TITLE: # Planck LFI – FS_DIP: A Flight Simulator Module to Simulate the Cosmological Dipole

DOC. TYPE: OAT-TN

PROJECT REF.: PL-LFI-OAT-TN–021 **PAGE:** I of IV, 00

ISSUE/REV.: 0 **DATE:** July 20, 2001

Issued by	**Michele Maris** **LFI DPC Development Team**	**Date:** 20 Jul 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** **LFI Principal Investigator**	**Date:** **Signature:** _____

DISTRIBUTION LIST

Recipient	Company / Institute	E-mail address	Sent

CHANGE RECORD

Issue	Date	Sheet	Description of Change	Release
0	2001-07-20		Initial Release of the Documentation	0

TABLE OF CONTENTS

1 SCOPE

Description of the package FS_DIP to simulate the cosmological dipole in temperature in the framework of the PLANCK mission.

1.1 LIMITS OF APPLICABILITY

In the first release, this module has been developed to generate the cosmological dipole required for the data compression simulations. Consequently only the component induced by the solar motion has been introduced, while the term induced by the spacecraft motion relative to the Sun has not been simulated. However, this term will be introduced in a future release. Ecliptic and Equatorial coordinates are assumed to be solar system baricentric.

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

2.2 REFERENCE DOCUMENTS

2.3 ACRONYMS LIST

FS	Flight Simulator
Bsys	Blind Systematic
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

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. The library module: `fs_dipole.f90`.
2. The package environment: `fs_dipole_env.f90`.
3. The example driver used for testing: `fs_dipole_test.f90`.
4. The interface with the Flight Simulator

The library module is designed according to the principles of OOP in F90. The library module assures maximal reusability of the software. It carries the definition of the *dipole* object plus the related methods to handle it. It has to carry only general purpose methods.

The package environment holds definitions and methods required to integrate the package inside an application. This module is less standardised, 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 example driver used for testing documents how to use the dipole module and its environment and gives some simple tests to validate it.

The interface with the Flight Simulator is based on the methods used in the `fs_dipole_test.f90` and the package environment.

4 LIST OF ROUTINES AND SERVICES

Routines and services are listed and described here.

4.1 UNIT: FS_DIPOLE.F90

The library module.

4.1.1 REAL FUNCTION FS_DIPOLE_T_VECL_CALC(PECL,DIPOLE)

Given the ecliptical pointing direction Pecl, computes the dipole temperature amplitude.
DIPOLE is a keyword for a structure of type T_DIPOLE, if not set, the GLOBAL DIPOLE keyword is used instead

4.1.2 REAL FUNCTION FS_DIPOLE_T_VEQ_CALC(PEQ,DIPOLE)

Given the equatorial pointing verso Peq, computes the dipole temperature amplitude
DIPOLE is a keyword for a structure of type T_DIPOLE, if not set, the GLOBAL DIPOLE keyword is used instead

4.1.3 REAL FUNCTION FS_DIPOLE_T_ECLIPTICAL_CALC(LONG_ECL,LAT_ECL,DIPOL

Given ecliptical coordinate (long_ecl, lat_ecl) both in deg.dec, computes the dipole temperature amplitude
DIPOLE is a keyword for a structure of type T_DIPOLE, if not set, the GLOBAL DIPOLE keyword is used instead

4.1.4 REAL FUNCTION FS_DIPOLE_T_EQUATORIAL_CALC(RA,DEC,DIPOLE)

Given equatorial coordinates (ra, dec) (hour.dec, deg.dec) computes the dipole temperature amplitude
DIPOLE is a keyword for a structure of type T_DIPOLE, if not set, the GLOBAL DIPOLE keyword is used instead

4.1.5 SUBROUTINE FS_DIPOLE_ECLIPTICAL_INIT

Parameters: DIPOLE, ar_dipole, ar_dipole_sigma, dec_dipole, dec_dipole_sigma, amplitude_dipole, amplitude_dipole_sigma, epsilon

Initialises the dipole DIPOLE structure in ecliptical coordinates given:
1. alpha (hh.dec), delta (deg.dec) and amplitude (mK) of the dipole
2. epsilon : the ecliptic obliquity (deg.dec)

DIPOLE is a T_DIPOLE object, if omitted the GLOBAL_DIPOLE global variable is initialised

4.1.6 SUBROUTINE FS_DIPOLE_WRITE(DIPOLE)

This subroutine displays the content of a DIPOLE Variable. It is usefull for debug purposes.
SUBROUTINE VERS3D2POLAR(long,lat,V)
Converts the 3D versor V in a longitude, latitude couple.

4.2 UNIT: FS_DIPOLE_ENV.F90

The environment module.

4.2.1 SUBROUTINE FS_DIPOLE_ENV_GET(UNIT,NAMELIST,INIT,VERBOSE)

Gets from an already open UNIT the data related to the dipole environment.
If UNIT is omitted then data are readed 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 run 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

4.2.1.1 PARAMETERS FILES FOR THE DIPOLE ENVIRONMENT

Parameters are passed trough the following parameters file:

```
&COSMOLOGICAL_DIPOLE
    iWantDipole             = T
    Epsilon_Planck          = 23.4382508
    amplitude_dipole        = 3.343             ! original 3.343
    amplitude_dipole_sigma  = 0.016
    alpha_dipole            = 11.203333333
    alpha_dipole_sigma      = 1.33333333e-2
    delta_dipole            = -7.06
    delta_dipole_sigma      = 0.16
    iWantWriteDipole        = F
    FName_Dipole            = '100GHZ/Q12MK_ALL/dipole.bin'
/
```

values are for a specific example.

4.2.2 SUBROUTINE FS_DIPOLE_ENV_CREATE(CIRCLESIZE,ECHO_UNIT)

Initializes auxiliaries data structures for dipole generation.

It uses parameters already readed by FS_DIPOLE_ENV_GET.
Short messages are generated, which, if required, are written in the unit ECHO_UNIT.

4.2.3 SUBROUTINE FS_DIPOLE_ENV_DESTROY()

Kills all the allocatable data structures and close all the units associated with the FS_DIPOLE_ENV.

5 THE FLIGHT SIMULATOR SIDE

This section describes how the module is used inside the FS.

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

```
!!!! COSMOLOGICAL DIPOLE
!
! COSMOLOGICAL DIPOLE libraries
! By M. Maris
!
    USE FS_DIPOLE
    USE FS_DIPOLE_ENV
```

Parameters required for the initialisation are read calling:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Parameters Input Block

! Reads the dipole related parameters from standard input
! using NameList method

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

The environment is then initialised using even other FS parameters:

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

After the spacecraft pointing is generated by the FS the dipole contribution is computed:

```
!
! Computes the cosmological dipole contribution
!
! Warning:
! ========
! Since the calculation of elong_beam_deg and elat_beam_deg is unsactisfactory,
! it is replaced by another routine
!

            call VERS3D2POLAR(lat=my%elat,long=my%elong,V=beam_dir)
            v_elat(ipsi)=my%elat
            v_elon(ipsi)=my%elong
            TDipole(ipsi)=FS_DIPOLE_T_Vecl_Calc(Pecl=beam_dir)
```

After the spacecraft pointing is generated by the FS the dipole contribution is added to the signal:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Summs the cosmological dipole contribution
!
                 if (iWantDipole) totsignal = totsignal + TDipole(ipsi)
```

The simulated signal is then processed as usual.

This is to help debug or to have a separated dipole:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! If required writes out the dipole signal (once per pointing)
            if ((iWantWriteDipole).and.(iround.eq.0)) then
               print*,'Writing Dipole'
               write(IOUnit_Dipole) &
                    & (v_elon(ipsi),v_elat(ipsi)&
                    &,real(TDipole(ipsi))       &
                    &,real(tod_totsignal(ipsi)),ipsi=0,dimpsi)
            endif
```

Before to leave destroy the dipole environment to close files:

```
!
!   ===============================
!   Destroys the DIPOLE environment
!   ===============================
CALL FS_DIPOLE_ENV_DESTROY()
```

6 OTHER USEFULL CODE

6.1 MAKE_DIPOLE_DOC.PL

Perl script to generates the library documentation.

6.2 MAKE_DIPOLE_LIB.PL

Perl script to generates just the .o and .mod files related to the dipole library and envronment.

6.3 MAKE_DIPOLE_TEST.PL

Perl script to generate even the tests.