TITLE: # Planck LFI – FS_ZOD: A Simulator of the Zodiacal Light Emission for the PLANCK Mission

DOC. TYPE: OAT-TN

PROJECT REF.: PL-LFI-OAT-TN–023 **PAGE:** I of IV, 00

ISSUE/REV.: 1 **DATE:** July 10, 2001

Issued by	**Michele Maris** **LFI DPC Development Team**	**Date:** 10 July 2001 **Signature:** _____
Agreed by	**F. PASIAN** **LFI Program Manager**	**Date:** December 12, 2001 **Signature:** _____
Approved by	**R.C. BUTLER** **LFI Program Manager**	**Date:** **Signature:** _____
Approved by	**N. MANDOLESI**	**Date:**

	LFI Principal Investigator	**Signature:** _____

DISTRIBUTION LIST

Recipient	Company / Institute	E-mail address	Sent

CHANGE RECORD

Issue	Date	Sheet	Description of Change	Release
1	2001-07-10		Initial Issue	0

TABLE OF CONTENTS

1 SCOPE

Description of the package FS_ZOD to simulate the zodiacal light signal in the framework of the PLANCK mission. The package starts from the COBE model for ZLE of Kelsall et al. (1998) [AD3].

1.1 LIMITS OF APPLICABILITY

The package is designed to work with the model outlined in [AD3].

In the present version the *Smooth Component* only has been introduced, but the package is ready to add the remaining components of the model.

2 APPLICABLE/REFERENCE DOCUMENTS

2.1 APPLICABLE DOCUMENTS

[AD1] *FORTRAN 90 Programming Guidelines for PLANCK/LFI*
M. Maris
1999 March 8, Issue 0.1, LFI-OAT- 0002.01

[AD2] *Planck LFI - FS_DIP: A Flight Simulator Module to Simulate the Cosmological Dipole*
M. Maris
2001 Jul 20, Issue 0.0, LFI-OAT-TN-00X

[AD3] Kelsall T., Weiland J. L., Franz B. A., et al., (1998), ApJ, 508, 44 (also astro-ph/9806250)

2.2 REFERENCE DOCUMENTS

[RD1] Wheelock, S.L., et al., (1994), "IRAS Sky Survey Atals Explanatory Supplement", JPL Pub. 94-11

2.3 ACRONYMS LIST

BSC	Bright Sources Catalog
Bsys	Blind Systematic
FS	Flight Simulator
IPD	Inter Planetary Dust
MOBJ(s)	Moving Object(s)
MPS(s)	Moving Point Source
PS(s)	Point Source(s)
PSC	Point Sources Catalog
SS	Solar System
Sys	Systematic
TOD(s)	Time Ordered Data
ZLE	Zodiacal Light Emission

3 PACKAGE STRUCTURE

The package is written in FORTRAN 90 according to the conventions described in [AD1].

The package is structured in three main levels:
1. A set of library modules.
2. The package environment: `zod_env.f90`.
3. A command line driver: `zod_main.f90`.
4. The interface with the Flight Simulator is assured through the `fs_zod_env.f90` module plus some code embedded in the flight simulator. It requires the code described in [AD2].

The library modules are designed according to the principles of OOP in F90. The library module assures maximal reusability of the software. They carries the definition of all the objects needed to handle the zodiacal light emission plus the related methods. They to carries only general purpose methods.

The package environment holds definitions and methods required to integrate the package inside an application. This module is less standardised than the previous ones, it is designed to simplify the integration but it may be modified whenever required during the integration phase introducing methods that are not rigorously standardised or of general use.

The driver allows to generate the ZLE starting from a list of pointing positions generated by the FS. It used for testing and to document how to use the package too.

The interface with the FS allows to extract the pointing list from the FS, and to save it on a separated file.

4 LIST OF ROUTINES AND SERVICES

Routines and services are listed and described here.

4.1 UNIT: PHYSICAL_PARAMETERS_CGS.F90

4.1.1 SUBROUTINE: PHYSICAL_PARAMETERS_CGS_DISPLAY

Subroutine to print on the screen the list of Physical Parameters in cgs

4.2 UNIT: INTERPOLATION.F90

4.2.1 SUBROUTINE: POLINT(XA,YA,N,X,Y,DY)

This is the POLINT routine from Numerical Recipes

4.2.2 SUBROUTINE HUNT(XX,N,X,JLO)

This is the HUNT routine from Numerical Recipes

JLO is the output value so that XX(JLO).le.X.le.XX(JLO+1)
JLO = 0 or N if no bracketing is find

Hunt by Num.REc. searches in the Array XX(1..N) an element so that X is between XX(JLO) and XX(JLO+1). JLO in input contains the starting element, while in output the result of the search, JLO on output is 0 or N if X is outside the range of XX values.

XX musts be ordered in increasing or decreasing.

4.2.3 FUNCTION DISCRITP(I,X1,X2,N)

Discrete interpolation at I-esimal point between x1 to x2, N the number of intervals (number of samples - 1).

4.2.4 FUNCTION CLITP(X,X0,X1,Y0,Y1)

CLITP is the linear interpolation formula between two points

4.2.5 FUNCTION RLTP(X,LX,LY,IMAX,LAST,ZERO)

Computes Y(X) for a function tabulated in the vector LX, LY by linear interpolation.
IMAX is the list length, LAST stores the last readed value, it musts left unchanged between one call and the other. This routine uses the HUNT routine from Numerical Recipes
At the first call put Last = 1.
ZERO is the value that function has to assume outside boundaries

4.2.6 FUNCTION RLINTERP(X,LX,LY,IMAX,LAST,ZERO)

Computes Y(X) for a function tabulated in the vector LX, LY by linear interpolation.
Imax is the list length, LAST stores the last readed value, it musts left unchanged between one call and the other
This routine uses the HUNT routine from Numerical Recipes
At the first call put Last = 1
ZERO is the value that function has to assume outside boundaries
If the result of the interpolation has to be multiplied by a normalizzation factor RNFAC and ZERO is the value of the function already multiplied by it, then the calling routine has to pass ZERO/RNFAC.
As an example:

y = RNFAC * (X,LX,LY,IMAX,Last,ZERO/RNFAC)

4.3 UNIT: INTEGRALCOM_MOD.F90

A unit containing common blocks for the use in INEGRATOR.

4.4 UNIT: INTEGRATOR.F90

4.4.1 SUBROUTINE TRAPZD(FUNC,A,B,S,N)

4.4.2 SUBROUTINE QCRUD(F,A,B,RES,N)

Crude estimation of an integral with n-iterated calls to Trapzd.
Res is the integration result.

4.4.3 SUBROUTINE QSIMP(F,A,B,RES,EPS,NMAX)

Estimation of an integral with a control over error. The procedure stops when the integration error is less then Eps or when a number of calls to Trapzd greater then Nmax is done. Res is the integration result
The error considered in the integration according to the rule: Eps > 0 then absolute error, Eps < 0 then relative error

4.4.4 SUBROUTINE QROMB(FUNC,A,B,SS,STATUS,ACC,NMAX)

Estimation of an integral with a control over error.
The procedure stops when the integration error is less then Eps or when a number of loops greater then Nmax is done. SS is the itegration result. Status is 0 if all whent good
The error considered in the integration according to the rule: Eps > 0 then absolute error, Eps < 0 then relative error

4.4.5 FUNCTION RINTEGRATE(F,XINF, XSUP)

Executes the integration of F(X) from XInf to XSup

4.5 UNIT: ZOD_DEBUG.F90

Debugging services.

4.5.1 SUBROUTINE DIE(CALLER,MOTIVATION,STAT)

Closes the program producing an error message.

4.6 UNIT: ZOD_BLACKBODY.F90

Black Body related routines.

4.6.1 REAL FUNCTION BBL_CGS(LAMBDA,T)

BB(lambda,T) thermal radiance function cgs
lambda in cm
bbl_cgs = erg/(cm2 sec cm sterad)

4.6.2 REAL FUNCTION BBN_CGS(NU,T)

BB(nu,T) thermal radiance function cgs
nu in Hz
bbn_cgs = erg/(cm2 sec cm sterad)

4.6.3 REAL FUNCTION BBL_RJ_CGS(LAMBDA,T)

BB thermal radiance function cgs in Rayleight-Jeans approx
lambda in cm
T in K
rj_cgs = erg/(cm2 sec cm sterad)

4.6.4 REAL FUNCTION BBN_RJ_CGS(NU,T)

BB thermal radiance function cgs in Rayleight-Jeans approx
nu in Hz
T in K
rj_cgs = erg/(cm2 sec Hz sterad)

listfsub.pl - 0.0 - By M. Maris - 26 May 2000 -

usage listfsub.pl <<InputFile>> [<<Verbosity>>]

4.7 UNIT: ZOD_DUSTTEMPERATURE.F90

4.7.1 REAL FUNCTION TDUST(R,DTPAR)

Computes the IPD Dust Temperature as a function of R expressed in AU.

4.7.2 REAL FUNCTION SIGMATDUST(R,DTPAR)

Computes the IPD Dust Temperature Uncertainty as a function of R
R in AU

4.7.3 SUBROUTINE ZOD_DUSTTEMPERATURE_SET(THIS,T0,SIGMAT0,DELTA,SIGMADE

Sets parameters for dust temperature calculation.

4.7.4 SUBROUTINE ZOD_DUSTTEMPERATURE_NEW(THIS)

Initializes a new DustTemperature object.

4.7.5 SUBROUTINE ZOD_DUSTTEMPERATURE_DESTROY(THIS)

Destroy a dust temperature object

4.7.6 SUBROUTINE ZOD_DUSTTEMPERATURE_SHOW(THIS,UNIT,TITLE)

Shows a dust temperature object.

4.8 UNIT: ZOD_SIZEDISTRIBUTION.F90

4.8.1 SUBROUTINE SIZEDISTRIBUTION_BASE_NEW(THIS)

Initializes a new size distribution

4.8.2 SUBROUTINE SIZEDISTRIBUTION_ISHI_NEW(THIS)

Initializes a new size distribution of ISHIMOTO type

4.8.3 SUBROUTINE SIZEDISTRIBUTION_IMF_NEW(THIS)

Initializes a new size distribution of IMF type

4.9 UNIT: ZOD_DENSDSTR.F90

4.9.1 SUBROUTINE ZOD_DENSDSTR_BASE_NEW(THIS)

Initializes a new density distribution

4.9.2 SUBROUTINE ZOD_DENSDSTR_COBE_NEW(THIS)

Initializes a new density distribution of COBE type

4.9.3 SUBROUTINE ZOD_DENSDSTR_COBE_SMOOTH_NEW(THIS)

Initializes a new density distribution of COBE type

4.9.4 REAL FUNCTION ZOD_DENSITY_COBE_SMOOTH(HPOS,DISTPAR)

for an heliocentr position HPos = (X, Y, Z) (UA) computes the local 3D optical density for a given set of model parameters P

4.9.5 SUBROUTINE PCEN_AECL(HPOS,HR,LONG,LAT,DISTANCE,SLONG,SRADIUS);

Heliocentric position (AU) of the point at the spacecraft centered coordinates (long,lat,distance) from a spacecraft on the ecliptic with ecliptical longitude Slong and ecliptical eliocentric distance Sradius

4.9.6 SUBROUTINE HCEN_AECL_VEC(HPOS,HR,HSC,P,DISTANCE);

Heliocentric position (AU) of a point individuated by the spacecraft pointing vector P at distance DISTANCE from the spacecraft

4.9.7 SUBROUTINE ZOD_DENSDSTR_STARTUP()

Intialization of ZOD_DENSDSTR.
To be runned only once at start time.

4.10 UNIT: ZOD_COLORCORRECTION.F90

4.10.1 SUBROUTINE COLORCORRECTION_BASE_NEW(THIS)

Inits a New Color Correction object

4.10.2 SUBROUTINE COLORCORRECTION_COBE_NEW(THIS)

Inits a New Color Correction object of COBE Type

4.11 UNIT: ZOD_BRINT.F90

4.11.1 SUBROUTINE ZOD_BRINT_RJ_COBE_SMOOTH

Parameters: THIS, Pointing, SpaceCraft, Sun, MaxPower, MinDistance, MaxDistance, ParMod, DustTemperaturePar, IMethod, Eps, MaxNLoops, FrequencyGHz, NoQuitAfterSetUp, SetUp, FirstCall

Computes the components for the brightness integral, along a given line of sight defined by the spacecraft Pointing and the spacecraft position SPACECRAFT. Results are saved in the THIS structure of type T_BRINT_COMPONENT. Positions are in baricentric coordinates.

Black Body is approximated by the Raileight-Jeans formula in cgs units
Optional parameters shall be passed at the first call, then they are saved and reused at each next call (see the subsequent comments). An internal counter checks that all the required parameters have been passed.

4.11.2 SUBROUTINE P_DECREMENTCOUNTER()

Used to decrement the NFILLS counter

4.11.3 SUBROUTINE ZOD_BRINT_STARTUP()

Intialization of ZOD_BRINT
To be runned only once at start time

4.11.4 REAL FUNCTION P_FX_COBE_SMOOTH_DENSITY(DISTANCE)

For an heliocentric DISTANCE in AU computes the local 3D optical density
Global Variables
Parameters are passed through Private Members

4.11.5 REAL FUNCTION P_FX_COBE_SMOOTH_DENS_RJ(DISTANCE)

For an heliocentric DISTANCE in AU computes the local 3D optical density scaled by the Black Body brightness in RJ approximation.
Parameters are passed through Private Members
The RJ is in cgs units and it is assumes Lambda=1cm

4.11.6 REAL FUNCTION P_FX_COBE_SMOOTH_DENS_RJ_TN(DISTANCE)

For an heliocentric DISTANCE in AU computes the local 3D optical density scaled by the Black Body brightness in RJ approximation and T^N.
Parameters are passed through Private Members
The RJ is in cgs units and it is assumes Lambda=1cm

4.11.7 REAL FUNCTION P_FX_COBE_T_SMOOTH_DENS(DISTANCE)

For an heliocentric DISTANCE in AU computes the local 3D optical density scaled by T
Parameters are passed through Private Members

4.11.8 REAL FUNCTION P_FX_COBE_T(DISTANCE)

For an heliocentric DISTANCE in AU computes the local T of Dust
Parameters are passed through Private Members

4.11.9 SUBROUTINE P_ZOD_BRINT_NEW_BASE(THIS)

The creator for the basic ZOD_BRINT object
At creation time all the integrated quantities are set to -1
MaxPower = P_UpperMaxTPower

4.11.10 SUBROUTINE P_ZOD_BRINT_DESTROY_BASE(THIS)

The DESTRUCTOR for the basic ZOD_BRINT object

4.11.11 SUBROUTINE P_ZOD_BRINT_SHOW_BASE(THIS,UNIT)

The displayer for the basic ZOD_BRINT object

4.12 UNIT: ZOD_ENV.F90

4.12.1 SUBROUTINE ZOD_ENV_GET(UNIT)

Reads the Environment variables from a file opened in UNIT

4.12.2 SUBROUTINE ZOD_ENV_NEW(UNIT)

Inits the Environment variables to their standard values

4.12.3 SUBROUTINE ZOD_ENV_DESTROY()

Destroys the ZOD Environment

4.13 UNIT: ZOD_MAIN.F90

This is the main program of the FS_ZOD package.

4.14 UNIT: FS_ZOD_ENV.F90

This unit is used to extract the pointing list from the FS to be used by the zodiacal light calculator. It assumes that the unit FS_DIP_ENV has been loaded [AD2].

4.14.1 SUBROUTINE FS_ZOD_ENV_GET(UNIT,NAMELIST,INIT,VERBOSE)

Gets from already open UNIT the data related to Zodiacal Light environment.
If UNIT is omitted then data are stored from standard input
If NameList = T or it is omitted reads data using a fixed sequence otherwise reads data using a namelist structure named: COSMOLOGICAL_DIPOLE
If INIT is absent or INIT=.true., the FS_Dipole_Ecliptical_INIT subroutine is runned and the global variable GLOBAL_DIPOLE is filled.

If INIT = .false. data are readed but not given in output
If VERBOSE .ne. 0 then a message is shown with the main parameters
Beware: Without INIT=.true. data are readed but not passed

4.14.2 SUBROUTINE FS_ZOD_ENV_CREATE(CIRCLESIZE,ECHO_UNIT)

Initializes auxiliaries data structures for Zodiacal Light generation
It uses parameters already readed by FS_ZOD_ENV_GET
Short messages are generated, which, if required, are written in the unit ECHO_UNIT

4.14.3 SUBROUTINE FS_ZOD_ENV_DESTROY()

Kills all the allocatable data structures and close all the units associated with the FS_ZOD_ENV

4.14.4 SUBROUTINE FS_ZOD_ENV_WRITE_POINTING_LIST(TIME,IPSI,ISPIN,SUN_PO

Writes time, position. beam and velocities
Vectors are formatted according to the fs_lib_planck_soalrsystem::cel_mech subroutine convention

4.14.5 INTEGER FUNCTION FS_ZOD_ENV_CALC_ISPIN(ISPIN_COUNT)

Computes ISPIN.

PARAMETER FILES

A typical parameter file for the `zod_main.f90` code is the following:

```
&ZODIACAL_LIGHT
     FNAME_POINTLIST   = 'PointList.dat',
     FORMAT_POINTLIST  = 1,
     NDAYS             = 366,
     NCIRC_DAY         = 1,
     NSAMPLES_CIRC     = 360,
     FNAME_ZOD_RESULT  = 'zod_12mth_857ghz.dat',
     FORMAT_ZOD_RESULT = 1,
     Integration_MinDistance = 0.0d0,
     Integration_MaxDistance = 5.2d0,
     Integration_Method = 1,
     Integration_Accuracy    = -1.d-3,
     Integration_MaxNLoops   = 20,
     MaxPower          = 1,
     Insamples         = -1,
     SkipSamples       = -1,
     FrequencyGHz      = 857.,
/
```

5 OTHER USEFUL CODE

Useful ancillary code is described here.

5.1 MAKE_DOC.PL

Perl script to generates the library documentation.

5.2 MAKE_ZLE.PL

Perl script to generates just the .o and .mod files related to the dipole library, its environment and the main program.

6 THE FLIGHT SIMULATOR SIDE

This section describes how the `fs_zod.f90` module is used inside the FS. Note that to operate the flight simulator has to be copied and compiled in its integrity.

6.1 POINTING LIST INTEGRATION

The `FS_ZOD_ENV.F90` assumes that the `FS_DIP_ENV.F90` [2] is present.

To link to the flight simulator introduce the following declarations in the declaration part:

```
!!!! ZODIACAL LIGHT
!
! ZODIACAL LIGHT libraries
! By M. Maris
!
!    USE FS_ZOD
    USE FS_ZOD_ENV
```

Parameters required for the initialisation are read calling:

```
! Reads the zodiacal light related parameters from standard input
! using NameList method

        CALL FS_ZOD_ENV_GET(NAMELIST=.true.,Verbose=11)
```

The environment is then initialised using even other FS parameters:

```
!
!  ============================================================
!  Initializes the ZOD environment with the required parameters
!  ============================================================
   print*,'Initializes the SC_TAB environment with the required parameters'
   CALL FS_ZOD_ENV_CREATE(ECHO_UNIT=16,CircleSize=dimpsi+1)
```

The ispin loop is modified:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!! ZODIACAL MODIFICATION
!!!!! This was the old structure
!                                    do ispin=ispin_start,ispin_end
                                   do ispin_count=ispin_start,ispin_end
        ispin = fs_zod_env_calc_ispin(ispin_count)
```

!!! TEST ZODIACAL
vec_pos(1,0) = -1.d0
vec_pos(2,0) = -1.d0
vec_pos(3,0) = -1.d0

After the spacecraft pointing is generated by the FS the pointing is written in output:

```
call FS_ZOD_ENV_WRITE_POINTING_LIST(      &
     &  time       = dt_after_startobs_jd &
     &, ipsi       = ipsi &
     &, ispin      = ispin &
     &, planck_pos = planck_pos         &
     &, planck_vel = planck_vel         &
     &, sun_pos    = sun_pos         &
     &, sun_vel    = sun_vel         &
     &, beam       = beam_dir             &
     &)
```

Before to leave destroy the environment to close files:

```
!
!   ======================================
!   Destroys the ZODIACAL LIGHT environment
!   ======================================
    CALL FS_ZOD_ENV_DESTROY()
```

6.2 PARAMETER FILE

A typical parameter file is:

```
&ZODIACAL_LIGHT
      iWantZod                    = T,
      ISPIN_STEP                  = 24,
      ISPIN_ZERO                  = 0,
      iWantWritePointList         = T,
      Format_PointList            = 1,
      FName_PointList             = '30GHZ/PointList'
/
```