Generation of the RBC Auxiliary file:

Detailed Processing Model

A. Dabas, M.L. Denneulin, P. Poli

Change log

Version	Date	Comment
1.0	20/10/05	Initial version
1.1	05/07/06	The document is updated in order to match the last definition of the RBC auxiliary file contained in version 1.0 of document AE-IF-ECMWF-L2BP-001 (L2B IODD / ICD).
1.2	25/05/2007	Corrections following comments from C. Caspar. A new reference is made to the document that explains how and when auxiliary files are generated/updated at the ACMF. Other references are updated.
1.3	11/04/2008	Corrections on units and extrapolation.
1.4	04/11/2008	Input to RBC is now AUX_CSR rather than AUX_ISR.
1.5	13/11/2008	Minor corrections (inversion of sign of F_d, correction of the definition of frb, notation f_d for the frequency Doppler shift, equation (4)).).
2.0	17/01/2012	New version following the MATLAB recoding of the AUX_RBC generator. Modifications of the document concern: - the computation of $Fint\_R$ in section 6.8. - Table 1 (list of input / output parameters).
2.1	23/04/2013	– Correction of a bug in the convolution of spectra and transmission curves. – Computation of Fint_R from ISR instead of CSR. – Update of references.
2.2	06/10/2014	– Update of references. – Update of code description. – Removal of Tenti S6 code. It is replaced by the analytical model developed by Witschas, *Appl. Opt.*, **50**, 267-270.
3.0	10/03/2015	– Update of references. – IPF 198: validity period. – Removal of reference to pure Rayleigh spectra.
3.2	15/12/2015	– Update references – Allow selection of either pure Rayleigh or Rayleigh-Brillouin spectrum for molecular spectra (see §6.6) – Update parameter settings in §5.2 and Table 1;
3.3	30/06/2016	– Assition of the possibility to filter ISR data before inverting the internal response.
4.0	24/07/2017	– Update of Tenti spectrum function (shear viscosity is now temperature dependent)

Table of content

1 Introduction.

The present document gives a detailed description of the RBC generator. This processor computes the information needed for correcting Rayleigh winds from pressure, temperature and Mie contamination effects – the so-called Rayleigh-Brillouin correction scheme.

A theoretical baseline of the Rayleigh-Brillouin correction scheme is given in [RD1]. For clarity purposes, a short summary is proposed in section 3. It is based on an interpolation in a 3D matrix – the so-called look-up table. This matrix is formed by the frequency shifts corresponding to an array of possible Rayleigh responses R_k under atmospheric conditions characterized by pressure and temperature combinations (P_i, T_j) paving the "area" of all possible meteorological conditions likely to be found in the atmosphere.

The output of the RBC generator is stored in a single auxiliary file (denoted AUX_RBC_L2, described in details in [RD4]). This file is input to the L2B processor. As its content depends on the characteristics of the instrument through the end-to-end transmission functions of the two Fabry-Perot interferometers (denoted $T_A(f)$ and $T_B(f)$ in section 3), it must be updated every time a new characterization or an update of $T_A(f)$ and $T_B(f)$ is carried out (see [RD2]). This is the case every time a new AUX_CSR is generated.

In the following, section 3 summarizes the theoretical baseline of the Rayleigh-Brillouin correction scheme. The input/output data needed by the generator algorithm are listed in section 4. At last, the detailed processing model is presented in section 6.

2 Reference documents.

	Ref	Title	Ver	Date
[RD1]	1833404/NL/MM	Impact of line shape on Aeolus-ADM Doppler estimates		
[RD2]	AE-TN-MFG-L2P-CAL-002	Generation/update of AUX_RBC and AUX_CAL at ACMF.	3.1	30/06/2016
[RD3]	AE-TN-MFG-CAL-003	Generation and update of AUX_CSR	4.0	24/07/2017
[RD4]	AE-TN-MFG-GS-0003	ADM-Aeolus Rayleigh-Brillouin Correction Lookup Tables Generator: Input/Output Data Definitions Interface Control Document	3.3	30/06/2016

3 Theoretical baseline.

The main output of the algorithm described in section 6 is a 3D look-up table $\mathcal{F}_{i,j,k}$ defined by:

$$R_{P_i,T_j}(\mathcal{F}_{i,j,k}) = R_k \tag{1}$$

where R_k is the k-th element of the array R of Rayleigh responses, P_i the i-th element of the pressure array P, T_j the j-th element of the temperature array T and $R_{P,T}(f_d)$ is the function giving the theoretical Rayleigh response for a frequency shift f_d, an atmospheric pressure P, and a temperature T.

$$R_{P,T}(f_d) = \frac{N^A_{P,T}(f_d) - N^B_{P,T}(f_d)}{N^A_{P,T}(f_d) + N^B_{P,T}(f_d)} \quad (2)$$

Here, $N^A_{P,T}(f_d)$ and $N^B_{P,T}(f_d)$ are the numbers of photo-electrons detected on channels A and B. These numbers are related to the spectrum of the backscattered Rayleigh light $I_{P,T}(f_d)$ by the convolution product:

$$N^{A,B}_{P,T}(f_d) = \int_{-FSR}^{+FSR} I_{P,T}(x - f_d) T_{A,B}(x) dx \quad (3)$$

FSR is the Free Spectral Range of the Fabry-Perot interferometers. The dependence of $I_{P,T}(f_d)$ as a function of P and T is discussed in [RD1]. It converges to the classical Gaussian line-shape at low pressures:

$$I_{P\to 0,T}(f) = \frac{1}{\sqrt{2\,\pi}\sigma_f(t)} \exp\left(-\frac{(f - f_{laser})^2}{2\,\sigma_f^2(T)}\right) \quad (4)$$

where f_{laser} is the laser frequency, and

$$\sigma_f(T) = \frac{2}{\lambda}\sqrt{\frac{kT\mathcal{N}_a}{M}} \quad (5)$$

Here, λ is the laser wavelength (in m), $k = 1.38\ 10^{-23}\ JK^{-1}$is Botzman's constant, $\mathcal{N}_a = 6.02\ 10^{23} mol^{-1}$ is Avogadro's number and $M = 0.029\ kg$ is the mass of one mole of air.

For larger values of the pressure P, the line-shape is modified. Two peaks called Brillouin doublets appear on either side of the standard, Gaussian curve. The total area of the return spectrum remains unchanged, that is, the Brillouin effect does not affect the molecular backscatter coefficient. As it can be seen from Figure 1 (note that it shows $I_{P,T}(f)$ in MHz^{-1}), the modification of the line-shape is small for all the conditions likely to be met in the atmosphere, but the impact on the velocities measured from the Rayleigh signals cannot be neglected and needs to be corrected. To compute the correction, a model is needed for $I_{P,T}(f)$. We use the model developed by Witschas, 2011[1]. A MATLAB version of the code is given in appendix 1.

The processor allows the selection of pure Gaussian spectra (equation (4) above) instead of a Rayleigh-Brillouin. The selection is commanded by a switch in the parameter settings (see Table 1). This was done for compatibility purposes with the simulation environment of the mission. For real data, the model shall be Rayliegh-Brillouin.

[1] Witschas, B., 2011 : Analytical model for Rayleigh-Brillouin line shapes in air. *Applied Optics*, **50**, 267-270.

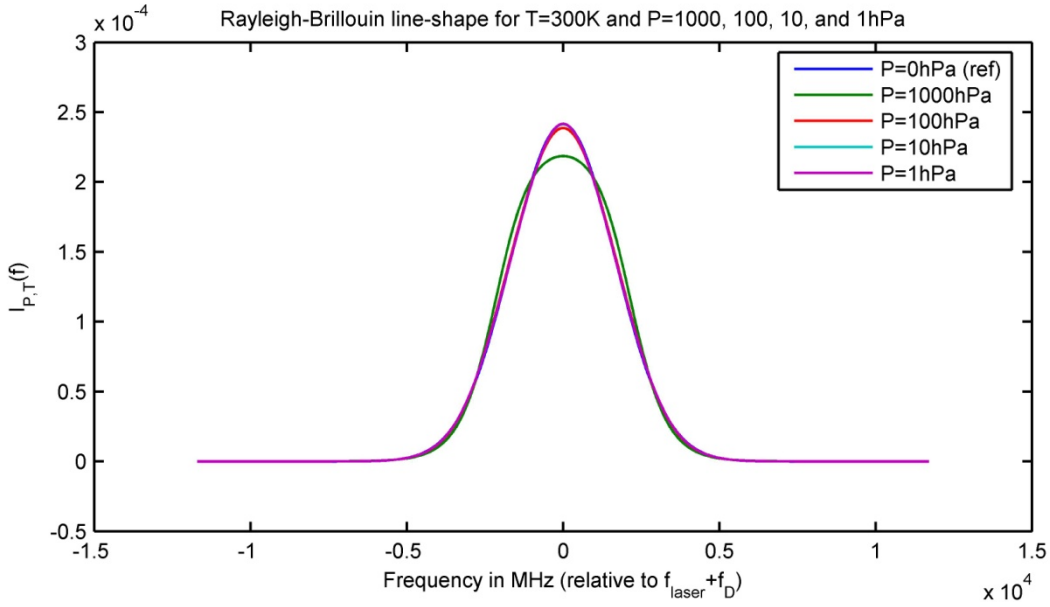

Figure 1: Rayleigh-Brillouin line-shapes for $\boldsymbol{T = 300K}$ and different pressures $\boldsymbol{P}$. The blue line is the classical Gaussian line-shape which serves here as a reference. The line-shapes for $\boldsymbol{P = 1hPa}$, $\boldsymbol{10hPa}$, $\boldsymbol{100hPa}$ and $\boldsymbol{1000hPa}$ are computed using Tenti S6 model. It can be seen that the departure from the Gaussian curve is larger as the pressure increases. The unit of the x-axis is $\boldsymbol{MHz}$. The y-axis shows $\boldsymbol{I_{P,T}(f)}$ in $\boldsymbol{MHz^{-1}}$.

4 Input / Output of RBC generator

The inputs and outputs of the RBC generator are summarized in Table 1. The parameters are listed in column 1. Columns 2 and 3 give a short description and indicate where and under which name the parameter can be found.

5 Validity Period

The validity period of the output AUX_RBC product is identical to the input AUX_CSR. It is determined by a call to the $UpdateValidityDates$ function.

6 Processing Model

6.1 Read the transmission functions T_A and T_B from last *AUX_CSR*.

The code looks for the latest version of the AUX_CSR in the AUX directory, reads its content. The outputs are two sets of vectors:

- One set is formed by three vectors of equal length denoted F_FP (in GHz) TA_FP and TB_FP. They characterize the *Coherent Spectral Registration* (CSR) of the Fabry-Pérot A and B interferometers (see [RD3] for details about the CSR).
- The other set contains the *Instrument Spectral Registration* (ISR) from which the CSR is derived. It forms the structure isr made of of the three fields f, TA and TB. In the three

fields are vectors of equal lengths characterizing the transmission of Fabry-Pérot A and B interferometers (TA and TB respectively) as a function of the frequency f.

Note that the code will not stop if several AUX_CSR files are found in the AUX_CSR directory. It will take the latest one.

MATLAB Code

```
files = dir(fullfile(diraux,'/AE_*_AUX_CSR_1B*.EEF'));
if isempty(files),
    error('[AUX_RBC]: No AUX_CSR file found in AUX directory.');
else
    nfiles = length(files);
    if nfiles > 1,
        warning('[AUX_RBC]: Beware, more than one AUX_CSR file found.');

        % Always take latest AUX_CSR files
        tfile = zeros(nfiles,1);
        for kfile = 1:nfiles,
            tfile(kfile) = datenum(files(kfile).date);
        end
        [xdum,kmax] = max(tfile);
        csrFileName = files(kmax).name;
        warning(sprintf('[AUX_RBC]: I take the latest AUX_CSR dated %s.',…
            datestr(tfile(kmax))));
    else
        csrFileName = files(1).name;
    end
    csr = readXml(fullfile(diraux,csrFileName));
    [tstart,tstop] = UpdateValidityDates(csrFileName,0,0);
    clear kfile nfiles
end
% Reads CSR part
node  =
csr.Earth_Explorer_File.Data_Block.Corrected_Spectral_Registration.List_of_Data_Se
t_Records.Data_Set_Record(1).List_of_CSR_Frequency_Steps.CSR_Frequency_Step;
F_FP  = str2double(vertcat(node.Laser_Freq_Offset));
TA_FP = str2double(vertcat(node.Rayleigh_A_Response));
TB_FP = str2double(vertcat(node.Rayleigh_B_Response));

% Reads ISR part
node   =
csr.Earth_Explorer_File.Data_Block.Internal_Spectral_Registration.List_of_Data_Set
_Records.Data_Set_Record(1).List_of_ISR_Results.ISR_Result;
isr.f  = str2double(vertcat(node.Laser_Freq_Offset));
isr.TA = str2double(vertcat(node.Rayleigh_A_Response));
isr.TB = str2double(vertcat(node.Rayleigh_B_Response));
```

6.2 Read parameter settings

The code looks for the AUX_PAR_RB file in the AUX directory and reads its content. The AUX_PAR_RB contains all the settings of the RBC generator. Its content is described in Table 1. Note that the code will not stop if more than one AUX_PAR_RB file is found. It will issue a warning and reads the first one it has found.

MATLAB Code

```
files = dir(fullfile(diraux,'AE_*_AUX_PAR_RB*.EEF'));
if isempty(files),
    error('[AUX_RBC]: No AUX_PAR_RB file found in AUX directory.');
elseif length(files)>1,
    error('[AUX_RBC]: More than one AUX_PAR_RB files found in AUX directory.');
else
```

```
    parFileName = files(1).name;
    param = readXml(fullfile(diraux,parFileName));
end
% Spectrum model (TENTI or GAUSS)
specMod =
str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.RBC_S
pec_Model{1});

% Grid for RBC LUT
Pmin =
str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.Pmin{
1});
Pmax =
str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.Pmax{
1});
DeltaP  =
str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.Delta
P{1});
Tmin =
str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.Tmin{
1});
Tmax =
str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.Tmax{
1});
DeltaT  =
str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.Delta
T{1});
Rmin =
str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.Rmin{
1});
Rmax =
str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.Rmax{
1});
DeltaRR =
str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.Delta
RR{1});

% FP FSR and initial value for FWHM
FP.FSR =
str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.Fabry
_Perot.FSR{1});
FP.FWHM =
str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.Fabry
_Perot.FWHM{1});

% Initial value for Fizeau FSR and selection of model for reflection on Fizeau
(DLR or E2S)
Fiz.FSR =
str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.Fizea
u.FSR{1});
Fiz.Fiz_Reflec_Model =
param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.Fizeau.Fiz_Refle
c_Model{1};

% Selection of raw ISR or fitted ISR for computation of internal Rayleigh response
% Parameters for ISR fitting
flagFitISR  =
param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.flagGen{1};
gen.iterMax =
str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.Gen.M
ax_Iterations{1});
gen.tol     =
str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params.Gen.T
olerance{1});

% Other parameters (converted into GHz)
USR =
0.001*str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params
.USR{1});
df  =
0.001*str2double(param.Earth_Explorer_File.Data_Block.RBC_Proc_Param_ADS.RB_Params
.df{1});
```

6.3 Input arrays.

Generate vectors P_Grid , T_Grid and RR (outputs 1, 2 and 3) from inputs 1, 2 and 3.

MATLAB Code

```
P_Grid = Pmin:DeltaP:Pmax;
T_Grid = Tmin:DeltaT:Tmax;
RR     = Rrmin:DeltaRR:Rrmax;
```

6.4 Resample transfer functions T_A and T_B.

The purpose is to obtain transmission functions across the spectral range $[-FSR, +FSR]$ with the frequency resolution df prescribed to the processor (input **Erreur ! Nous n'avons pas trouvé la source du renvoi.** in Table 1). In principle, the frequency range of CSR is already $[-FSR, +FSR]$ but the processor makes sure this is actually the case. This is done by assuming the transmission functions are periodical with the known periodicity FSR (input parameter 8 in Table 1). The outputs are a new vector of frequencies $fint$ and two new transmission functions $Taint$ and $Tbint$. Note that the vector of frequencies $fint$ must be centred about the zero frequency.

MATLAB Code

```
% Resample CSR.
fint=[df:df:FSR];fint=[-fliplr(fint),0,fint];
fwrap=fint-FSR*floor(fint/FSR+0.5);
Taint=interp1(F_FP,TA_FP,fwrap,'method');
Tbint=interp1(F_FP,TB_FP,fwrap,'method');
```

In the preceding code, 'method' designates one of the possible interpolation method that MATLAB can handle. In the ILIAD study, a 'spline' method was systematically retained.

6.5 Define the vector of frequency offsets f_d for which $R_{P,T}(f_d)$ must be calculated

The vector paves the frequency interval $[-USR/2, USR/2]$ with the frequency resolution df. Note that it must be centered about 0 and contain frequencies **in a descending order**. Note also that the code below assumes the USR (parameter 1 in Table 1) is in GHz while it is in MHz in the AUX_PAR_RB file. A conversion must be done beforehand.

MATLAB Code

```
% Frequency Offsets for which NA and NB are computed
Fd=[df:df:USR/2];
Fd=[fliplr(Fd),0,-Fd];
```

6.6 Compute Rayleigh-Brillouin spectra

The result is a 3D array $Spec$. The spectra are given for all the pressures in P_Grid (dimension 1), temperatures T_Grid (dimension 2), and frequencies in array frb spanning the interval $[-FSR - \max(F_d), FSR + \max(F_d)]$. The computation of the spectra is done with the $Tenti\_Spectrum$ function detailed in appendix 1. The $Tenti\_Spectrum$ function simulates Rayleigh-Brillouin spectra.

MATLAB Code

```
frb=[df:df:FSR+Fd(1)]; % in GHz
frb=[-fliplr(frb),0,frb];
if strcmp(specMod,'TENTI'),
   for i=1:length(P_Grid),
      for j=1:length(T_Grid),
          Spec(i,j,:)=Tenti_Spectrum(frb,0,T_Grid(j),P_Grid(i));
      end
   end
elseif strcmp(specMod,'GAUSS'),
   for i=1:length(P_Grid),
      for j=1:length(T_Grid),
          Spec(i,j,:)=Rayleigh_Spectrum(frb,0,T_Grid(j),P_Grid(i));
      end
   end
else
   error('[GENERATERBC]: Spectral model must be either TENTI or GAUSS.');
end
```

6.7 Convolve the spectra and the transfer functions, compute and inverse the expected responses.

The convolution of the spectra and the transfer functions is done by a finite difference approximation of equation (3) followed by equation (2). The result is stored in $RResponse$. The calculation is carried out for all the frequencies contained in the vector Fd. Then the response curve is inverted using an interpolator (linear or spline). The result is stored in the 3D matrix (the look-up table) $Fcalib\_PTR$.

MATLAB Code

```
NT=length(Taint);
Na_F=zeros(length(P_Grid),length(T_Grid),length(Fd));
Nb_F=zeros(length(P_Grid),length(T_Grid),length(Fd));
Fcalib_PTR= zeros(length(P_Grid),length(T_Grid),length(RR));
for i=1:length(P_Grid),
   for j=1:length(T_Grid),
     for k=1:length(Fd),
       Na_F(i,j,k)=df*sum(Taint'.* squeeze(Spec(i,j,k:k+NT-1)));
       Nb_F(i,j,k)=df*sum(Tbint'.* squeeze(Spec(i,j,k:k+NT-1)));
       RResponse(k)=(Na_F(i,j,k)-Nb_F(i,j,k))...
                              /(Na_F(i,j,k)+Nb_F(i,j,k));
     end
   Fcalib_PTR(i,j,:)=interp1(Rresponse,Fd,RR,'method','extrap');
   end
end
```

The 'extrap' option of the interp1 interpolator commands extrapolated frequencies are returned when a response in RR is outside the interval covered by RResponse. Without this option, the interp1 returns a missing value in that case. The extrapolation is required by the L2B processor, it avoids having to deal with missing values. In principle, a linear method of interpolation should work but it was not tested. The method 'spline' was systematically used in the AUX_RBC prototype generator.

6.8 Compute and inverse the theoretical response on the internal reference channel.

This is obtained by forming and inverting with an interpolator the ratio $(isr.TB - isr.TB)/(isr.TA + isr.TB)$ where $isr.TA$ and $isr.TB$ are the transmissions at the frequency offsets $isr.f$ of the internal path as recorded in the ISR data set of the AUX_CSR file (see Table 1 below). The inversion is done on a frequency interval $[-USR/2, USR/2]$.

If the option $flagFitISR$ is equal to "TRUE", the computation of the internal response is preceded by a filtering of ISR data. The filtering is done by the function FitISR described in details section 7.4 of [RD3].

MATLAB Code

```
if strcmp(flagFitISR,'TRUE'),

    paramfit.gen = gen;
    paramfit.FP  = FP;
    paramfit.Fiz = Fiz;
    paramfit.df  = 1000*df;

    fittedisr = FitISR(isr.f,isr.TA,isr.TB,paramfit);

    indf = find(abs(fittedisr.f)<=0.5*USR);
    IntResp = (fittedisr.TA-fittedisr.TB)./(fittedisr.TA+fittedisr.TB);
    Fint_R  = interp1(IntResp(indf),fittedisr.f(indf),RR,'spline','extrap');

elseif strcmp(flagFitISR,'FALSE'),

    indf = find(abs(isr.f)<=0.5*USR);
    IntResp = (isr.TA-isr.TB)./(isr.TA+isr.TB);
    Fint_R  = interp1(IntResp(indf),isr.f(indf),RR,'spline','extrap');

else
    error('[GENERATERBC]: FLAGFITISR MUST BE TRUE OR FALSE.');
end
```

6.9 Flip `Fd`, `Na_F` and `Nb_F`.

At this stage, `Fd` contains frequencies in descending order. This was required by the numerical scheme that computes `Na_F` and `Nb_F` (see 6.7 above). In the output file, frequency offsets must be in ascending order. The vector `Fd` must thus be flipped as well as the third dimension of `Na_F` and `Nb_F` (since `Na_F(i,j,k)` contains the number of photons for pressure `P_Grid(i)`, temperature `T_Grid(j)` and `Fd(k)`).

MATLAB Code

```
Fd = fliplr(Fd);
Na_F = Na_F(:,:,end:-1:1);
Nb_F = Nb_F(:,:,end:-1:1);
```

6.10 Prepare data for the output

Before they are output, data must be converted to the units defined in the IODD. Pressure and temperatures must be in units of $0.01hPa$ and $0.01K$ respectively, frequencies must be either in GHz (FSR), or MHz (USR and df) or otherwise in Hz, and the spectra must be in Hz^{-1}.

MATLAB Code

```
rbc.FSR         = FSR;
rbc.USR         = 1000*USR;
rbc.df          = 1000*df;
rbc.P_Grid      = 100*P_Grid;
rbc.T_Grid      = 100*T_Grid;
rbc.F_Gridtmp   = 1e9*frb;
rbc.Spec_Grid   = 1e-9*Spec;
rbc.F_FP        = 1e9*F_FP;
rbc.TA_FP       = TA_FP;
rbc.TB_FP       = TB_FP;
rbc.Fd          = 1e9*Fd;
rbc.RR          = RR;
rbc.Fcalib_PTR = 1e9*Fcalib_PTR;
rbc.Fcalib_Err = zeros(size(Fcalib_PTR));
rbc.Na_PTFd     = Na_F;
rbc.Nb_PTFd     = Nb_F;
rbc.Fint_R      = 1e9*Fint_R;
```

Table 1: Input and output parameters of RBC generator.

INPUT Parameters			
1	specMod	Name of the spectral model used for simulating molecular spectra. It can be either GAUSS (pure molecular return) or TENTI (Rayleigh-Brillouin spectra). The latter must be used for real data, the former is kept for compatibility purposes with E2S data.	Set by the user via the AUX_PAR_RB file
2	Pmin, Pmax, DeltaP	Minimum pressure, maximum pressure and pressure step defining the pressure array input to the RBC generator. All are specified in hPa	Set by the user via the AUX_PAR_RB file (see [RD4])
3	Tmin, Tmax, DeltaT	Minimum temperature, maximum temperature and temperature step defining the temperature array input to the RBC generator. All are specified in K.	
4	Rrmin, Rrmax, DeltaRR	Minimum response, maximum response and response step defining the response array input to the RBC generator	
5	F_FP	Frequencies (in GHz) for which TA_FP and TB_FP are given	Output of *Coherent Spectral Registration* (Laser_freq_offset, Rayleigh_A_Response and Rayleigh_B_Response, see table 11 of [RD3].
6	TA_FP	Transfer function of FP A as a function of the frequency (relative to f_0).	
7	TB_FP	Transfer function of FP A as a function of the frequency (relative to f_0).	
8	FP.FSR	Free spectral range of FP A and B (in GHz)	Characterized at ground, valid for the whole mission. Reported nowhere, it is input to the RBC generator via the AUX_PAR_RB.
9	FP.FWHM	Full-Width-Hal-Max of FP A and FP B (in GHz). This value is used as first guess for the filtering of ISR data before the Inversion of the internal response.	Set by the user via the AUX_PAR_RB file (see [RD4]).
10	Fiz.FSR	First guess of the periodicity of the reflection of the Fizeau (in GHz)	Set by the user via the AUX_PAR_RB file (see [RD4]).
11	Fiz.Fiz_Reflec_Model	Either E2S or DLR. E2S is for testing purposes with the E2S. For real data select DLR.	Set by the user via the AUX_PAR_RB file (see [RD4]).
12	flagFitISR	Either TRUE or FALSE. If TRUE, the ISR is filtered before the inversion of the internal response.	Set by the user via the AUX_PAR_RB file (see [RD4]).
13	gen.iterMax	Parameters required by the FitISR function.	Set by the user via the AUX_PAR_RB file (see [RD4]).
14	gen.tol		
15	isr.f	Frequencies (in GHz) for which isr.TA and isr.TB are given	ISR data recorded in the AUX_CSR (in the ISR GADS). See table 9 of [RD3].
16	isr.TA	Internal path transmission of FPA.	
17	isr.TB	Internal path transmission of FPB	
18	USR	Useful Spectral Range in MHz.	Set by the user via the AUX_PAR_RB file.
19	df	Spectral resolution in MHz (default df=25 MHz).	Set by the user via the AUX_PAR_RB file.
OUTPUT Parameters			
1	FSR	Free spectral range of the Fabry-Perot interferometers.	Read from AUX_PAR_RB (input 8) and reported in AUX_RBC_L2 (see [RD4]).
2	USR	Useful spectral range	Read from AUX_PAR_RB (input 1) and reported in AUX_RBC_L2 (see [RD4]).

3	df	Frequency resolution	Read from AUX_PAR_RB (input **Erreur ! Nous n'avons pas trouvé la source du renvoi.**) and reported in AUX_RBC_L2 (see [RD4]).
4	P_Grid	Array of the pressures for which the RBC look-up tables and photocounts are computed.	Stored in AUX_RBC_L2 (see [RD4]).
5	T_Grid	Array of the temperatures for which the RBC look-up tables and photocounts are computed.	
6	RR	Array of Rayleigh responses.	
7	Fd	Array of frequency shifts (in GHz) for which the photocounts Na_f and Nb_f are computed.	
8	Fcalib_PTR	RBC Look-up table . Contains the frequency shifts (in GHz) yielding the Rayleigh responses in RR under the all the combinations of atmospheric conditions P_Grid and T_Grid.	
9	Fint_R	Frequency shifts (in GHz) yielding the responses RR on the internal reference path.	
10	Na_f	Theoretical number of Photocounts in FP A.	
11	Nb_f	Same for FP channel B	
12	Fcalib_R_error	Error quantifier of Fcalib data. Presently set to 0 by default.	
13	Frb	Array of frequencies between –FSR-USR/2 and +FSR+USR/2 (in GHz)	
14	Spec_Grid_PTF	Rayleigh-Brillouin spectra in Hz^{-1}, for all the pressures in P_Grid, all the temperatures in T_Grid, and all the frequencies in frb	

APPENDIX A TENTI

```
function spec=Tenti_Spectrum(freq,f0,T,P)
% FUNCTION SPEC=TENTI(FREQ,F0,PRESSURE,TEMPERATURE)
%
% Computes the spectrum of light scattered by air molecules at pressure P
% (in hPa) and temperature T (in K). The spectrum is given in GHz-1. FREQ
% (in GHz) is a vector containing the input frequencies (offsets from the
% baseline frequency). F0 (in GHz) is the central frequency of the spectrum
% expressed as a Doppler shift from the baseline frequency.

% The computation is based on Witschas, Applied Optics, vol 50, n°3, pages
% 267-270.
%
% 27/07/2016: The shear viscosity is not constant anymore but a function of
%             the temperature.

%--------------------------------------------------------------------------
%_____________________ Control input arguments _____________________________
%--------------------------------------------------------------------------
if nargin < 4,
  error('[Tenti_Spectrum] : Not enough input arguments.');
elseif nargin > 4,
  error('[Tenti_Spectrum] : Too many input arguments.');
elseif isempty(freq),
  error('[Tenti_Spectrum] : Parameter "F" is empty');
elseif isempty(f0),
  error('[Tenti_Spectrum] : Parameter "F0" is empty');
elseif isempty(T),
  error('[Tenti_Spectrum] : Parameter "T" is empty');
elseif isempty(P),
  error('[Tenti_Spectrum] : Parameter "P" is empty');
end

% Declaration of constants
lambda = 354.8e-9;                % Wavelength in [m]
kb     = 1.38e-23;                % Boltzman's constant [J/s]
M      = 4.789e-26;      % Mass of an air molecule [kg]
eta0   = 1.846e-5;       % Shear viscosity

% Width of spectrum and normalized frequency offsets
v0=sqrt(2*kb*T/M);               % Width in m/s
deltanu=2e-9*v0/lambda;          % Width in GHz
x=(freq-f0)/deltanu;             % Normalized frequency shifts.

% Shear viscosity
TrefBrillouin = 300;
TrefShear     = 110.4;
eta =
eta0*sqrt(power(T/TrefBrillouin,3).*(TrefBrillouin+TrefShear)./(T+TrefShear));

% y coefficient
y = 1e-7*P/(2*pi*deltanu*eta);

% Coefficients of Witschas' model
A     = 0.18526*exp(-1.31255*y)+0.07103*exp(-18.26117*y)+0.74421;
sigR  = 0.70813-0.16366*y*y+0.19132*y*y*y-0.07217*y*y*y*y;
sigB  = 0.07845*exp(-4.88663*y)+0.80400*exp(-0.15003*y)-0.45142;
xB    = 0.80893-0.30208*power(0.10898,y);

% Spectrum
spec = A*exp(-0.5*(x/sigR).*(x/sigR))/sigR + 0.5*(1-A)*(exp(-0.5*(x-xB).*(x-
xB)/(sigB*sigB))+exp(-0.5*(x+xB).*(x+xB)/(sigB*sigB)))/sigB;
```

```
spec = spec/(sqrt(2*pi)*deltanu);
```