radiance structure (radiance%lcloud) and the array radiance%downcld is calculated and should be allocated by the user.

Annex H: RTTOV K interface

Subroutine **rttov_k**(errorstatus, nfrequencies, nchannels, nbtout, nprofiles, channels, polarisations, lprofiles, profiles, coef, addcloud, 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 nbtout, the array channels, polarisations and lprofiles contains the corresponding channel, polarisation and profile number for each computed radiance.

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus(nprofiles)	return flag
Integer	Intent(in)	nfrequencies	Number of channels used * profiles (=nfreq * profiles)
Integer	Intent(in)	nchannels	Number of radiance streams computed internally (= nfreq * npol * profiles) npol is required polarisation/channel.
Integer	Intent(in)	nbtout	Number of output BTs or radiances returned per call
Integer	Intent(in)	nprofiles	Number of profiles
Integer	Intent(in)	channels(nfrequencies)	Channel indices (see Table 8)
Integer	Intent(in)	polarisations(nchannels,3)	Polarisation indices (see Table 8)
Integer	Intent(in)	lprofiles(nfrequencies)	Profiles indices (see Table 8)
Type(profile_type)	Intent(in)	profiles(nprofiles)	Profiles
Type(rttov_coef)	Intent(in)	coef	Coefficients
Logical	Intent(in)	addcloud	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
Type(profile_type)	Intent(inout)	profiles_k(nchannels)	K matrix on profile variables
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)	radiancedata	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 the 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.

State variables considered in the K code
Profile:

All profile structure elements are considered as state variables except:

profile%skin%surftype	constant value
profile%s2m%o	never used in direct model
profile%ozone_data	constant value; if false then no K for profile%ozone
profile%co2_data	constant value; if false then no K for profile%co2
profile%clw_data	constant value; if false or not MW then no K for profile%clw
profile%zenangle	constant value
profile%azangle	constant value

Annex I: RTTOV TL interface

Subroutine **rttov_tl**(errorstatus, nfrequencies, nchannels, nbtout, nprofiles, channels, polarisations, lprofiles, profiles, coef, addcloud, calcemiss, emissivity, profiles_tl, 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 nbtout, the array channels, polarisations and lprofiles contain the corresponding channel, polarisation and profile indices for each computed radiance.

Type	In/Out	Variable	Description
Integer	Intent(out)	errorstatus(nprofiles)	return flag
Integer	Intent(in)	nfrequencies	Number of channels used * profiles (=nfreq * profiles)
Integer	Intent(in)	nchannels	Number of radiance streams computed internally (= nfreq * npol * profiles) npol is required polarisation/channel.
Integer	Intent(in)	nbtout	Number of output BTs or radiances returned per call
Integer	Intent(in)	nprofiles	Number of profiles
Integer	Intent(in)	channels(nfrequencies)	Channel indices (see Table 8)
Integer	Intent(in)	polarisations(nchannels,3)	Polarisation indices (see Table 8)
Integer	Intent(in)	lprofiles(nfrequencies)	Profiles indices (see Table 8)
Type(profile_type)	Intent(in)	profiles(nprofiles)	Profiles
Type(rttov_coef)	Intent(in)	coef	Coefficients
Logical	Intent(in)	addcloud	switch for cloud computations
Logical	Intent(in)	calcemiss(nchannels)	switch for emissivity calc.
Real	Intent(inout)	emissivity(nchannels)	surface emissivity
Type(radiance_type)	Intent(inout)	radiancedata	Forward model output radiances (mw/cm-1/ster/sq.m) & degK
Type(profile_type)	Intent(in)	profiles_tl(nprofiles)	Input profile variable increments
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

State variables considered in the TL code
Profile:

All profile structure elements are considered as state variables except:

• profile%skin%surftype	constant value
• profile%s2m%o	never used in direct model
• profile%ozone_data	constant value; if false then no TL for profile%ozone
• profile%co2_data	constant value; if false then no TL for profile%co2
• profile%clw_data	constant value; if false or not MW then no TL for profile%clw
• profile%zenangle	constant value
• profile%azangle	constant value

Annex J: RTTOV AD interface

Subroutine **rttov_ad**(errorstatus, nfrequencies, nchannels, nbtout, nprofiles, channels, polarisations, lprofiles, profiles, coef, addcloud, switchrad, calcemiss, emissivity, profiles_ad, 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)	nfrequencies	Number of channels used * profiles (=nfreq * profiles)
Integer	Intent(in)	nchannels	Number of radiance streams computed internally (= nfreq * npol * profiles) npol is required polarisation/channel.
Integer	Intent(in)	nbtout	Number of output BTs or radiances returned per call
Integer	Intent(in)	nprofiles	Number of profiles
Integer	Intent(in)	channels(nfrequencies)	Channel indices (see Table 8)
Integer	Intent(in)	polarisations(nchannels,3)	Polarisation indices (see Table 8)
Integer	Intent(in)	lprofiles(nfrequencies)	Profiles indices (see Table 8)
Type(profile_type)	Intent(in)	profiles(nprofiles)	Profiles
Type(rttov_coef)	Intent(in)	coef	Coefficients
Logical	Intent(in)	addcloud	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
Type(profile_type)	Intent(inout)	profiles_ad(nchannels)	AD of profile variables
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	input perturbation radiance (or BT)

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.For cloudy calculations (flag addcloud set) take care of allocating and initialising (0) the overcast and downcld arrays.

State variables considered in the AD code

Profile:

All profile structure elements are considered as state variables except:

- profile%skin%surftype constant value
- profile%s2m%o never used in direct model
- profile%ozone_data constant value; if false then no AD for profile%ozone
- profile%co2_data constant value; if false then no AD for profile%co2
- profile%clw_data constant value; if false or not MW then no AD for profile%clw
- profile%zenangle constant value

Annex K: RTTOV-7 interface to RTTOV_8_7 code

For RTTOV-7 users who do not want to change their programs which call RTTOV but still want to benefit from the improvements, they may use the RTTOV-7 calls for the **direct model only** compiled using the new RTTOV_8_7 library. The RTTOV_8_7 library contains the emulation of the modules **mod_cparam** and the RTTOV-7 subroutines **rttvi**, **rttov**. The interfaces are strictly unchanged from the RTTOV-7 code.

This emulation is only valid for coefficient files on the 43 RTTOV-7 fixed pressure levels with FASTEM-1/2. It also only assumes defaults for the polarisation arrays and coefficient files to be consistent with RTTOV-7 (i.e. SSM/I has 7 channels not 4). The user still has to modify the **mod_cparam** module for the maximum number of channels, profiles, satellites he wants to process as for the RTTOV-7 code. Note the unit for concentration of gases is mass mixing ratio (kg/kg) for RTTOV-7.

Please refer to the RTTOV-7 documentation for interface details.

Annex L: 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.

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. The structures are defined in the following pages.

Type	Variable	Description
Surface skin		
Type skin_type		
Integer	surftype	0=land, 1=sea, 2=sea-ice
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 wind component (m/s)
Real	v	V wind component (m/s)
Atmospheric Profile		
Type profile_type		
Integer	nlevels	number of atmospheric levels
logical	ozone_data	ozone profiles available
logical	co2_data	carbon dioxide profiles available
logical	clw_data	cloud liquid water profiles available
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)
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)
Black body cloud		
Real	ctp	cloud top pressure (hPa)
Real	cfraction	cloud fraction (0 - 1) 1 for 100% cloud cover

structure for atmospheric profiles with information on clouds for each level		
Type profile_cloud_type		
Integer	nlevels	number of atmospheric levels
atmosphere defined on nlevels (nlevels+1 for ph)		
Real	p(:)	full-level model pressure (hPa)
Real	ph(:)	half-level model pressure (hPa)
Real	t(:)	temperature (K)
Real	q(:)	specific humidity kg/kg
Real	cc(:)	cloud cover
Real	clw(:)	cloud liquid water (kg/kg)
Real	ciw(:)	cloud ice water (kg/kg)
Real	rain(:)	rain (kg/m^2)
Real	sp(:)	solid precipitation (kg/m^2)
Integer	kice	Ice crystal type (0=hex columns, 1=aggregates)
Integer	kradip	Ice effective size scheme (0=Ou-Liou, 1=Wyser, 2=Boudala, 3=McFarquhar)
Structure for one RTTOV coefficient set		
Type rttov_coef		
Integer	id_platform	platform
Integer	id_sat	satellite (.....)
Integer	id_inst	instrument (.....)
Integer	id_sensor	sensor 1 = Infrared 2 = Micro Wave 3 = High resolution
Integer	id_comp_lvl	RTTOV version compatibility level
Integer ,Dimension(3)	id_creation_date	YYYY MM DD
Character (len=80)	id_creation	Creation comment
Character (len=32)	id_Common_name	usual name of the satellite
FAST_MODEL_VARIABLES section		
Character (len=32)	fmv_model_def	FMV definition (RTTOV6, RTTOV7, RTTOV8)
Integer	fmv_model_ver	fast model version compatibility level
Integer	fmv_chn	number of channels in file
Integer	fmv_gas	number of gases in file
integer	fmv_gas_id(:)	gas id. number i gas_id list (fmv_gas)
Integer	fmv_gas_pos(:)	respective position of each gas of gas_id list (fmv_gas)
Integer	fmv_var(:)	number of variables/predictors by gaz (fmv_gas)
Integer	fmv_lvl(:)	number of levels(pres/absorber) by gaz (fmv_gas)
Integer	nmixed	number of variables/predictors for Mixed Gases
Integer	nwater	number of variables/predictors for Water Vapour
Integer	nozone	number of variables/predictors for

		Ozone
Integer	nwvcont	number of variables/predictors for WV continuum
Integer	nco2	number of variables/predictors for CO2
Integer	nn2o	number of variables/predictors for N2O
Integer	nco	number of variables/predictors for CO
Integer	nch4	number of variables/predictors for CH4
Integer	nlevels	number of levels(pres/absorber) same for all gases
GAZ_UNITS section array size is fmv_chn		
Integer	gaz_units(:)	unit for gaz concentration for each gaz. Specific concentration is 1 (kg/kg), volume mixing ratio is 2 (ppmv) (see rttov_const for definition of integer values)
FILTER_FUNCTIONS section array size is fmv_chn		
Integer	ff_ori_chn(:)	original chan number
Integer	ff_val_chn(:)	validity of the channel (1=OK)
Real	ff_cwn (:)	central wave number (cm-1)
Real	ff_bco (:)	band correction offset (K)
Real	ff_bcs (:)	band correction slope (K/K)
Real	ff_gam (:)	gamma factor transm. correction
FUNDAMENTAL_CONSTANTS section		
Real	fc_speedl	speed of light (cm/s)
Real	fc_planck_c1	first radiation constant (mW/(m2*sr*cm-4))
Real	fc_planck_c2	second radiation constant (cm*K)
Real	fc_sat_height	satellite nominal altitude (km)
FASTEM section		
Integer	fastem_ver	fastem version number
Integer	fastem_coef_nb	number of coefficients
Real	fastem_coef(:)	coefficients (fastem_coef_nb)
Integer	fastem_polar(:)	polarisation of each channel 0 = 0.5 V+H 1 = 90 - incident angle 2 = incident angle 3 = vertical 4 = horizontal 5 = vertical and horizontal 6 = full stokes vector
SSIREM section array size is fmv_chn		
Integer	ssirem_ver	version number
Integer	ssirem_chn(:)	original chan number
Real	ssirem_a0(:)	constant coef
Real	ssirem_a1(:)	first coef
Real	ssirem_a2(:)	second coef
Real	ssirem_xzn1(:)	1st exponent on zenith angle
Real	ssirem_xzn2(:)	2nd exponent on zenith angle
REFERENCE_PROFILE section defined on Mixed gases pressure levels		

Real	ref_prfl_p(:)	pressure (hPa) (levels)
Real	ref_prfl_t(:,:)	temperature (K) (levels, gases)
Real	ref_prfl_mr(:,:)	mixing ratio (ppmv) (levels, gases)
PROFILE_LIMITS section		
Real	lim_prfl_p(:)	pressure (hPa) (levels)
Real	lim_prfl_tmax(:)	max temperature (K) (levels)
Real	lim_prfl_tmin(:)	min temperature (K) (levels)
Real	lim_prfl_gmax(:,:)	max mixing r (ppmv) (levels, gases)
Real	lim_prfl_gmin(:,:)	min mixing r (ppmv) (levels, gases)
FAST_COEFFICIENTS section		
Real	mixedgas(:,:,:)	Mixed gases coefs (levels, channels, variables)
Real	watervapour(:,:,:)	Water vapour coefs (levels, channels, variables)
Real	ozone(:,:,:)	Ozone coefs (levels, channels, variables)
Real	wvcont(:,:,:)	WV Cont coefs (levels, channels, variables)
Real	co2(:,:,:)	CO2 coefs (levels, channels, variables)
Real	n2o(:,:,:)	N2O coefs (levels, channels, variables)
Real	co(:,:,:)	CO coefs (levels, channels, variables)
Real	ch4(:,:,:)	CH4 coefs (levels, channels, variables)
Auxillary variables		
real	ratoe	ratio (H+R)/R H=sat height, R=Earth radius
real	planck1(:)	C1 * Nu**3
real	planck2(:)	C2 * Nu
real	frequency_ghz(:)	frequency in GHz
Variables for predictors see Science and Validation report		
real	dp(:)	interval between standard p levels (hPa)
real	dpp(:)	pressure based variable (hPa**2)
real	tstar(:)	layer temp (K)
real	to3star(:)	layer temp for O3 calculations (K)
real	wstar(:)	layer WV (ppmv)
real	ostar(:)	layer O3 (ppmv)
real	co2star(:)	layer CO2 (ppmv)
Radiance and corresponding brightness temperature **type radiance_type** *Array size is of size nchannels except for cloudy calculations (nlevels, nchannels)* *unit for radiance is mw/cm-1/ster/sq.m* *unit for temperature is Kelvin*		
logical	lcloud	if true the last array is calculated if false it does not need to be

		allocated
real	clear(:)	internal clear sky radiance
real	clear_out(:)	clear sky radiance output
real	cloudy(:)	100% cloudy radiance for given cloud
real	total(:)	internal cloudy radiance for given cloud
real	total_out(:)	cloudy radiance for given cloud output
real	out(:)	BT equivalent to total radiance output
real	out_clear(:)	BT equivalent to clear radiance output
real	bt(:)	internal BT equivalent to total radiance
real	bt_clear(:)	internal BT equivalent to clear radiance
real	upclear(:)	clear sky radiance without reflection term
real	dnclear(:)	clear sky downwelling radiance
real	reflclear(:)	reflected clear sky downwelling radiance
real	overcast(:,:)	overcast radiance at given cloud top (levels,channels)
real	downcld(:,:)	contribution to radiance of downward cloud emission at given cloud top (levels,channels)
Emissivity and radiance arrays for cloudy conditions **type radiance_cloud_type** *see rttov_cld* *Array size is of size nchannels* *except for cloudy calculations (nlevels, nchannels)* *First part, same definition as the radiance type*		
logical	lcloud	if true the last array is calculated if false is not allocated
real	clear(:)	internal clear sky radiance
real	clear_out(:)	clear sky radiance output
real	cloudy(:)	100% cloudy radiance for given cloud
real	total(:)	internal cloudy radiance for given cloud
real	total_out(:)	cloudy radiance for given cloud output
real	out(:)	BT equivalent to total radiance output
real	out_clear(:)	BT equivalent to clear radiance output
real	bt(:)	Brightness temp equivalent to total radiance
real	bt_clear(:)	Brightness temp equivalent to clear radiance
real	upclear(:)	clear sky radiance without reflection term

real	dnclear(:)	clear sky downwelling radiance
real	reflclear(:)	reflected clear sky downwelling radiance
real	overcast(:,:)	overcast radiance at given cloud top (levels,channels)
real	downcld(:,:)	contribution to radiance of downward cloud emission at given cloud top (levels,channels)
Second part Cloud specific		
real	cldemis(:,:)	cloud emissivity (levels, channels)
real	wtoa(:,:)	toa weights of of cloud layers
real	wsurf(:,:)	surface weights of cloud layers
real	cs_wtoa(:)	contribution from clear sky fraction
real	cs_wsurf(:)	contribution from clear sky fraction
Third part scatt specific **type rttov_scatt_coef** *see rttov_scatt* *Array sizes for Mie tables are defined by mfreq, mtype, mtemp, mwc in that order*		
integer	nhydro	Number of hydrometeors in computation
integer	mtype	Number of hydrometeors in Mie tables
integer	mfreqm	Number of frequencies in Mie tables
integer	mtemp	Number of temperature bins in Mie tables
integer	mwc	Number of water bins in Mie tables
real	offset_temp_rain	temperature offset in table for rain type
real	offset_temp_sp	temperature offset in table for solid prec. type
real	offset_temp_liq	temperature offset in table for cloud water type
real	offset_temp_ice	temperature offset in table for cloud ice type
real	offset_temp_water	liquid/ice water offset in table
real	scale_water	log10(liquid/ice water) scaling factor in table
real	from_scale_water	coefficients for rain unit conversion (mm.h-1 to g.m-3)
real	conv_rain(2)	coefficients for solid prec. unit conversion (mm.h-1 to g.m-3)
real	conv_sp (2)	coefficients for cld water conversion (not used)
real	conv_liq (2)	coefficients for cloud ice conversion (not used)
real	conv_ice (2)	coefficients for cloud ice conversion (not used)
real	mie_freq(:,:,:,:)	list of frequencies in Mie table
real	ext(:,:,:,:)	extinction coefficent table
real	ssa(:,:,:,:)	single scattering albedo table
real	asp(:,:,:,:)	assymetry parameter table
transmissions and optical depths **type transmission_type**		

! Transmissions and optical depths (unitless)		
real	tau_surf(:)	transmittance from surface (array size is nchannels)
real	tau_layer(:,:)	transmittance from each standard pressure level
real	od_singlelayer(:,:)	single-layer optical depth

Annex M: Example of user interface program to run RTTOV_8_7

```
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 2004, EUMETSAT, All Rights Reserved.
  !
  !     ***************************************************************
  !
  !     TEST PROGRAM FOR RTTOV SUITE FORWARD MODEL ONLY
  !          RTTOV VERSION 8_7
  ! 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 unix scripts available to set up the files above and
  ! run this program (e.g. tstrad_full.scr)
  ! 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
  ! 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
  Use rttov_types, Only : &
       rttov_coef     ,&
       profile_Type   ,&
       transmission_Type ,&
       radiance_Type

  Use parkind1, Only : jpim     ,jprb
  !
  Implicit None
  !
#include "rttov_direct.interface"
#include "rttov_readcoeffs.interface"
#include "rttov_initcoeffs.interface"
#include "rttov_v2q.interface"
#include "rttov_setupchan.interface"
#include "rttov_setupindex.interface"
#include "rttov_errorhandling.interface"
#include "rttov_dealloc_coef.interface"
#include "rttov_errorreport.interface"
  ! Commons
  !
  ! Functions


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

  ! One profile per run
  Integer (Kind=jpim) :: nprof = 1

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

  ! RTTOV_readcoeffs interface
  !====================
  Integer(Kind=jpim) :: errorstatus
  Integer(Kind=jpim) :: instrument(3)
  Type( rttov_coef ) :: coef         ! coefficients
  Integer(Kind=jpim), Allocatable :: lchan(:)

  ! RTTOV interface
  !====================
  Integer(Kind=jpim), Allocatable :: rttov_errorstatus(:)  ! rttov error return code
  Integer(Kind=jpim) :: nfrequencies
  Integer(Kind=jpim) :: nchannels
  Integer(Kind=jpim) :: nbtout
  Integer(Kind=jpim), Allocatable :: channels   (:)
  Integer(Kind=jpim), Allocatable :: polarisations   (:,:)
  Integer(Kind=jpim), Allocatable :: lprofiles  (:)
  Type(profile_Type)  :: profiles(1)! ONE profile but need array
  !Logical             :: addcloud = .True.
  Logical             :: addcloud = .False.
  Logical, Allocatable         :: calcemiss(:)
  Real(Kind=jprb), Allocatable :: emissivity (:)
  Type(transmission_Type)  :: transmission ! transmittances and layer optical depths
  Type(radiance_Type)      :: radiance

  Real(Kind=jprb),    Allocatable :: input_emissivity (:)
  Character (len=80) :: errMessage
  Character (len=6)  :: NameOfRoutine = 'tstrad'


  ! variables for input
  !===================
  ! Parameter for WV conversion used in all tstrad suite
  Real(Kind=jprb), Parameter :: q_mixratio_to_ppmv  = 1.60771704e+6_JPRB

  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), Allocatable    :: ems(:)
  Real(Kind=jprb)    :: zenith
  Real(Kind=jprb)    :: azimut
  Integer(Kind=jpim) :: ivch, ich
  Real(Kind=jprb)    :: ems_val
  Integer(Kind=jpim), Allocatable :: nchan(:)
  Integer(Kind=jpim) :: isurf


  ! 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(:,:)
  Real(Kind=jprb), Allocatable :: pr_upclr(:)
  Real(Kind=jprb), Allocatable :: pr_dncld(:,:)
  Real(Kind=jprb), Allocatable :: pr_refclr(:)
  Real(Kind=jprb), Allocatable :: pr_ovcst(:,:)

  ! loop variables
  Integer :: j, jpol
  Integer :: np
  Integer :: ilev, nprint
  Integer :: ios

  Integer(Kind=jpim) :: alloc_status(40)

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

  errorstatus     = 0
  alloc_status(:) = 0

  !===================================================
  !========== Interactive inputs == start ==============
  Write(0,*) 'enter platform number'
  Read(*,*) instrument(1)
  Write(0,*) 'enter satellite number '
  Read(*,*) instrument(2)
  Write(0,*) 'enter instrument number'
  Read(*,*) instrument(3)
  Write(0,*) 'enter surface type (0=land, 1=sea, 2=ice/snow)'
  Read(*,*) isurf
  Write(0,*) 'enter zenith angle in degrees'
  Read(*,*) zenith
```

```
Write(0,*) 'enter azimut angle in degrees'
Read(*,*) azimut
!
nchan = 0
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

!Pack channels and emmissivity arrays
Allocate(nchan(nprof))
Allocate(lchan(nchan(nprof))) ! Note these array sizes nchan can vary per profile
Allocate(ems(nchan(nprof)))      ! but for this example assume 1 profile/call with same channels
lchan(:) = input_chan(1:nchan(nprof))
ems(:)   = input_ems(1:nchan(nprof))
!
!========== 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)

!Read and initialise coefficients
!---------------------------------------------------------
Call rttov_readcoeffs (errorstatus, coef, instrument, channels = lchan(:))
If(errorstatus /= 0) Then
  Write(*,*) 'error rttov_readcoeffs :',errorstatus
  Stop "error rttov_readcoeffs"
Else
  Write(*,*) 'rttov_readcoeffs OK:'
Endif
Call rttov_initcoeffs (errorstatus,coef)
If(errorstatus /= 0) Then
  Write(*,*) 'error rttov_initcoeffs :',errorstatus
  Stop "error rttov_initcoeffs"
Else
  Write(*,*) 'rttov_initcoeffs OK:'
Endif

! security if input number of channels is higher than number
! stored in coeffs
If( nchan(nprof) > coef % fmv_chn ) Then
  nchan(nprof) = coef % fmv_chn
Endif

!Open output file
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 profile arrays with the number of levels.
! Take care that the number and pressure levels should be
! the same as the ones of the coefficient file.
profiles(1) % nlevels = coef % nlevels
Allocate(profiles(1) % p(coef % nlevels)  ,stat= alloc_status(1))
Allocate(profiles(1) % t(coef % nlevels)  ,stat= alloc_status(2))
Allocate(profiles(1) % q(coef % nlevels)  ,stat= alloc_status(3))
Allocate(profiles(1) % o3(coef % nlevels) ,stat= alloc_status(4))
Allocate(profiles(1) % clw(coef % nlevels),stat= alloc_status(5))
If( Any(alloc_status /= 0) ) Then
  errorstatus = errorstatus_fatal
  Write( errMessage, '( "mem allocation error for profile")' )
  Call Rttov_ErrorReport (errorstatus, errMessage, NameOfRoutine)
  Stop
End If

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

! we have an ozone profile
```

```
profiles(1) % ozone_Data =.True.
! we do not have CO2 profile
profiles(1) % co2_Data   =.False.
! check Cloud liquid water profile
profiles(1) % clw_Data   = profiles(1) % clw(1) >= 0.0_JPRB


! Other variables from interactive inputs
profiles(1) % skin % surftype  = isurf
profiles(1) % zenangle   = zenith
profiles(1) % azangle    = azimut

!========== Read profile == end ==============
!============================================


! Setup default number of frequencies, channels , output BTs
!   for the coeff file. These are then used by rttov_indexsetup
!   to set up channel and polarisation indices.
! Take care that this routine is only valid if
!   the user has selected a list of channels (channels = )
!   for the rttov_readcoeffs or rttov_setup routine
Call rttov_setupchan(nprof,nchan,coef,nfrequencies, &
     & nchannels,nbtout)

Allocate( rttov_errorstatus(1)           ,stat= alloc_status(1))
Allocate( channels ( nfrequencies )      ,stat= alloc_status(2))
Allocate( lprofiles  ( nfrequencies )    ,stat= alloc_status(3))
Allocate( emissivity ( nchannels )       ,stat= alloc_status(4))
Allocate( input_emissivity ( nchannels ) ,stat= alloc_status(5))
Allocate( calcemiss  ( nchannels )       ,stat= alloc_status(6))
Allocate( polarisations(nchannels,3)     ,stat= alloc_status(7))

! allocate transmittance structure
Allocate( transmission % tau_surf      ( nchannels )                ,stat= alloc_status(8))
Allocate( transmission % tau_layer     ( coef % nlevels, nchannels ) ,stat= alloc_status(9))
Allocate( transmission % od_singlelayer( coef % nlevels, nchannels  ),stat= alloc_status(10))

! allocate radiance results arrays with number of channels
Allocate( radiance % clear    ( nchannels ) ,stat= alloc_status(11))
Allocate( radiance % cloudy   ( nchannels ) ,stat= alloc_status(12))
Allocate( radiance % total    ( nchannels ) ,stat= alloc_status(13))
Allocate( radiance % bt       ( nchannels ) ,stat= alloc_status(14))
Allocate( radiance % bt_clear ( nchannels ) ,stat= alloc_status(15))
Allocate( radiance % upclear  ( nchannels ) ,stat= alloc_status(16))
Allocate( radiance % dnclear  ( nchannels ) ,stat= alloc_status(17))
Allocate( radiance % reflclear( nchannels ) ,stat= alloc_status(18))
Allocate( radiance % overcast ( coef % nlevels, nchannels ) ,stat= alloc_status(19))
! allocate the cloudy radiances with full size even
! if not used
Allocate( radiance % downcld  ( coef % nlevels, nchannels ) ,stat= alloc_status(20))

Allocate( radiance % out      ( nbtout ) ,stat= alloc_status(21))
Allocate( radiance % out_clear( nbtout ) ,stat= alloc_status(22))
Allocate( radiance % total_out( nbtout ) ,stat= alloc_status(23))
Allocate( radiance % clear_out( nbtout ) ,stat= alloc_status(24))

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



! Build the list of channels/profiles indices
!   outputs are lprofiles,channels,polarisations,emissivity
! Take care that this routine is only valid if
!   the user has selected a list of channels
!   for the rttov_readcoeffs or rttov_setup routine (channels = )
Call rttov_setupindex (nchan,nprof,nfrequencies,nchannels,nbtout,coef,&
     & ems,lprofiles,channels,polarisations,emissivity)


! save input values of emissivities for all calculations
! calculate emissivity where the input emissivity value is less than 0.01
input_emissivity(:) = emissivity(:)
calcemiss(:) = emissivity(:) < 0.01_JPRB

! Call RTTOV forward model
Call rttov_direct( &
     rttov_errorstatus, & ! out
     nfrequencies, & ! in
     nchannels,   & ! in
     nbtout,      & ! in
     nprof,       & ! in
     channels,    & ! in
     polarisations,& ! in
     lprofiles,   & ! in
     profiles,    & ! in
     coef,        & ! in
     addcloud,    & ! in
     calcemiss,   & ! in
     emissivity,  & ! inout
     transmission,& ! out
     radiance     ) ! inout

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(nbtout)  ,stat= alloc_status(1))
Allocate(pr_trans(nbtout)   ,stat= alloc_status(2))
Allocate(pr_emis(nbtout)    ,stat= alloc_status(3))
Allocate(pr_trans_lev(coef % nlevels,nbtout) ,stat= alloc_status(4))
Allocate(pr_upclr(nbtout)                  ,stat= alloc_status(5))
Allocate(pr_dncld(coef % nlevels,nbtout)    ,stat= alloc_status(6))
Allocate(pr_refclr(nbtout)                 ,stat= alloc_status(7))
Allocate(pr_ovcst(coef % nlevels,nbtout)    ,stat= alloc_status(8))
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
pr_upclr(:)   = 0.0_JPRB
pr_dncld(:,:) = 0.0_JPRB
pr_refclr(:)  = 0.0_JPRB
pr_ovcst(:,:) = 0.0_JPRB
!
Do j = 1 , nchannels
  jpol = polarisations(j,2)
  pr_radcld(jpol) = radiance % cloudy(j)
  pr_trans(jpol)  = Transmission % tau_surf(J)
  pr_emis(jpol)   = emissivity(j)
  pr_upclr(jpol)  = radiance % upclear(J)
  pr_refclr(jpol) = radiance % reflclear(J)
  Do ilev = 1 , coef % nlevels
    pr_trans_lev(ilev,jpol) = Transmission % tau_layer(ilev,J)
    pr_dncld(ilev,jpol)     = radiance % downcld(ILEV,J)
    pr_ovcst(ilev,jpol)     = radiance % overcast(ILEV,J)
  Enddo
Enddo



!    OUTPUT RESULTS
!
NPRINT = 1+ Int((nbtout-1)/10)
Write(IOOUT,*)' -----------------'
Write(IOOUT,*)' Instrument ', instrument(3)
Write(IOOUT,*)' -----------------'
Write(IOOUT,*)' '

Write(IOOUT,777)instrument(2), profiles(1)%zenangle,profiles(1)%azangle,profiles(1)%skin%surftype
Write(IOOUT,222) radiance % out(:)
Write(IOOUT,*)' '
Write(IOOUT,*)'CALCULATED RADIANCES: SAT =', instrument(2)
Write(IOOUT,222) radiance % total_out(:)
Write(IOOUT,*)' '
Write(IOOUT,*)'CALCULATED OVERCAST RADIANCES: SAT =', instrument(2)
Write(IOOUT,222) pr_radcld(:)
Write (IOOUT,*)' '
Write(IOOUT,*)'CALCULATED SURFACE TO SPACE TRANSMITTANCE: S'&
   &      ,'AT =',instrument(2)
Write(IOOUT,4444) pr_trans(:)
Write (IOOUT,*)' '
Write(IOOUT,*)'CALCULATED SURFACE EMISSIVITIES '&
   &      ,'SAT =',instrument(2)
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) (LCHAN(J),&
       &   J = 1+(NP-1)*10,Min(10+(NP-1)*10,nbtout))
    Do  ILEV = 1 , coef % NLEVELS
      Write(IOOUT,4445)ILEV,(pr_trans_lev(ilev,J),&
         &   J = 1+(NP-1)*10,Min(10+(NP-1)*10,nbtout))
    End Do
    Write(IOOUT,1115) (LCHAN(J),&
       &   J = 1+(NP-1)*10,Min(10+(NP-1)*10,nbtout))
  End Do
Endif
!
! deallocate model profiles atmospheric arrays
Deallocate( profiles(1) % p   ,stat=alloc_status(1))
Deallocate( profiles(1) % t   ,stat=alloc_status(2))
Deallocate( profiles(1) % q   ,stat=alloc_status(3))
Deallocate( profiles(1) % o3  ,stat=alloc_status(4))
Deallocate( profiles(1) % clw ,stat=alloc_status(5))
If( Any(alloc_status /= 0) ) Then
  errorstatus = errorstatus_fatal
  Write( errMessage, '( "mem deallocation error")' )
  Call Rttov_ErrorReport (errorstatus, errMessage, NameOfRoutine)
  Stop
End If

! number of channels per RTTOV call is only nchannels
Deallocate( channels   ,stat=alloc_status(2))
Deallocate( lprofiles  ,stat=alloc_status(3))
Deallocate( emissivity ,stat=alloc_status(4))
Deallocate( calcemiss  ,stat=alloc_status(5))

! allocate transmittance structure
Deallocate( transmission % tau_surf      ,stat= alloc_status(6))
Deallocate( transmission % tau_layer     ,stat= alloc_status(7))
Deallocate( transmission % od_singlelayer,stat= alloc_status(8))

! allocate radiance results arrays with number of channels
Deallocate( radiance % clear    ,stat=alloc_status(9))
Deallocate( radiance % cloudy   ,stat=alloc_status(10))
Deallocate( radiance % total    ,stat=alloc_status(11))
Deallocate( radiance % bt       ,stat=alloc_status(12))
Deallocate( radiance % bt_clear ,stat=alloc_status(13))
Deallocate( radiance % upclear  ,stat=alloc_status(14))
Deallocate( radiance % dnclear  ,stat=alloc_status(15))
Deallocate( radiance % reflclear,stat=alloc_status(16))
Deallocate( radiance % overcast ,stat=alloc_status(17))
Deallocate( radiance % downcld  ,stat=alloc_status(18))
Deallocate( radiance % out       ,stat= alloc_status(19))
Deallocate( radiance % out_clear ,stat= alloc_status(20))
Deallocate( radiance % total_out ,stat= alloc_status(21))
Deallocate( radiance % clear_out ,stat= alloc_status(22))
Deallocate(pr_radcld ,stat= alloc_status(31))
Deallocate(pr_trans ,stat= alloc_status(32))
Deallocate(pr_emis ,stat= alloc_status(33))
Deallocate(pr_trans_lev ,stat= alloc_status(34))
If( Any(alloc_status /= 0) ) Then
  errorstatus = errorstatus_fatal
  Write( errMessage, '( "mem deallocation error")' )
  Call Rttov_ErrorReport (errorstatus, errMessage, NameOfRoutine)
```

```
    Stop
  End If

  Call rttov_dealloc_coef (errorstatus, coef)
  If(errorstatus /= errorstatus_success) Then
    Write( errMessage, '( "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

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(1X,'CALCULATED BRIGHTNESS TEMPERATURES: SAT =',I2,&
      &' ZENITH ANGLE=',F6.2, &
      &' AZIMUTH ANGLE=',F7.2,' SURFACE TYPE=',I2)

End Program example_fwd
```