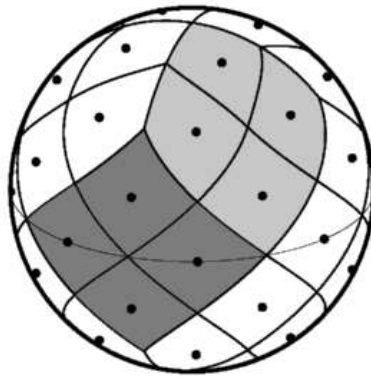


# HEALPix Fortran90 Subroutines Overview



Revision: Version 3.50; December 10, 2018

Prepared by: Eric Hivon, Hans K. Eriksen, Frode K. Hansen, Benjamin D. Wandelt, Krzysztof M. Górski, Anthony J. Banday and Martin Reinecke

Abstract: This document is an overview of the **HEALPix** Fortran90 subroutines.

<https://healpix.sourceforge.io>  
<http://healpix.sf.net>

## TABLE OF CONTENTS

Conventions . . . . .	6
Changes between releases 3.40 and 3.50 . . . . .	6
Older Changes . . . . .	6
add_card . . . . .	10
add_dipole* . . . . .	13
alm2cl* . . . . .	15
alm2map* . . . . .	18
alm2map_der* . . . . .	22
alm2map_spin* . . . . .	26
alms2fits* . . . . .	29
alter_alm* . . . . .	32
ang2vec . . . . .	35
angdist . . . . .	36
apply_mask . . . . .	38
assert, assert_alloc, assert_directory_present, assert_present, fatal_error . . .	40
brag_openmp . . . . .	42
complex_fft . . . . .	43
compute_statistics* . . . . .	44
concatnl . . . . .	46
convert_inplace* . . . . .	48
convert_nest2ring* . . . . .	50
convert_ring2nest* . . . . .	52
coordsys2euler_zyz . . . . .	54
create_alm* . . . . .	56
del_card . . . . .	60
dist2holes_nest . . . . .	62
dump_alms* . . . . .	64
fill_holes_nest . . . . .	66
fits2alms* . . . . .	68
fits2cl* . . . . .	71
gaussbeam . . . . .	73
generate_beam . . . . .	75

get_card . . . . .	77
get_healpix_data_dir, get_healpix_main_dir, get_healpix_test_dir . . . . .	79
get_healpix_pixel_weight_file, ... get_healpix_weight_file . . . . .	80
getArgument . . . . .	81
getEnvironment . . . . .	82
getdisc_ring . . . . .	83
getnumext_fits . . . . .	84
getsize_fits . . . . .	86
healpix_modules module . . . . .	89
healpix_types module . . . . .	90
in_ring . . . . .	92
input_map* . . . . .	94
input_tod* . . . . .	97
long_count, long_size . . . . .	99
map2alm* . . . . .	101
map2alm_iterative* . . . . .	104
map2alm_spin* . . . . .	109
maskborder_nest . . . . .	112
medfiltmap* . . . . .	114
median* . . . . .	116
merge_headers . . . . .	118
mpi_alm_tools* . . . . .	120
mpi_alm2map* . . . . .	122
mpi_alm2map_simple* . . . . .	124
mpi_alm2map_slave . . . . .	126
mpi_cleanup_alm_tools . . . . .	128
mpi_initialize_alm_tools . . . . .	130
mpi_map2alm* . . . . .	133
mpi_map2alm_simple* . . . . .	135
mpi_map2alm_slave . . . . .	138
nArguments . . . . .	140
neighbours_nest . . . . .	141
nest2uniq . . . . .	143
npix2nside . . . . .	145

inside2npix . . . . .	147
inside2npweights . . . . .	149
inside2ntemplates . . . . .	150
number_of_alms . . . . .	152
output_map* . . . . .	154
parse_init, parse_int, ..., parse_finish . . . . .	156
pixel_window . . . . .	160
pix2xxx,ang2xxx,vec2xxx, nest2ring,ring2nest . . . . .	162
planck_rng derived type . . . . .	165
plm_gen . . . . .	166
query_disc . . . . .	169
query_polygon . . . . .	171
query_strip . . . . .	173
query_triangle . . . . .	175
rand_gauss . . . . .	177
rand_init . . . . .	179
rand_uni . . . . .	181
read_asctab* . . . . .	183
read_bintab* . . . . .	184
read_conbintab* . . . . .	187
read_dbintab . . . . .	189
read_fits_cut4 . . . . .	191
read_par . . . . .	193
real_fft . . . . .	195
remove_dipole* . . . . .	196
ring_analysis . . . . .	199
ring_num . . . . .	201
ring_synthesis . . . . .	203
rotate_alm* . . . . .	205
same_shape_pixels_nest, same_shape_pixels_ring . . . . .	208
scan_directories . . . . .	211
size_holes_nest . . . . .	213
string, strlowercase, struppercase . . . . .	215
surface_triangle . . . . .	217

template_pixel_nest, template_pixel_ring . . . . .	219
udgrade_nest* . . . . .	222
udgrade_ring* . . . . .	225
unfold_weightsfile . . . . .	228
uniq2nest . . . . .	230
vec2ang . . . . .	232
vect_prod . . . . .	233
write_asctab* . . . . .	235
write_bintab* . . . . .	237
write_bintabh* . . . . .	239
write_dbintab . . . . .	242
write_fits_cut4 . . . . .	243
write_minimal_header . . . . .	246
write_plm . . . . .	249
xcc_v_convert . . . . .	251

## Conventions

Here we list some conventions which are used in this document.

---

*	Fortran90 allows generic names which refer to several specific subroutines. Which one of the specific routines is called depends on the type and rank of the arguments supplied in the call. We tag generic names with a * in this document.
$N_{\text{side}}$	<b>HEALPix</b> resolution parameter — see the <b>HEALPix</b> Primer.
map	We use the word “map” referring to a function, defined on the set of all <b>HEALPix</b> pixels.
$\theta$	The polar angle or colatitude on the sphere, ranging from 0 at the North Pole to $\pi$ at the South Pole.
$\phi$	The azimuthal angle on the sphere, $\phi \in [0, 2\pi[$ .

## Changes between releases 3.40 and 3.50

- correction of a bug in `map2alm_iterative`, when a `mask` is used in combination with `iter_order > 0`,
- addition of `zbounds` in `alm2map`, `alm2map_der`, `alm2map_spin` in order to simulate (faster) a signal on only a fraction of the sphere,
- introduction of `apply_mask` to apply an arbitrary mask and/or a latitude cut to a map,
- improved support for version 18 and more of Intel C and F90 compilers in `configure` script,
- edition to `fitstools.F90` allowing a proper compilation with g95.

## Older Changes

### Changes between releases 3.00 and 3.40

#### Version 3.40

- The facilities `anafast` and `smoothing` now support pixel-based quadrature weights. Introduction of the supporting `nside2npweights`, `unfold_weightsfile`, `get_healpix_weight_file`, `get_healpix_pixel_weight_file`.

- The subroutine `input_map` in its default mode test the value of the `POLCCONV` FITS keyword when reading a polarized map, and interpret the polarization accordingly, as described in the [note on POLCCONV](#) in [The HEALPix Primer](#).
- `median` subroutine: faster by moving an internal array from heap to stack; does not crash anymore when dealing with empty data sets.

## Version 3.31

- Bug correction in `input_map` routine for reading of polarized multi-HDU cut sky FITS files;
- Introduction of `winfiledir_*` and `windowfile_*` qualifiers in `alteralm` facility.

## Version 3.30

- new routines `nest2uniq` and `uniq2nest` for conversion of standard pixel index to/from Unique ID number. See "The Unique Identifier scheme" section in ["HEALPix Introduction Document"](#) for more details.
- `alm2cl` can now produces nine spectra (TT, EE, BB, TE, TB, EB, ET, BT and BE), instead of six previously, when called with two sets of polarized  $a_{\ell m}$  and can also symmetrize the output  $C(\ell)$  if requested
- the  $a_{\ell m}$  generated by `create_alm` can now take into account non-zero (exotic) TB and EB cross-spectra (option `polar=2`) if the input FITS file contains the relevant information
- addition of `asym_cl` optional keyword in `write_minimal_header` routine
- addition of `extno` optional keyword in `write_asctab` routine to write in arbitrary HDU
- improved `repeat` behavior in `write_bintabh` routine
- edited `map2alm_iterative` routine to avoid a bug specific to Intel's Ifort 15.0.2
- CFITSIO version 3.20 (August 2009) or more now required

## Version 3.20

- **HEALPix-F90** routines and facilities can now also be compiled with the free Fortran95 compiler **g95** ([www.g95.org](http://www.g95.org))
- a separate `build` directory is used to store the objects, modules, ... produced during the compilation of the source codes
- bug correction in `query_disc` for some very small discs in standard mode
- improved handling of long FITS keywords, now producing FITS files fully compatible with the **PyFITS** and **Astropy** (<https://www.astropy.org>) **Python** libraries
- improved FITS file parsing in `generate_beam`, affecting the external  $B(\ell)$  reading in the F90 facilities `alteralm`, `synfast`, `sky_ng_sim`, `smoothing`.

## Version 3.11

- `libsharp` C routines used for Spherical Harmonics Transforms and introduced in **HEALPix** 3.10 can now be compiled with any `gcc` version.
- bug correction in `query_disc` routine in `inclusive` mode
- bug correction in `alm2map_spin` routine, which had its `spin` value set to 2

## Version 3.10

- Support for `cfitsio` "Extended File Name Syntax", and usage of `libsharp` Spherical Harmonics Transform library. See ["Fortran Facilities"](#) for details.
- Faster Spherical Harmonics Transform routines thanks to `libsharp` C routines<sup>1</sup>.

---

<sup>1</sup> To *revert* to the original F90 implementation of these routines, the preprocessing variable `DONT_USE_SHARP` must be set during compilation.

## Changes up to release 3.00

### Version 3.00

- all *input* FITS files can now be compressed (with a `.gz`, `.Z`, `.z`, or `.zip` extension) and/or remotely located (with a `ftp://` or `http://` prefix). Besides, the `fits2cl` routine, used to read external beam window functions from FITS files, supports (part of) the CFITSIO [Extended File Name Syntax](#) in order to read an arbitrary extension identified by its number or its name.  
*Version 3.14 (March 2009) or newer of CFITSIO is required for HEALPix 3.00.*
- new code `process_mask` and new module `mask_tools` containing the routines `dist2holes_nest`, `fill_holes_nest`, `maskborder_nest`, `size_holes_nest` useful for mask apodization,
- improved accuracy of the co-latitude calculation in the vicinity of the poles at high resolution in `nest2ring`, `ring2nest`, `pix2ang_*`, `pix2vec_*`, ...,
- the pixel query routine `query_disc` has been improved and will return fewer false positive pixels in the inclusive mode.

### Version 2.20

- Spherical Harmonics Transform routines now transparently call `libpsht` C routines, leading to a significant (2 to 4) speed-up factor. This concerns temperature and polarized transforms (`alm2map`, `map2alm`) *without precomputation* of the  $P_{\ell m}$  as well as spin weighted (`alm2map_spin`, `map2alm_spin`) transforms for  $0 < |s| \leq 100$ , but *not* the generation of spatial derivatives (`alm2map_der`) which still uses the original F90 code. The compilation and linking to `libpsht`, now shipped with **HEALPix**, is done automatically, without any extra download or installation for the user<sup>2</sup>.
- All routines for Spherical Harmonics Transforms and most routines for pixel manipulations ( `ang2xxx`, `pix2xxx`, `vec2xxx`, ..., `nside2npix`, `npix2nside`, `nside2ntemplates`, ...) pixel queries ( `query_*`, ...) and FITS I/O (`input_map`, `output_map`, `read_bintab`, `write_bintab`, ...) of sky maps now support resolution parameters  $N_{\text{side}} > 8192$ . This means that the number of pixels and the pixel indexes can now be stored in either `integer(I4B)` or `integer(I8B)` variables (on systems supporting 64 bit variables).  
The reading and writing of  $a_{\ell m}$  containing files remains limited to  $\ell < 46340$ , though. This restriction does not apply to  $C(\ell)$  containing files.
- As a positive side effect of their upgrade, the F90 [pixel/coordinate conversion routines](#) are now up to 20% faster.
- Introduction of `long_count` and `long_size` functions.

### Version 2.14

- In `alm2map_der` routine, a numerical bug affecting the accuracy of the Stokes parameter derivatives  $\partial X / \partial \theta$ ,  $\partial^2 X / (\partial \theta \partial \phi \sin \theta)$ ,  $\partial^2 X / \partial \theta^2$ , for  $X = Q, U$  has been corrected. See ["Fortran Facilities"](#) Appendix for details.

### Versions 2.10 and 2.13

- New functions in version 2.13:
  - `get_healpix_data_dir`, `get_healpix_main_dir`, `get_healpix_test_dir` return full path to **HEALPix** directories.
- New routines in version 2.10:
  - `alm2map_spin`: synthesis of maps of arbitrary spin
  - `map2alm_iterative`: iterative analysis of map
  - `map2alm_spin`: analysis of maps of arbitrary spin
  - `healpix_modules`: meta-module
  - `write_minimal_header`: routine to write minimal FITS header
  - `parse_check_unused`: prints out parameters present in parameter file but not used by the code.

---

<sup>2</sup> To *revert* to the original F90 implementation of all these routines, the preprocessing variable `DONT_USE_PSHT` must be set during compilation.



- Improved routines:
  - `query_strip`: the `inclusive` option now returns *all* (and only) the pixels overlapping, even partially, with the strip
  - `query_disc`: when the disc center is on one of the poles, *only* the pixels overlapping with the disc are now returned.
  - `remove_dipole`: can now deal with non-uniform pixel weights.
  - `parse_init`: silent mode
  - `parse_string`: can expand environment variables (`${XXX}`) and leading `~/`

## Version 2.0

Some new features have been added

- Most routines dealing with maps and  $a_{\ell m}$  (eg. `create_alm`, `map2alm`, `alm2map`, `convert_inplace`, `convert_nest2ring`, `udgrade_nest`, `udgrade_ring`) or inputting or outputting data (`read_*`, `write_*`) now accept both single and double precision arguments.
- The routines `map2alm` and `remove_dipole` can now deal with *non-symmetric* azimuthal cut sky. For backward compatibility, the former calling sequence is still accepted.
- most routines are now parallelized with OpenMP (for shared memory architecture), and some of them are also parallelized with MPI (for distributed memory architecture)

Some new routines have been introduced since version 1.2, as listed below.

- New routines in version 2.0
  - `add_dipole`, `alm2cl`, `alm2map_der`, `fits2cl` (replaces `read_asctab`), `nside2ntemplates`, `plm_gen`, `rand_gauss`, `rand_init`, `rand_uni`, `same_shape_pixels_nest`, `same_shape_pixels_ring`, `template_pixel_nest`, `template_pixel_ring`, `write_plm` (replaces `write_dbintab`).
- New modules or modules with new name
  - **`misc_utils`**: `assert`, `assert_alloc`, `assert_directory_present`, `assert_not_present`, `assert_present`, `fatal_error`, `file_present`, `string`, `strupcase`, `strlowcase`, `upcase`, `lowcase`, `wall_clock_time`, `brag_omp`
  - **`rngmod`**: `rand_gauss`, `rand_init`, `rand_uni`
- The following routines are superseded.
  - `read_asctab` (replaced by `fits2cl`)
  - `write_dbintab` (replaced by `write_plm`)

## Version 1.2

Some new routines have been introduced since version 1.1, as listed below.

- New routines in version 1.2
  - `angdist`, `complex_fft`, `concatnl`, `del_card`, `get_card`, `getargument`, `getenvironment`, `input_tod*`, `nArguments`, `parse_double`, `parse_init`, `parse_int`, `parse_lgt`, `parse_long`, `parse_real`, `parse_string` (see `parse_xxx`), `query_disc` (replaces `getdisc_ring`), `query_polygon`, `query_strip`, `query_triangle`, `read_fits_cut4`, `real_fft`, `scan_directories`, `surface_triangle`, `vect_prod`, `write_bintab`, `write_fits_cut4`,
- New modules or modules with new name
  - the modules `extension` (C extensions), `healpix_fft` (FFT operations), `paramfile_io` (parameter parsing) have been introduced,
  - the module `wrap_fits` has been renamed `head_fits` to reflect its extended capabilities in manipulating FITS headers.
- The following routines are superseded. They have been moved to the `obsolete` module.
  - `ask_inputmap`, `ask_outputmap`, `ask_lrange` (initially in `fitstools` module)
  - `setpar`, `getpar`, `anafast_parser`, `anafast_setpar`, `anafast_getpar`, `hotspots_parser`, `hotspots_setpar`, `hotspots_getpar`, `udgrade_parser`, `udgrade_setpar`, `udgrade_getpar`, `smoothing_parser`, `smoothing_setpar`, `smoothing_getpar` (initially in `utilities` module).

Location in HEALPix directory tree: `src/f90/mod/head_fits.F90`

This routine writes a keyword of any kind into a FITS header. It is a wrapper to other routines that write keywords of different kinds.

---

**FORMAT**            call add\_card(**header**, kwd, value[, *comment*,  
*update*] )

Arguments appearing in *italic* are optional.

name & dimensionality	kind	in/out	description
header(LEN=80) DIMENSION(:)	CHR	INOUT	The header to write the key-word to.
kwd(LEN=*)	CHR	IN	the FITS keyword to write. Should be shorter or equal to 8 characters.
value	any	IN	the value (double, real, integer, logical or character string) to give to the keyword. Note that long string values (more than 68 characters in length) are supported.
<i>comment</i> (LEN=*)	CHR	IN	comment to the keyword.

---

name & dimensionality	kind in/out	description
<i>update</i>	LGT IN	if set to <code>.true.</code> , the first occurrence of the keyword <code>kwd</code> in <code>header</code> will be updated (and all other occurrences removed); otherwise, the keyword will be appended at the end (and any previous occurrence removed). If the keyword is either 'HISTORY' or 'COMMENT', <code>update</code> is ignored and the keyword is peacefully appended at the end of the header.

---



---

## EXAMPLE:

```
character(len=80), dimension(1:120) :: header
header = " ! very important
call add_card(header,'NSIDE',256,'the nside of the map')
```

Gives the keyword 'NSIDE' the value 256 in the given header-string. It is important to make sure that the `header` string array is empty before attempting to write anything in it.

---

## MODULES & ROUTINES

This section lists the modules and routines used by `add_card`.

<code>write_hl</code>	more general routine for adding a keyword to a header.
<code>cfitsio</code>	library for FITS file handling.

---

## RELATED ROUTINES

This section lists the routines related to `add_card`.

<code>write_minimal_header</code>	routine to write <b>HEALPix</b> compliant baseline FITS header
-----------------------------------	--

<code>get_card</code>	general purpose routine to read any keywords from a header in a FITS file.
<code>del_card</code>	routine to discard a keyword from a FITS header
<code>read_par, number_of_alms</code>	routines to read specific keywords from a header in a FITS file.
<code>getsize_fits</code>	function returning the size of the data set in a fits file and reading some other useful FITS keywords
<code>merge_headers</code>	routine to merge two FITS headers

---

Location in HEALPix directory tree: `src/f90/mod/pix_tools.F90`

This routine provides a means to add a monopole and dipole to a **HEALPix** map.

```

FORMAT      call add_dipole*(nside, map, ordering, degree,
                        multipoles[, fmissval])

```

Arguments appearing in *italic* are optional.

## ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	value of $N_{\text{side}}$ resolution parameter for input map
map(0:12*nside*nside-1)	SP/ DP	INOUT	<b>HEALPix</b> map to which the monopole and dipole will be added. Those are added to <i>all unflagged pixels</i> .
ordering	I4B	IN	<b>HEALPix</b> scheme 1:RING, 2:NESTED
degree	I4B	IN	multipoles to add. It is either 0 (nothing done), 1 (monopole only) or 2 (monopole and dipole)
multipoles(0:degree*degree-1)	DP	IN	values of monopole and dipole to add. The monopole is described as a scalar in the same units as the input map, the dipole as a 3D cartesian vector, in the same units.
fmissval	SP/ DP	IN	value used to flag bad pixel on input ( <b>default:</b> -1.6375e30). Pixels with that value are left unchanged.

---

**EXAMPLE:**

```
call add_dipole*(128, map, 1, 2, (\ 10.0_dp, 0.0_dp, 1.2_dp, 0.0_dp \) )
```

`map` is a **HEALPix** map of resolution  $N_{\text{side}} = 128$ , with the RING ordering scheme. A monopole of amplitude 10 and a dipole of amplitude 1.2 and directed along the  $y$  axis will be added to it.

---

**MODULES & ROUTINES**

This section lists the modules and routines used by **add\_dipole\***.

**pix\_tools**                      module, containing:

---

**RELATED ROUTINES**

This section lists the routines related to **add\_dipole\***.

**remove\_dipole**                      routine to remove the best fit monopole and monopole from a map.

---

Location in HEALPix directory tree: `src/f90/mod/alm_tools.F90`

$$C_{12}^{XY}(\ell) = \frac{1}{2\ell+1} \sum_{m=-\ell}^{\ell} a_{1,\ell m}^X a_{2,\ell m}^{Y*}, \quad (1)$$

If requested, for  $X \neq Y$ , symmetrized power spectra

$$C_{\{12\}}^{\{XY\}}(\ell) \equiv \frac{C_{12}^{XY}(\ell) + C_{12}^{YX}(\ell)}{2} = \frac{C_{12}^{XY}(\ell) + C_{21}^{XY}(\ell)}{2} \quad (2)$$

```

FORMAT      call alm2cl*(nlmax, nmmax, alm1, [alm2,] cl,
               [symmetric])

```

Arguments appearing in *italic* are optional.

## HEALPix 3.50

name & dimensionality	kind	in/out	description
nlmax	I4B	IN	the maximum $\ell$ value used for the $a_{\ell m}$ .
nmmax	I4B	IN	the maximum $m$ value used for the $a_{\ell m}$ .
alm1(1:p, 0:nlmax, 0:nmmax)	SPC/ DPC	IN	First set of $a_{\ell m}$ values. $p$ is 3 or 1 depending on whether polarisation is included or not. In the former case, the first index runs from 1 to 3 corresponding to (T,E,B).
alm2(1:p, 0:nlmax, 0:nmmax)	SPC/ DPC	IN	Second set of $a_{\ell m}$ values.
cl(0:nlmax,1:d)	SP/ DP	OUT	resulting auto or cross power spectra. If both <b>alm1</b> and <b>alm2</b> are present, <b>cl</b> will be their cross power spectrum. If only <b>alm1</b> is present, <b>cl</b> will be its power spectrum. If $d = 1$ , only the temperature spectrum $C_\ell^{TT}$ will be output. If $d = 4$ and $p = 3$ , the output will be $C_\ell^{TT}$ , $C_\ell^{EE}$ , $C_\ell^{BB}$ and $C_\ell^{TE}$ . If $d \geq 6$ and $p = 3$ , $C_\ell^{TB}$ and $C_\ell^{EB}$ will also be output, and if $d \geq 9$ , $p = 3$ , and <b>symmetric</b> is not set, $C_\ell^{ET}$ , $C_\ell^{BT}$ and $C_\ell^{BE}$ will be included.
<i>symmetric</i>	LGT	IN	If set to <b>.true.</b> when $d \geq 4$ , $p = 3$ and <b>alm2</b> is present then a symmetrized version of the cross spectra will be output in <b>cl</b> , namely $C_\ell^{TT}$ , $C_\ell^{EE}$ , $C_\ell^{BB}$ , $(C_\ell^{TE} + C_\ell^{ET})/2$ , $(C_\ell^{TB} + C_\ell^{BT})/2$ and $(C_\ell^{EB} + C_\ell^{BE})/2$ . ( <b>default:</b> <b>.false.</b> (un-symmetrized output))

---

### EXAMPLE:

```

lmax = 128 ; mmax = lmax
call alm2cl(lmax, mmax, alm1, cl_auto)
call alm2cl(lmax, mmax, alm1, alm2, cl_cross)
call alm2cl(lmax, mmax, alm1, alm2, cl_sym, symmetric=.true.)

```



`cl_auto` will contain the (auto) power spectrum of the  $a_{\ell m}$  coefficients `alm1` up to  $\ell = 128$ , `cl_cross` will be the cross power spectra of the two sets of  $a_{\ell m}$  coefficients `alm1` and `alm2`, while `cl_sym` will be a symmetrized version of `cl_cross`.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **alm2cl\***.

none

---

## RELATED ROUTINES

This section lists the routines related to **alm2cl\***.

<code>map2alm</code>	routine extracting the $a_{\ell m}$ coefficients from a <b>HEALPix</b> map
<code>create_alm</code>	routine to generate randomly distributed $a_{\ell m}$ coefficients according to a given power spectrum

---

## alm2map\*

Location in HEALPix directory tree: [src/f90/mod/alm\\_tools.F90](#)

This routine is a wrapper to 10 other routines: alm2map\_sc\_X, alm2map\_sc\_pre\_X, alm2map\_pol\_X, alm2map\_pol\_pre1\_X, alm2map\_pol\_pre2\_X, where X stands for either s or d. These routines synthesize a **HEALPix** *RING ordered* temperature map (and if specified, polarisation maps) from input  $a_{\ell m}^T$  (and if specified  $a_{\ell m}^E$  and  $a_{\ell m}^B$ ) values. The different routines are called dependent on what parameters are passed. Some routines synthesize maps with or without precomputed harmonics (note that since **HEALPix** v2.20 pre-computed harmonics most likely won't speed up computation) and some with or without polarisation. The routines accept both single and double precision arrays for alm\_TGC and map\_TQU. The precision of these arrays should match.

---

**FORMAT**                      call    alm2map\*([nsmax](#),    [nlmax](#),    [nmmax](#),  
                                 [alm\\_TGC](#), [map\\_TQU](#)[, [plm](#)=| [zbounds](#)=])

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	the $N_{\text{side}}$ value of the map to synthesize.
nlmax	I4B	IN	the maximum $\ell$ value used for the $a_{\ell m}$ .
nmmax	I4B	IN	the maximum $m$ value used for the $a_{\ell m}$ .
alm_TGC(1:p, 0:nlmax, 0:nmmax)	SPC or DPC	IN	The $a_{\ell m}$ values to make the map from. $p$ is 3 or 1 depending on whether polarisation is respectively included or not. In the former case, the first index runs from 1 to 3 corresponding to (T,E,B).

map_TQU(0:12*nsmax**2-1)	SP or DP	OUT	if only a temperature map is to be synthesized, the map-array should be passed with this rank.
map_TQU(0:12*nsmax**2-1, 1:3)	SP or DP	OUT	if both temperature and polarisation maps are to be synthesized, the map array should have this rank, where the second index is (1,2,3) corresponding to (T,Q,U).
plm(0:n_plm-1), OPTIONAL	DP	IN	If this optional matrix is passed with this rank, precomputed $P_{\ell m}(\theta)$ are used instead of recursion. (n_plm = nsmax*(nmmax+1)*(2*n_lmax-nmmax+2)).
plm(0:n_plm-1,1:3), OPTIONAL	DP	IN	If this optional matrix is passed with this rank, precomputed $P_{\ell m}(\theta)$ AND precomputed tensor harmonics are used instead of recursion. (n_plm = nsmax*(nmmax+1)*(2*n_lmax-nmmax+2)).
zbounds(1:2), OPTIONAL	DP	IN	section of the sphere on which to perform the map synthesis, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$ . If zbounds(1)<zbounds(2), it is performed <i>on</i> the strip zbounds(1)< z <zbounds(2); if not, it is performed <i>outside</i> the strip zbounds(2)≤ z ≤zbounds(1). If absent, the whole map is processed. Currently, <b>zbounds</b> and <b>plm</b> can not be used together.

---

## EXAMPLE:

```

use healpix_types
use pix_tools, only : nside2npix
use alm_tools, only : alm2map
integer(I4B) :: nside, lmax, mmax, npix

```

```

real(SP), dimension(:,:), allocatable :: map
complex(SPC), dimension(:,:,:), allocatable :: alm
real(DP), dimension(1:2) :: zrange
...
nside=256 ; lmax=512 ; mmax=lmax
npix=nside2npix(nside)
allocate(alm(1:3,0:lmax,0:mmax))
allocate(map(0:npix-1,1:3))
...
zrange =(/ 0.0_dp, 0.5_dp /)
call alm2map(nside, lmax, mmax, alm, map, zbounds=zrange)

```

Make temperature and polarisation maps from the scalar and tensor  $a_{\ell m}$  passed in alm. The maps have  $N_{\text{side}}$  of 256, and are constructed from  $a_{\ell m}$  values up to 512 in  $\ell$  and  $m$ . In order to save time, the maps are only generated on the range  $0 < z < 0.5$  (leaving the other pixels to 0) even though the input  $a_{\ell m}$  are those of a full sky map.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **alm2map\***.

<b>ring_synthesis</b>	Performs FFT over $m$ for synthesis of the rings.
compute_lam_mm, get_pixel_layout,	
gen_lamfac, gen_mfac, gen_normpol,	
gen_recfac, init_rescale, l_min_ylm	Ancillary routines used for $Y_{\ell m}$ recursion
<b>misc_utils</b>	module, containing:
<b>assert_alloc</b>	routine to print error message, when an array can not be allocated properly

Note: Starting with **version 3.10**, **libsharp** routines will be called when precomputed  $P_{\ell m}$  are not provided.

---

## RELATED ROUTINES

This section lists the routines related to **alm2map\***.

<b>alm2map_der</b>	routine generating a map and its derivatives from its $a_{\ell m}$
--------------------	--

---

<code>alm2map_spin</code>	routine generating maps of arbitrary spin from their ${}_s a_{\ell m}$
<code>smoothing</code>	executable using alm2map* to smooth maps
<code>synfast</code>	executable using alm2map* to synthesize maps.
<code>map2alm</code>	routine performing the inverse transform of alm2map*.
<code>create_alm</code>	routine to generate randomly distributed $a_{\ell m}$ coefficients according to a given power spectrum
<code>pixel_window, generate_beam</code>	return the $\ell$ -space <b>HEALPix</b> -pixel and beam window function respectively
<code>alter_alm</code>	modifies $a_{\ell m}$ to emulate effect of real space filtering

---

## alm2map\_der\*

---

Location in HEALPix directory tree: [src/f90/mod/alm\\_tools.F90](#)

This routine is a wrapper to four other routines that synthesize a **HEALPix** temperature (and polarisation) map(s), its (their) first derivatives, and optionally its (their) second derivatives. The routines accept both single and double precision arrays for alm, map, der1 and der2. The precision of these arrays should match. All maps produced are RING ordered.

See "[Fortran Facilities](#)" [Appendix](#) for a note on a bug affecting the calculation of polarisation derivatives on past versions of this routine.

---

<b>FORMAT</b>	call alm2map_der*( <a href="#">nsmax</a> , <a href="#">nlmax</a> , <a href="#">nmmax</a> , <a href="#">alm</a> , <a href="#">map</a> , <a href="#">der1</a> [, <a href="#">der2</a> ==, <a href="#">zbounds</a> ==])
---------------	---

---

## ARGUMENTS

---

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	the $N_{\text{side}}$ value of the map to synthesize.
nlmax	I4B	IN	the maximum $\ell$ value used for the $a_{\ell m}$ .
nmmax	I4B	IN	the maximum $m$ value used for the $a_{\ell m}$ .
alm(1:p, 0:nlmax, 0:nmmax)	SPC/ DPC	IN	The $a_{\ell m}$ values to make the map from. p is either 1 (temperature only) or 3 (temperature+polarisation).
map(0:12*nsmax**2-1) or (0:12*nsmax**2-1,1:3)	SP/ DP	OUT	temperature map $T(p)$ or temperature + polarisation maps $T(p)$ , $Q(p)$ , $U(p)$ to be synthesized.
der1(0:12*nsmax**2-1, 1:2*p)	SP/ DP	OUT	contains on output the first derivatives of T: $(\partial T/\partial\theta, \partial T/\partial\phi/\sin\theta)$ or the interleaved derivatives of T, Q, and U: $(\partial T/\partial\theta, \partial Q/\partial\theta, \partial U/\partial\theta; \partial T/\partial\phi/\sin\theta, \dots)$
der2(0:12*nsmax**2-1,1:3*p), OPTIONAL	SP/ DP	OUT	If this optional matrix is passed with this rank, it will contain on output the second derivatives $(\partial^2 T/\partial\theta^2, \partial^2 T/\partial\theta\partial\phi/\sin\theta, \partial^2 T/\partial\phi^2/\sin^2\theta)$ or $(\partial^2 T/\partial\theta^2, \partial^2 Q/\partial\theta^2, \partial^2 Q/\partial\theta^2, \dots)$
zbounds(1:2), OPTIONAL	DP	IN	section of the sphere on which to perform the map synthesis, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$ . If $\text{zbounds}(1) < \text{zbounds}(2)$ , it is performed <i>on</i> the strip $\text{zbounds}(1) < z < \text{zbounds}(2)$ ; if not, it is performed <i>outside</i> the strip $\text{zbounds}(2) \leq z \leq \text{zbounds}(1)$ . If absent, the whole map is processed.

---

### EXAMPLE:

```
use healpix_types
use pix_tools, only : nside2npix
```

```

use alm_tools, only : alm2map_der
integer(I4B) :: nside, lmax, mmax, npix
real(SP), dimension(:), allocatable :: map
real(SP), dimension(:,:), allocatable :: der1, der2
complex(SPC), dimension(:,:,:), allocatable :: alm
...
nside=256 ; lmax=512 ; mmax=lmax
npix=nside2npix(nside)
allocate(alm(1:1,0:lmax,0:mmax))
allocate(map(0:npix-1))
allocate(der1(0:npix-1,1:2), der2(0:npix-1,1:3))
...
call alm2map_der(nside, lmax, mmax, alm, map, der1, der2=der2)

```

Make temperature maps and its derivatives from the  $a_{\ell m}$  passed in alm. The maps have  $N_{\text{side}}$  of 256, and are constructed from  $a_{\ell m}$  values up to 512 in  $\ell$  and  $m$ .

---

## MODULES & ROUTINES

This section lists the modules and routines used by **alm2map\_der\***.

<b>ring_synthesis</b>	Performs FFT over $m$ for synthesis of the rings.
compute_lam_mm, get_pixel_layout,	
gen_lamfac_der, gen_mfac,	
gen_recfac, init_rescale, l_min_ylm	Ancillary routines used for ${}_s Y_{\ell m}$ recursion
<b>misc_utils</b>	module, containing:
<b>assert_alloc</b>	routine to print error message, when an array can not be allocated properly

---

## RELATED ROUTINES

This section lists the routines related to **alm2map\_der\***.

<b>alm2map</b>	routine generating maps of temperature and polarisation from their $a_{\ell m}$
<b>alm2map_spin</b>	routine generating maps of arbitrary spin from their ${}_s a_{\ell m}$



---

synfast	executable using alm2map_der* to synthesize maps.
create_alm	routine to generate randomly distributed $a_{\ell m}$ coefficients according to a given power spectrum

---

## alm2map\_spin\*

Location in HEALPix directory tree: **src/f90/mod/alm\_tools.F90**

This routine produces the maps of arbitrary spin  $s$  and  $-s$  given their alm coefficients. A (complex) map  $S$  of spin  $s$  is a linear combination of the spin weighted harmonics  ${}_sY_{\ell m}$

$${}_sS(p) = \sum_{\ell m} {}_s a_{\ell m} {}_s Y_{\ell m}(p) \quad (3)$$

for  $\ell \geq |m|$ ,  $\ell \geq |s|$ , and is such that  ${}_sS^* = {}_{-s}S$ .

The **usual phase convention for the spin weighted harmonics** is  ${}_sY_{\ell m}^* = (-1)^{s+m} {}_{-s}Y_{\ell -m}$  and therefore  ${}_s a_{\ell m}^* = (-1)^{s+m} {}_{-s} a_{\ell -m}$ .

alm2map\_spin\* expects the alm coefficients to be provided as

$$|s| a_{\ell m}^+ = -(|s| a_{\ell m} + (-1)^s {}_{-s} a_{\ell m})/2 \quad (4)$$

$$|s| a_{\ell m}^- = -(|s| a_{\ell m} - (-1)^s {}_{-s} a_{\ell m})/(2i) \quad (5)$$

for  $m \geq 0$ , knowing that, just as for spin 0 maps, the coefficients for  $m < 0$  are given by

$$|s| a_{\ell -m}^+ = (-1)^m |s| a_{\ell m}^{+*}, \quad (6)$$

$$|s| a_{\ell -m}^- = (-1)^m |s| a_{\ell m}^{-*}. \quad (7)$$

The two (real) maps produced by alm2map\_spin\* are defined respectively as

$$|s| S^+ = (|s| S + {}_{-s} S)/2 \quad (8)$$

$$|s| S^- = (|s| S - {}_{-s} S)/(2i). \quad (9)$$

With these definitions,  ${}_2a^+$ ,  ${}_2a^-$ ,  ${}_2S^+$  and  ${}_2S^-$  match **HEALPix** polarization  $a^E, a^B, Q$  and  $U$  respectively. However, for  $s = 0$ ,  ${}_0a_{\ell m}^+ = -a_{\ell m}^T$ ,  ${}_0a_{\ell m}^- = 0$ ,  ${}_0S^+ = T$ ,  ${}_0S^- = 0$ .

---

**FORMAT**                      call alm2map\_spin\*(**nsmax**, **nlmax**, **nmmax**,  
                                 **spin**, **alm**, **map**[, **zbounds**=])

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	the $N_{\text{side}}$ value of the map to synthesize.
nlmax	I4B	IN	the maximum $\ell$ value used for the $a_{\ell m}$ .
nmmax	I4B	IN	the maximum $m$ value used for the $a_{\ell m}$ .
spin	I4B	IN	spin $s$ of the maps to be generated (only its absolute value is relevant).
alm(1:2, 0:nlmax, 0:nmmax)	SPC/ DPC	IN	The $_{ s }a_{\ell m}^+$ and $_{ s }a_{\ell m}^-$ values to make the map from.
map(0:12*nsmax**2-1, 1:2)	SP/ DP	OUT	$_{ s }S^+$ and $_{ s }S^-$ output maps
zbounds(1:2), OPTIONAL	DP	IN	section of the sphere on which to perform the map synthesis, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$ . If $\text{zbounds}(1) < \text{zbounds}(2)$ , it is performed <i>on</i> the strip $\text{zbounds}(1) < z < \text{zbounds}(2)$ ; if not, it is performed <i>outside</i> the strip $\text{zbounds}(2) \leq z \leq \text{zbounds}(1)$ . If absent, the whole map is processed.

---

## EXAMPLE:

```

use healpix_types
use pix_tools, only : nside2npix
use alm_tools, only : alm2map_spin
integer(I4B) :: nside, lmax, mmax, npix, spin
real(SP), dimension(:,:), allocatable :: map
complex(SPC), dimension(:,:,:), allocatable :: alm
...
nside=256 ; lmax=512 ; mmax=lmax ; spin=4
npix=nside2npix(nside)
allocate(alm(1:2,0:lmax,0:mmax))
allocate(map(0:npix-1,1:2))
...
call alm2map_spin(nside, lmax, mmax, spin, alm, map)

```

Make spin-4 maps from the  $a_{\ell m}$  passed in alm. The maps have  $N_{\text{side}}$  of 256, and are constructed from  $a_{\ell m}$  values up to 512 in  $\ell$  and  $m$ .

---

## MODULES & ROUTINES

This section lists the modules and routines used by **alm2map\_spin\***.

<b>ring_synthesis</b>	Performs FFT over $m$ for synthesis of the rings.
compute_lam_mm, get_pixel_layout, gen_lamfac_der, gen_mfac, gen_mfac_spin, do_lam_lm_spin, gen_recfac, gen_recfac_spin, init_rescale, l_min_ylm	Ancillary routines used for $Y_{\ell m}$ recursion
<b>misc_utils</b>	module, containing:
<b>assert_alloc</b>	routine to print error message, when an array can not be allocated properly

Note: Starting with **version 3.10**, **libsharp** routines will be called if  $0 < |s| \leq 100$ .

---

## RELATED ROUTINES

This section lists the routines related to **alm2map\_spin\***.

<b>alm2map</b>	routine generating maps of temperature and polarisation from their $a_{\ell m}$
<b>alm2map_der</b>	routine generating maps of temperature and polarisation, and their spatial derivatives, from their $a_{\ell m}$
<b>map2alm_spin</b>	routine performing the inverse transform of alm2map.
<b>create_alm</b>	routine to generate randomly distributed $a_{\ell m}$ coefficients according to a given power spectrum

---

# alms2fits\*

Location in HEALPix directory tree: **src/f90/mod/fitstools.F90**

This routine stores  $a_{\ell m}$  values in a binary FITS file. Each FITS file extension created will contain one integer column with  $index = \ell^2 + \ell + m + 1$ , and 2 or 4 single (or double) precision columns with real/imaginary  $a_{\ell m}$  values and real/imaginary standard deviation. One can store temperature  $a_{\ell m}$  or temperature and polarisation,  $a_{\ell m}^T$ ,  $a_{\ell m}^E$  and  $a_{\ell m}^B$ . If temperature is specified, a FITS file with one extension is created. If polarisation is specified, a FITS file with 3 extensions one for each set of  $a_{\ell m}$ ,  $a_{\ell m}^T$ ,  $a_{\ell m}^E$  and  $a_{\ell m}^B$  is created.

**FORMAT**                      call alms2fits\*(filename, nalms, alms, ncl, header, nlheader, next)

## ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=filenamelen)	CHR	IN	filename for the FITS file to store the $a_{\ell m}$ in.
nalms	I4B	IN	number of $a_{\ell m}$ to store.
ncl	I4B	IN	number of columns in the FITS file. If an standard deviation is given, this number is 5, otherwise it is 3.
next	I4B	IN	the number of extensions. 1 for temperature only, 3 for temperature and polarisation.

name & dimensionality	kind	in/out	description
alms(1:nalms,1:ncl+1,1:next)	SP/ DP	IN	the $a_{\ell m}$ to write to the file. alms(i,1,j) and alms(i,2,j) contain the $\ell$ and $m$ values for the ith $a_{\ell m}$ (j=1,2,3 for (T,E,B)). alms(i,3,j) and alms(i,4,j) contain the real and imaginary value of the ith $a_{\ell m}$ . Finally, the standard deviation for the ith $a_{\ell m}$ is contained in alms(i,5,j) (real) and alms(i,6,j) (imaginary).
nlheader	I4B	IN	number of header lines to write to the file.
header(LEN=80) 1:next)	CHR	IN	the header to the FITS file.

---

### EXAMPLE:

```
call alms2fits ('alms.fits', 65*66/2, alms, 3, header, 80, 3)
```

Creates a FITS file with the  $a_{\ell m}^T$ ,  $a_{\ell m}^E$  and  $a_{\ell m}^B$  values given in alms(1:65\*66/2,1:4,1:3). The last index specifies (T,E,B). The second index gives l, m, real(  $a_{\ell m}$  ), imaginary(  $a_{\ell m}$  ) for each of the  $a_{\ell m}$ . The number 65\*66/2 is the number of  $a_{\ell m}$  values up to an  $\ell$  value of 64. 80 lines from header(1:80,1:3) is written to each extension.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **alms2fits\***.

<b>write_alms</b>	routine called by alms2fits* for each extension.
<b>fitstools</b>	module, containing:
<b>prnterror</b>	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.

---

## RELATED ROUTINES

This section lists the routines related to **alms2fits\***.

<code>fits2alms</code> , <code>read_conbintab</code>	routines to read $a_{\ell m}$ from a FITS file
<code>dump_alms</code>	has the same function as <code>alms2fits*</code> but with parameters passed differently.

---

## alter\_alm\*

Location in HEALPix directory tree: [src/f90/mod/alm\\_tools.F90](#)

This routine modifies scalar (and tensor)  $a_{\ell m}$  by multiplying them by a beam window function described by a FWHM (in the case of a gaussian beam) or read from an external file (in the more general case of a circular beam)  $a_{\ell m} \rightarrow a_{\ell m} b(\ell)$ . It can also be used to multiply the  $a_{\ell m}$  by an arbitray function of  $\ell$ .

---

**FORMAT**                    call   alter\_alm\*([nsmax](#),   [nlmax](#),   [nmmax](#),  
                              [fwhm\\_arcmin](#), [alm\\_TGC](#)[, [beam\\_file](#), [win-](#)  
                              [dow](#)])

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	$N_{\text{side}}$ resolution parameter of the map associated with the $a_{\ell m}$ considered. Currently has no effect on the routine.
nlmax	I4B	IN	maximum $\ell$ value for the $a_{\ell m}$ .
nmmax	I4B	IN	maximum $m$ value for the $a_{\ell m}$ .
fwhm_arcmin	SP/ DP	IN	fwhm size of the gaussian beam in arcminutes.
alm_TGC(1:p,0:nlmax,0:nmmax)	SPC/ DPC	INOUT	complex $a_{\ell m}$ values to be altered. The first index here runs from 1:1 for temperature only, and 1:3 for polarisation. In the latter case, 1=T, 2=E, 3=B.



---

beam_file(LEN=filenamelen) (OPTIONAL)	CHR IN	name of the file containing the (non necessarily gaussian) window function $B_\ell$ of a circular beam. If present, it will override the argument <code>fwhm_arcmin</code> .
window(0:nlw,1:d) (OPTIONAL)	SP/ IN DP	arbitrary window by which to multiply the $a_{\ell m}$ . If present, it overrides both <code>fwhm_arcmin</code> and <code>beam_file</code> . If $nlw < nlmax$ , the $a_{\ell m}$ with $\ell \in \{nlw+1, nlmax\}$ are set to 0, and a warning is issued. If $d < p$ the window for temperature is replicated for polarisation.

---

## EXAMPLE:

```
call alter_alm(64, 128, 128, 1, 5.0, alm_TGC)
```

Alters scalar and tensor  $a_{\ell m}$  of a map with  $N_{\text{side}} = 64$ ,  $\ell_{\text{max}} = m_{\text{max}} = 128$  by multiplying them by the beam window function of a gaussian beam with FWHM = 5 arcmin.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **alter\_alm\***.

<b>alm_tools</b>	module, containing:
<code>generate_beam</code>	routine to generate beam window function
<code>pixel_window</code>	routine to generate pixel window function

---

## RELATED ROUTINES

This section lists the routines related to **alter\_alm\***.

<code>create_alm</code>	Routine to create $a_{\ell m}$ coefficients.
<code>rotate_alm</code>	Routine to rotate $a_{\ell m}$ coefficients between 2 different arbitrary coordinate systems.
<code>map2alm</code>	Routines to analyze a <b>HEALPix</b> sky map into its $a_{\ell m}$ coefficients.

<code>alm2map</code>	Routines to synthesize a <b>HEALPix</b> sky map from its $a_{\ell m}$ coefficients.
<code>alms2fits, dump_alms</code>	Routines to save a set of $a_{\ell m}$ in a FITS file.

---

# ang2vec

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Routine to convert the position angles  $(\theta, \phi)$  of a point on the sphere into its 3D position vector  $(x, y, z)$  with  $x = \sin \theta \cos \phi$ ,  $y = \sin \theta \sin \phi$ ,  $z = \cos \theta$ .

**FORMAT**                      call ang2vec([theta](#), [phi](#), [vector](#))

## ARGUMENTS

name & dimensionality	kind	in/out	description
theta	DP	IN	colatitude in radians measured southward from north pole (in $[0, \pi]$ ).
phi	DP	IN	longitude in radians measured eastward (in $[0, 2\pi]$ ).
vector(3)	DP	OUT	three dimensional cartesian position vector $(x, y, z)$ normalised to unity. The north pole is $(0, 0, 1)$

## RELATED ROUTINES

This section lists the routines related to **ang2vec**.

<a href="#">angdist</a>	computes the angular distance between 2 vectors
<a href="#">vec2ang</a>	converts the 3D position vector of point into its position angles on the sphere.
<a href="#">vect_prod</a>	computes the vector product between two 3D vectors

# angdist

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Returns the angular distance in radians between two vectors. The input vectors do not have to be normalised. For almost colinear or anti-colinear vectors, renders numerically more accurate results than the  $\cos^{-1}$  of the scalar product.

---

**FORMAT**                      call angdist([v1](#), [v2](#), [dist](#))

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
v1(3)	DP	IN	cartesian vector.
v2(3)	DP	IN	cartesian vector.
dist	DP	OUT	angular distance in radians between the 2 vectors.

---

## EXAMPLE:

```
use healpix_types
use pix_tools, only : angdist
real(DP) :: dist, one = 1.0_dp
call angdist((/1,2,3/)*one, (/1,2,4/)*one, dist)
print*, dist
```

Returns the angular distance between 2 vectors.

---

## RELATED ROUTINES

This section lists the routines related to **angdist**.

[ang2vec](#)                      converts the position angles of a point on the sphere into its 3D position vector.

---

<code>vec2ang</code>	converts the 3D position vector of point into its position angles on the sphere.
<code>vect_prod</code>	computes the vector product between two 3D vectors

---

# apply\_mask

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

This routine multiplies a **HEALPix** map by an arbitrary pixel mask and/or sets to 0 pixels lying in or out a constant latitude strip

---

**FORMAT**                    call apply\_mask( [map](#), [ordering](#)[, [mask](#)=, [zbounds](#)=])

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
map(0:npix-1,1:p)	SP/ DP	INOUT	<b>HEALPix</b> map(s) on which the mask(s) and/or cut will be applied. The map will obviously be modified on output if either <a href="#">mask</a> or <a href="#">zbounds</a> are provided.
ordering	I4B	IN	<b>HEALPix</b> scheme of the map(s) (and mask(s)) 1:RING, 2: NESTED (relevant for <a href="#">zbounds</a> ).
mask(0:npix-1,1:q) OPTIONAL	SP/ DP	IN	pixel mask(s). Should have the same <a href="#">ordering</a> and number of pixels (npix) as <a href="#">map</a> . If several maps are provided ( $p > 1$ ), each map is multiplied by its respective mask, and if there are fewer masks than maps ( $q < p$ ) the last mask is duplicated as many times as necessary to match the number of maps.
zbounds(1:2) OPTIONAL	DP	IN	section of the map to be set to 0, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$ . If $\text{zbounds}(1) < \text{zbounds}(2)$ , pixels <i>outside</i> the strip $\text{zbounds}(1) < z < \text{zbounds}(2)$ are set to 0; if not, pixels <i>on</i> the strip $\text{zbounds}(2) \leq z \leq \text{zbounds}(1)$ are set to 0. If absent, the map is unchanged.

---

**EXAMPLE:**

```
s = sin(15.0_dp * DEG2RAD)
call apply_mask(map, 1, zbounds=(\ s, -s \) )
```

Will set to 0 pixels of the input (RING ordered) map lying in the strip  $|b| \leq 15^\circ$ .

---

**MODULES & ROUTINES**

This section lists the modules and routines used by **apply\_mask**.

**pix\_tools**            module, containing:

---

## assert, assert\_\_alloc, assert\_\_directory\_\_present, ...

---

Location in HEALPix directory tree: [src/f90/mod/misc\\_utils.F90](#)

The Fortran90 module `misc_utils` contains a few routines to test an assertion and return an error message if it is false.

---

### SUBROUTINES:

`call assert(test [, msg, errcode])`

if `test` is true, proceeds with normal code execution. If `test` is false, issues a standard error message (unless `msg` is provided) and stops the code execution with the status `errcode` (or 1 by default).

`call assert_alloc(status, code, array)`

if `status` is 0, proceeds with normal code execution. If not, issues an error message indicating a problem during memory allocation of `array` in program `code`, and stops the code execution.

`call assert_directory_present(directory)`

issues an error message and stops the code execution if the directory named `directory` can not be found

`call assert_not_present(filename)`

issues an error message and stops the code execution if a file with name `filename` already exists.

`call assert_present(filename)`

issues an error message and stops the code execution if the file named `filename` can not be found.

`call fatal_error([msg])`

`call fatal_error`

issue an (optional user defined) error message and stop the code execution.



---

## ARGUMENTS

name & dimensionality	kind	in/out	description
test		LGT IN	result of a logical test
msg	OPTIONAL	CHR IN	character string describing nature of error
errorcode	OPTIONAL	I4B IN	error status given to code interruption
status		I4B IN	value of the <b>stat</b> flag returned by the F90 <b>allocate</b> command
code		CHR IN	name of program or code in which allocation is made
array		CHR IN	name of array allocated
directory		CHR IN	directory name (contains a '/')
filename		CHR IN	file name

---

## EXAMPLE:

```

program my_code
use misc_utils
real, allocatable, dimension(:) :: vector
integer :: status
real :: a = -1.

allocate(vector(12345),stat=status)
call assert_alloc(status, 'my_code', 'vector')

call assert_directory_present('/home')

call assert(a > 0., 'a is NEGATIVE !!!')

end program my_code

```

Will issue a error message and stops the code if **vector** can not be allocated, will stop the code if **'/home'** is not found, and will stop the code and complain loudly about it because **a** is actually negative.

---

## brag\_openmp

---

Location in HEALPix directory tree: [src/f90/mod/misc\\_utils.F90](#)

If compiled with shared memory libraries (OpenMP), this routine prints out the number of CPUs used (controlled by the environment variable `OMP_NUM_THREADS`) and the number of CPUs available.

---

**FORMAT**                      call brag\_openmp()

---

---

### EXAMPLE:

```
use misc_utils
call brag_openmp()
```

Will print out:

```
-----
Number of OpenMP threads in use:  2
Number of CPUs available:  2
-----
on a bi-pro (or dual core) computer
```

---

# complex\_fft

Location in HEALPix directory tree: [src/f90/mod/healpix\\_fft.F90](#)

This routine performs a forward or backward Fast Fourier Transformation on its argument `data`.

---

**FORMAT**                      call complex\_fft([data](#), [backward](#))

---

## ARGUMENTS

name&dimensionality	kind	in/out	description
<code>data(:)</code>	XXX	INOUT	array containing the input and output data. It can be of type <code>real(sp)</code> , <code>real(dp)</code> , <code>complex(spc)</code> or <code>complex(dpc)</code> . If it is of type <code>real</code> , it is interpreted as an array of <code>size(data)/2</code> complex variables.
<code>backward</code>	LGT	IN	Optional argument. If present and true, perform backward transformation, else forward

---

## EXAMPLE:

```
use healpix_fft
call complex_fft (data, backward=.true.)
```

Performs a backward FFT on data.

---

## RELATED ROUTINES

This section lists the routines related to **complex\_fft**.

[real\\_fft](#)                      routine for FFT of real data

---

## compute\_statistics\*

Location in HEALPix directory tree: [src/f90/mod/statistics.f90](#)

This routine computes the min, max, absolute deviation and first four order moment of a data set

---

**FORMAT**                      call compute\_statistics\*(*data*,*stats*[, *badval*])

Arguments appearing in *italic* are optional.

---

### ARGUMENTS

name & dimensionality	kind	in/out	description
<i>data</i> (:)	SP/ DP	IN	data set $\{x_i\}$
<i>stats</i>	tstats	OUT	structure containing the statistics of the data. The respective fields ( <i>stats%field</i> ) are:
<i>ntot</i>	I8B	–	total number of data points
<i>nvalid</i>	I8B	–	number $n$ of valid data points
<i>mind</i> , <i>maxd</i>	DP	–	minimum and maximum valid data
<i>average</i>	DP	–	average of valid points $m = \sum_i x_i/n$
<i>absdev</i>	DP	–	absolute deviation $a = \sum_i  x_i - m /n$
<i>var</i>	DP	–	variance $\sigma^2 = \sum (x_i - m)^2/(n - 1)$
<i>rms</i>	DP	–	standard deviation $\sigma$
<i>skew</i>	DP	–	skewness factor $s = \sum (x_i - m)^3/(n\sigma^3)$
<i>kurt</i>	DP	–	kurtosis factor $k = \sum (x_i - m)^4/(n\sigma^4) - 3$
<i>badval</i> (OPTIONAL)	SP/ DP	IN	sentinel value given to bad data points. Data points with this value will be ignored during calculation of the statistics. If not set, all points will be considered. <b>Do not set to 0!</b>

---

### EXAMPLE:

```
use statistics, only: compute_statistics, print_statistics, tstats
type(tstats) :: stats
```

```
...  
compute_statistics(map, stats)  
print*,stats%average, stats%rms  
print_statistics(stats)
```

Computes the statistics of `map`, prints its average and *rms* and prints the whole list of statistical measures.

---

## RELATED ROUTINES

This section lists the routines related to `compute_statistics*`.

<code>median</code>	routine to compute median of a data set
---------------------	---

# concatnl

Location in HEALPix directory tree: [src/f90/mod/paramfile\\_io.F90](#)

Function to concatenate up to 10 substrings interspaced with LineFeed character. Upon printing each substring will be on a different line.

---

**FORMAT**                      var=concatnl(string1[, string2, string3, ...])

---

## ARGUMENTS

---

name & dimensionality	kind	in/out	description
string1	CHR	IN	the first substring to be concatenated.
string2	CHR	IN	the second substring (if any) to be concatenated.
		optional	
string3	CHR	IN	... up to 10 substrings can be concatenated.
		optional	
var	CHR	OUT	concatenation of the substrings interspaced with LineFeed character.

---

## EXAMPLE:

---

```
use paramfile_io
print*,concatnl('a','bbbbbbbb','C 10 3')
```

Will return:

```
a
bbbbbbbb
C 10 3
```

---

## RELATED ROUTINES

This section lists the routines related to **concatnl**.

[parse\\_XXX](#)                      parse an ASCII file for parameters definition



## convert\_inplace\*

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Routine to convert a **HEALPix** map from NESTED to RING scheme or vice versa. The conversion is done in place, meaning that it doesn't require memory for a temporary map, like the *convert\_nest2ring* or *convert\_ring2nest* routines. But for that reason, this routine is slower and not parallelized. The routine is a wrapper for 6 different routines and can therefore process integer, single precision and double precision maps as well as mono or bi dimensional arrays.

---

**FORMAT**                      call convert\_inplace\*(*subcall*, *map*)

---

### ARGUMENTS

---

name & dimensionality	kind	in/out	description
subcall	—	IN	routine to be called by convert_inplace_real. Set this to <i>ring2nest</i> or <i>nest2ring</i> dependent on whether the conversion is RING to NESTED or vice versa.
map(0:npix-1)	I4B/ SP/ DP	INOUT	mono-dimensional full sky map to be converted, the routine finds the size itself.
map(0:npix-1,1:nd)	I4B/ SP/ DP	INOUT	bi-dimensional (nd > 0) full sky map to be converted, the routine finds both dimensions itself. Processing a bidimensional map with nd > 1 should be faster than each of the nd 1D-maps consecutively.

---



**EXAMPLE:**

```
call convert_inplace(ring2nest,map)
```

Converts an map from RING to NESTED scheme.

---

**MODULES & ROUTINES**

This section lists the modules and routines used by **convert\_inplace\***.

<b>nest2ring</b>	routine to convert a NESTED pixel index to RING pixel number.
<b>ring2nest</b>	routine to convert a RING pixel index to NESTED pixel number.

---

**RELATED ROUTINES**

This section lists the routines related to **convert\_inplace\***.

<b>convert_nest2ring</b>	convert from NESTED to RING scheme using a temporary array. Requires more space than convert_inplace, but is faster.
<b>convert_ring2nest</b>	convert from RING to NESTED scheme using a temporary array. Requires more space than convert_inplace, but is faster.

---

## convert\_\_nest2ring\*

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Routine to convert a **HEALPix** map from NESTED to RING scheme.

The routine is a wrapper for 6 different routines and can therefore process integer, single precision and double precision maps as well as mono or bi dimensional arrays.

This routine is fast, and is parallelized for shared memory architecture, but requires extra memory to store a temporary map in.

---

**FORMAT**                      call convert\_\_nest2ring\*([nside](#), [map](#))

---

### ARGUMENTS

---

name & dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\text{side}}$ parameter of the map to be converted.
map(0:12*nside**2-1)	I4B/ SP/ DP	INOUT	mono-dimensional full sky map to be converted to RING scheme.
map(0:12*nside**2-1,1:nd)	I4B/ SP/ DP	INOUT	bi-dimensional full sky map to be converted to RING scheme. The routine finds the second dimension (nd) by itself. Processing a bidimensional map with $\text{nd} > 1$ should be faster than each of the nd 1D-maps consecutively.

---

### EXAMPLE:

```
call convert__nest2ring(256,map)
```

Converts an  $N_{\text{side}} = 256$  map given in array `map` from NESTED to RING scheme.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **convert\_\_nest2ring\***.

<b>nest2ring</b>	routine to convert a NESTED pixel index to RING pixel number.
------------------	---

---

## RELATED ROUTINES

This section lists the routines related to **convert\_\_nest2ring\***.

<b>convert__ring2nest</b>	convert between RING and NESTED schemes.
<b>convert__inplace</b>	convert between NESTED and RING schemes inplace. This routine is slower than <code>convert__nest2ring*</code> , but doesn't require as much memory.

---

## convert\_ring2nest\*

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Routine to convert a **HEALPix** map from RING to NESTED scheme.

The routine is a wrapper for 6 different routines and can therefore process integer, single precision and double precision maps as well as mono or bi dimensional arrays.

This routine is fast, and is parallelized for shared memory architecture, but requires extra memory to store a temporary map in.

---

**FORMAT**                      call convert\_ring2nest\*([nside](#), [map](#))

---

### ARGUMENTS

---

name & dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\text{side}}$ parameter of the map to be converted.
map(0:12*nside**2-1)	I4B/ SP/ DP	INOUT	mono-dimensional full sky map to be converted to RING scheme.
map(0:12*nside**2-1,1:nd)	I4B/ SP/ DP	INOUT	bi-dimensional full sky map to be converted to RING scheme. The routine finds the second dimension (nd) by itself. Processing a bidimensional map with $\text{nd} > 1$ should be faster than each of the nd 1D-maps consecutively.

---

### EXAMPLE:

call convert\_ring2nest(256,map)

Converts an  $N_{\text{side}} = 256$  map given in array `map` from RING to NESTED scheme.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **convert\_ring2nest\***.

<code>ring2nest</code>	routine to convert a RING pixel index to NESTED pixel number.
------------------------	---

---

## RELATED ROUTINES

This section lists the routines related to **convert\_ring2nest\***.

<code>convert_nest2ring</code>	convert between NESTED and RING schemes.
<code>convert_inplace</code>	convert between RING and NESTED schemes inplace. This routine is slower than <code>convert_ring2nest*</code> , but doesn't require as much memory.

---

## coordsys2euler\_\_zyz

Location in HEALPix directory tree: [src/f90/mod/coord\\_v\\_convert.f90](#)

This routine returns the three Euler angles  $\psi, \theta, \varphi$ , corresponding to a rotation between standard astronomical coordinate systems. These angles can then be used in `rotate_alm`

---

**FORMAT**                    call coordsys2euler\_\_zyz([iePOCH](#), [oePOCH](#), [isys](#),  
                              [osys](#), [psi](#), [theta](#), [phi](#))

---

### ARGUMENTS

name & dimension- ality	kind	in/out	description
iePOCH	DP	IN	epoch of the input astronomical coordinate system.
oePOCH	DP	IN	epoch of the output astronomical coordinate system.
isys(len=*)	CHR	IN	input coordinate system, should be one of 'E'=Ecliptic, 'G'=Galactic, 'C'/'Q'=Celestial/eQuatorial.
osys(len=*)	CHR	IN	output coordinate system, same choice as above.
psi	DP	OUT	first Euler angle: rotation $\psi$ about the z-axis.
theta	DP	OUT	second Euler angle: rotation $\theta$ about the original (unrotated) y-axis;
phi	DP	OUT	third Euler angle: rotation $\varphi$ about the original (unrotated) z-axis;

---

### EXAMPLE:

```
use coord_v_convert, only: coordsys2euler_zyz
use alm_tools, only: rotate_alm
...
call coordsys2euler_zyz(2000.0_dp, 2000.0_dp, 'E', 'G', psi, theta, phi)
```

```
call rotate_alm(64, alm_TGC, psi, theta, phi)
```

Rotate the  $a_{\ell m}$  from Ecliptic to Galactic coordinates.

---

## RELATED ROUTINES

This section lists the routines related to **coordsys2euler\_zyz**.

<code>rotate_alm</code>	apply arbitrary sky rotation to a set of $a_{\ell m}$ coefficients.
<code>xcc_v_convert</code>	rotates a 3D coordinate vector from one astronomical coordinate system to another.

---

## create\_alm\*

---

Location in HEALPix directory tree: [src/f90/mod/alm\\_tools.F90](#)

This routine generates scalar (and tensor)  $a_{\ell m}$  for a temperature (and polarisation) power spectrum read from an input FITS file. The  $a_{\ell m}$  are gaussian distributed with a zero mean, and their amplitude is multiplied with the  $\ell$ -space window function of a gaussian beam characterized by its FWHM or an arbitrary circular beam and a pixel window read from an external file.

---

**FORMAT**                      call create\_alm\*([nsmax](#), [nlmax](#), [nmmax](#), [polar](#), [filename](#), [rng\\_handle](#), [fwhm\\_arcmin](#), [alm\\_TGC](#), [header](#)[, [windowfile](#), [units](#), [beam\\_file](#)])

Arguments appearing in *italic* are optional.

---

## ARGUMENTS



name & dimensionality	kind	in/out	description
nsmax	I4B	IN	$N_{\text{side}}$ of the map to be synthesized from the $a_{\ell m}$ created by this routine.
nlmax	I4B	IN	maximum $\ell$ value to be considered (MAX=4 $N_{\text{side}}$ if <b>windowfile</b> is provided).
nmmax	I4B	IN	maximum $m$ value for the $a_{\ell m}$ .
polar	I4B	IN	<b>if set to 0</b> , only Temperature (scalar) $a_{\ell m}$ are generated using TT spectrum. <b>If set to 1</b> , 'conventional' polarization is added, based on EE, BB and TE spectra. <b>If set to 2</b> , and if the relevant information is in <b>filename</b> , polarization is generated assuming non-zero correlation of Curl (B) modes with Temperature (T) and Gradient (E) modes (TB and EB cross-spectra). Note that the <b>synfast</b> facility calls create_alm* with <b>polar=0</b> or <b>polar=1</b>
filename(LEN=filename <sub>len</sub> )	CHR	IN	name of FITS file containing power spectra in the order TT, [EE, BB, TE, [TB, EB]] (terms in brackets are optional, see <b>polar</b> )
rng_handle	<b>planck_rng</b>	INOUT	structure containing information necessary to continue a random sequence initiated <i>previously</i> with the subroutine <b>rand_init</b> . Consecutive calls to create_alm* can be made after a single invocation to <b>rand_init</b> .
fwhm_arcmin	SP/ DP	IN	FWHM size of the gaussian beam in arcminutes.
alm_TGC(1:p,0:nlmax,0:nmmax)	SPC/ DPC	OUT	complex $a_{\ell m}$ values generated from the power spectrum in the FITS file. The first index here runs from 1:1 for temperature only, and 1:3 for polarisation. In the latter case, 1=T, 2=E, 3=B.

name & dimensionality	kind	in/out	description
<code>header(LEN=80),dimension(60)</code>	CHR	OUT	part of header which will be included in the FITS-file containing the map synthesised from the $a_{\ell m}$ which <code>create_alm</code> generates.
<code>windowfile(LEN=filenamelen)</code>	CHR	IN	full filename specification of the FITS file with the pixel window function (defined for $\ell \leq 4N_{\text{side}}$ )
<code>units(LEN=80),dimension(1:)</code>	CHR	OUT	physical units of the created $a_{\ell m}$ (square-root of the input power spectrum units).
<code>beam_file(LEN=filenamelen)</code>	CHR	IN	name of the file containing the (non necessarily gaussian) window function $B_\ell$ of a circular beam. If present, it will override the argument <code>fwhm_arcmin</code> .

---

### EXAMPLE:

```

use alm_tools, only: create_alm
use rngmod, only: rand_init, planck_rng
type(planck_rng) :: rng_handle

call rand_init(rng_handle, -1)
call create_alm(64, 128, 128, 1, 'cl.fits', rng_handle, 5.0, alm_TGC, &
& header, 'data/pixel_window_n0064.fits')
```

Creates scalar and tensor  $a_{\ell m}$  from the power spectrum given in the file 'cl.fits'. The map to be created from these  $a_{\ell m}$  is assumed to have  $N_{\text{side}} = 64$ .  $C_\ell$ s from the power spectrum are used up to an  $\ell$  value of 128. Corresponding  $a_{\ell m}$  values up to  $\ell=128$  and  $m=128$  are created as gaussian distributed complex numbers. Their are drawn from a sequence of pseudo-random numbers initiated with a seed of -1. The produced  $a_{\ell m}$  are convolved with a gaussian beam of FWHM 5 arcminutes and a pixel window read from 'data/pixel\_window\_n0064.fits'. It is assumed that after the return from this routine, a map is generated from the created  $a_{\ell m}$ . For this purpose, `header` is updated with FITS format information describing the origin and history of these  $a_{\ell m}$ .

---

## MODULES & ROUTINES

This section lists the modules and routines used by **create\_alm\***.

<b>alm_tools</b>	<u>module</u> , containing:
pow2alm_units	routine to convert from power spectrum units to $a_{\ell m}$ units
generate_beam	routine to generate beam window function
pixel_window	routine to read in pixel window function
<b>utilities</b>	<u>module</u> , containing:
die_alloc	routine that prints an error message if there is not enough space for allocation of variables.
<b>fitstools</b>	<u>module</u> , containing:
fits2cl	routine to read a FITS file containing a power spectrum.
read_dbintab	routine to read a FITS-binary file containing the pixel window functions.
<b>head_fits</b>	<u>module</u> , containing:
add_card	routine to add a keyword to a FITS header.
get_card	routine to read a keyword value from FITS header.
merge_headers	routine to merge two FITS headers.
<b>rngmod</b>	<u>module</u> , containing:
rand_gauss	function which returns a gaussian distributed random number.

---

## RELATED ROUTINES

This section lists the routines related to **create\_alm\***.

rand_init	subroutine to initiate a random number sequence.
synfast	executable using create_alm* to synthesize CMB maps from a given power spectrum.
alm2map	Routine to transform a set of $a_{\ell m}$ created by create_alm* to a <b>HEALPix</b> map.
alms2fits, dump_alms	Routines to save a set of $a_{\ell m}$ in a FITS file.

---

## del\_card

Location in HEALPix directory tree: [src/f90/mod/head\\_fits.F90](#)

This routine removes one or several keywords from a FITS header.

---

**FORMAT**            call del\_card([header](#), [kwds](#))

---

### ARGUMENTS

name & dimensionality	kind	in/out	description
header(LEN=80)(1:nlheader)	CHR	INOUT	The header to remove the keyword(s) from. The routine finds out the header size.
kwds(LEN=20)(1:nkws)	CHR	IN	list of FITS keywords to remove. The routine accepts either a vector a keywords or a single one in a scalar variable
kwds(LEN=20)	CHR	IN	the one FITS keyword to remove.

---

### EXAMPLES: #1

```
call del_card(header, (/ 'NSIDE ', 'COORD ', 'ORDERING' /) )
```

Removes the keywords 'NSIDE', 'COORD' and 'ORDERING' from Header

---

### EXAMPLES: #2

```
call del_card(header, 'ORDERING' )
```

Removes the keyword 'ORDERING' from Header

---

## MODULES & ROUTINES

This section lists the modules and routines used by **del\_card**.

<code>write_hl</code>	more general routine for adding a keyword to a header.
<code>cfitsio</code>	library for FITS file handling.

---

## RELATED ROUTINES

This section lists the routines related to **del\_card**.

<code>add_card</code>	general purpose routine to write any keywords into a FITS file header
<code>get_card</code>	general purpose routine to read any keywords from a header in a FITS file.
<code>read_par, number_of_alms</code>	routines to read specific keywords from a header in a FITS file.
<code>getsize_fits</code>	function returning the size of the data set in a fits file and reading some other useful FITS keywords
<code>merge_headers</code>	routine to merge two FITS headers

---

## dist2holes\_\_nest

Location in HEALPix directory tree: [src/f90/mod/mask\\_tools.F90](#)

For a input binary mask in NESTED ordering, `dist2holes__nest` returns the angular distance (in radians) from each *valid* (1-valued) pixel to the closest *invalid* (0-valued) pixel. Distances are measured between pixel centers.

---

**FORMAT**                      call `dist2holes__nest`([nside](#), [mask](#), [distance](#))

---

### ARGUMENTS

name & dimensionality	kind	in/out	description
<code>nside</code>	I4B	IN	the $N_{\text{side}}$ value of the input mask.
<code>mask(0:Npix-1)</code>	I4B	IN	Input NESTED-ordered mask. $N_{\text{pix}} = 12 * n_{\text{side}} * n_{\text{side}}$
<code>distance(0:Npix-1)</code>	DP	OUT	Output NESTED-ordered angular-distance map

### EXAMPLE:

```
use healpix_types
use healpix_modules
...
call dist2holes__nest(nside, mask, distance)
```

???

---

### MODULES & ROUTINES

This section lists the modules and routines used by **dist2holes\_\_nest**.

<b>mask__tools</b>	mask processing module (see related routines below)
--------------------	---

---

## RELATED ROUTINES

This section lists the routines related to **dist2holes\_\_nest**.

<code>dist2holes__nest</code>	angular distance to closest invalid pixel of the given mask
<code>fill_holes__nest</code>	turn to <i>valid</i> all pixels located in 'holes' containing fewer pixels than the given threshold
<code>maskborder__nest</code>	identify inner boundary pixels of 'holes' for given mask
<code>size_holes__nest</code>	returns size (in pixels) of holes found in input mask

---

## dump\_alms\*

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine stores  $a_{\ell m}$  values in a binary FITS file. The FITS file created will contain one integer column with  $index = \ell^2 + \ell + m + 1$  and 2 single precision columns with real/imaginary  $a_{\ell m}$  values. One can store temperature  $a_{\ell m}$  or polarisation,  $a_{\ell m}^E$  or  $a_{\ell m}^B$ . If temperature is specified, a FITS file is created. If polarisation is specified, an old FITS file is opened and extra extensions is created.

---

**FORMAT**                      call   dump\_alms\*(filename,   alms,   nlmax,  
                                 header, nlheader, extno)

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=filenamelen)	CHR	IN	filename for the FITS-file to store the $a_{\ell m}$ in.
nlmax	I4B	IN	maximum $\ell$ value to store.
alms(0:nlmax,0:nlmax)	SPC/ DPC	IN	array with $a_{\ell m}$ , in the format used by eg. <a href="#">map2alm</a> , so <a href="#">alms(1,m)</a> corresponds to $a_{\ell m}$
extno	I4B	IN	extension number. If 0 is specified, a FITS file is created and $a_{\ell m}$ is stored in the first FITS extension as temperature $a_{\ell m}$ . If 1 or 2 is specified, an already existing file is opened and a 2nd or 3rd extension is created, treating $a_{\ell m}$ as $a_{\ell m}^E$ or $a_{\ell m}^B$ .
nlheader	I4B	IN	number of header lines to write to the file.
header(LEN=80) (1:nlheader)	CHR	IN	the header to the FITS-file.

---



**EXAMPLE:**

```
call dump_alms ('alms.fits', alms, 64, header, 100, 1)
```

Opens an already existing FITS file which contains temperature  $a_{\ell m}$ . An extra extension is added to the file where the  $a_{\ell m}$  array are written in a three-column format as described above. 100 header lines are written to the file from the array header(1:80).

---

**MODULES & ROUTINES**

This section lists the modules and routines used by **dump\_alms\***.

<b>fitstools</b>	module, containing:
<b>printerror</b>	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.

---

**RELATED ROUTINES**

This section lists the routines related to **dump\_alms\***.

<b>fits2alms</b> , <b>read_conbintab</b>	routines to read $a_{\ell m}$ from a FITS-file
<b>alms2fits</b>	has the same function as dump_alms* but is more general.

---

## fill\_holes\_nest

Location in HEALPix directory tree: [src/f90/mod/mask\\_tools.F90](#)

For a input binary mask in NESTED ordering, fill\_holes\_nest flip to 1 all pixels located in invalid regions with fewer pixels than the threshold provided.

Two pixels are adjacent (and belong to the same region or hole) if they have at least one point in common.

---

**FORMAT**                    call    fill\_holes\_nest([nside](#),    [new\\_min\\_size](#),  
                              [mask\\_in](#), [mask\\_out](#))

---

### ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	The $N_{\text{side}}$ value of the input mask.
new_min_size	I4B	IN	Minimal size of hole (in pixels) on output
mask_in(0:Npix-1)	I4B	IN	Input NESTED-ordered mask. Npix = 12*nside*nside
mask_out(0:Npix-1)	I4B	OUT	Output NESTED-ordered mask. Can be the same array as mask_in.

---

### EXAMPLE:

```
use healpix_types
use healpix_modules
...
call fill_holes_nest(nside, new_min_size, mask_in, mask_in)
```

???

---

### MODULES & ROUTINES

This section lists the modules and routines used by **fill\_holes\_nest**.

<b>mask_tools</b>	mask processing module (see related routines below)
-------------------	---

---

## RELATED ROUTINES

This section lists the routines related to **fill\_holes\_nest**.

<b>dist2holes_nest</b>	angular distance to closest invalid pixel of the given mask
<b>fill_holes_nest</b>	turn to <i>valid</i> all pixels located in 'holes' containing fewer pixels than the given threshold
<b>maskborder_nest</b>	identify inner boundary pixels of 'holes' for given mask
<b>size_holes_nest</b>	returns size (in pixels) of holes found in input mask

---

## fits2alms\*

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine reads  $a_{\ell m}$  values from a binary FITS file. Each FITS file extension is supposed to contain one integer column with  $index = \ell^2 + \ell + m + 1$  and 2 or 4 single (or double) precision columns with real/imaginary  $a_{\ell m}$  values and real/imaginary standard deviation. One can read temperature  $a_{\ell m}$  or temperature and polarisation,  $a_{\ell m}^T$ ,  $a_{\ell m}^E$  and  $a_{\ell m}^B$ .

---

**FORMAT**                      call fits2alms\*([filename](#), [nalms](#), [alms](#), [ncl](#),  
                                  [header](#), [nlheader](#), [next](#))

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=filenamelen)	CHR	IN	filename of the FITS-file to read the $a_{\ell m}$ from.
nalms	I4B	IN	number of $a_{\ell m}$ to read.
ncl	I4B	IN	number of columns to read in the FITS file. If an standard deviation is to be read, this number is 5, otherwise it is 3.
next	I4B	IN	the number of extensions to read. 1 for temperature only, 3 for temperature and polarisation.

---

alms(1:nalms,1:(ncl+1),1:next)	SP/ DP	OUT	the $a_{\ell m}$ to read from the file. alms(i,1,j) and alms(i,2,j) contain the $\ell$ and $m$ values for the $i$ th $a_{\ell m}$ ( $j=1,2,3$ for (T,E,B)). alms(i,3,j) and alms(i,4,j) contain the real and imaginary value of the $i$ th $a_{\ell m}$ . Finally, the standard deviation for the $i$ th $a_{\ell m}$ is contained in alms(i,5,j) (real) and alms(i,6,j) (imaginary).
nlheader	I4B	IN	number of header lines to read from the file.
header(LEN=80) 1:next)	(1:nlheader,	CHR OUT	the header(s) read from the FITS-file.

---

## EXAMPLE:

```
call fits2alms ('alms.fits', 65*66/2, alms, 3, header, 80, 3)
```

Reads a FITS file with the  $a_{\ell m}^T$ ,  $a_{\ell m}^E$  and  $a_{\ell m}^B$  values read into alms(1:65\*66/2,1:4,1:3). The last index specifies (T,E,B). The second index gives  $\ell$ ,  $m$ , real(  $a_{\ell m}$  ), imaginary(  $a_{\ell m}$  ) for each of the  $a_{\ell m}$ . The number 65\*66/2 is the number of  $a_{\ell m}$  values up to an  $\ell$  value of 64. 80 lines is read from the header in each extension and returned in header(1:80,1:3).

---

## MODULES & ROUTINES

This section lists the modules and routines used by **fits2alms\***.

read_alms	routine called by fits2alms* for each extension.
<b>fitstools</b>	module, containing:
prnterror	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.

---

## RELATED ROUTINES

This section lists the routines related to **fits2alms\***.

<code>alms2fits</code> , <code>dump_alms</code>	routines to store $a_{\ell m}$ in a FITS-file
<code>read_conbintab</code>	has the same function as <code>fits2alms*</code> but with parameters passed differently.
<code>number_of_alms</code> , <code>getsize_fits</code>	can be used to find out the number of $a_{\ell m}$ available in the file.

---

## fits2cl\*

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine reads a power spectrum or beam window function from a FITS ASCII or binary table. The routine can read temperature coefficients  $C_\ell^{TT}$  or both temperature and polarisation coefficients  $C_\ell^{TT}$ ,  $C_\ell^{EE}$ ,  $C_\ell^{BB}$ ,  $C_\ell^{TE}$  (and  $C_\ell^{TB}$ ,  $C_\ell^{EB}$ ,  $C_\ell^{ET}$ ,  $C_\ell^{BT}$ ,  $C_\ell^{BE}$  when applicable). If the keyword PDM-TYPE is found in the header, fits2cl assumes the table to be in the special format used by *Planck* and will ignore the first data column. If the input FITS file contains several extensions or HDUs, the one to be read can be specified thanks to the CFITSIO [Extended File Name Syntax](#), using its number (eg, `file.fits[2]` or `file.fits+2`) or its EXTNAME value (eg, `file.fits[beam_100x100]`). By default, only the first valid extension will be read.

---

**FORMAT**                      call fits2cl\*([filename](#), [clin](#), [lmax](#), [ncl](#), [header](#),  
                                  [\[units\]](#))

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=filenamelen)	CHR	IN	the FITS file containing the power spectrum.
lmax	I4B	IN	Maximum $\ell$ value to be read.
ncl	I4B	IN	1 for temperature coefficients only, 4 for polarisation.
clin(0:lmax,1:ncl)	SP/ DP	OUT	the power spectrum read from the file.
header(LEN=80) (1:)	CHR	OUT	the header read from the FITS-file.
units(LEN=80) (1:)	CHR	OUT	the column units read from the FITS-file.

---

**EXAMPLE:**

```

use healpix_modules
real(SP), allocatable, dimension(:, :) :: cl
character(len=80), dimension(1:300) :: header
character(len=80), dimension(1:100) :: units
integer(I4B) :: lmax, ncl, np
character(len=filenamelen) :: fitsfile='cl.fits'
np = getsize_fits(fitsfile, nmaps=ncl, mlpol=lmax)
allocate(cl(0:lmax, 1:ncl))
call fits2cl(fitsfile, cl, lmax, ncl, header, units)

```

Reads a power spectrum from the FITS file 'cl.fits' and stores the result in `cl(0:lmax, 1:ncl)` which are the `ncl`  $C_\ell$  coefficients up to  $\ell = \text{lmax}$ . The FITS header is returned in `header`, the column units in `units`.

---

**MODULES & ROUTINES**

This section lists the modules and routines used by `fits2cl*`.

<code>fitstools</code>	module, containing:
<code>prnterror</code>	routine for printing FITS error messages.
<code>cfitsio</code>	library for FITS file handling.

---

**RELATED ROUTINES**

This section lists the routines related to `fits2cl*`.

<code>create_alm</code>	Routine to create $a_{\ell m}$ values from an input power spectrum.
<code>write_asctab</code>	Routine to create an ascii FITS file containing a power spectrum.
<code>getsize_fits</code>	Routine to parse FITS file header, and determine the data storage features.
<code>getnumext_fits</code>	Routine to determine number of extensions of a FITS file.

---



# gaussbeam

Location in HEALPix directory tree: [src/f90/mod/alm\\_tools.F90](#)

This routine generates the beam window function in multipole space of a gaussian beam parametrized by its FWHM. The polarization beam is also provided assuming a perfectly co-polarized beam (eg, Challinor et al 2000, [astro-ph/0008228](#))

---

**FORMAT**                      call gaussbeam([fwhm\\_arcmin](#), [lmax](#), [beam](#))

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
fwhm_arcmin	DP	IN	FWHM of the gaussian beam in arcminutes.
lmax	I4B	IN	maximum $\ell$ value of the window function.
beam(0:lmax,1:p)	DP	OUT	beam window function generated. The second index runs from 1:1 for temperature only, and 1:3 for polarisation. In the latter case, 1=T, 2=E, 3=B.

---

## EXAMPLE:

```
call gaussbeam(5.0_dp, 1024, beam)
```

Generates the window function of a gaussian beam of FWHM = 5 arcmin, for  $\ell \leq 1024$ .

---

## RELATED ROUTINES

This section lists the routines related to **gaussbeam**.

[generate\\_beam](#)                      Routine returning a beam window function.

`pixel_window`

Routine returning a pixel window function.



# generate\_beam

Location in HEALPix directory tree: [src/f90/mod/alm\\_tools.F90](#)

This routine generates the beam window function in multipole space. It is either a gaussian parametrized by its FWHM in arcmin in real space, or it is read from an external file.

---

**FORMAT**                      call    generate\_beam([fwhm\\_arcmin](#),    [lmax](#),  
                                 [beam](#)[, [beam\\_file](#)])

Arguments appearing in *italic* are optional.

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
fwhm_arcmin	DP	IN	fwhm size of the gaussian beam in arcminutes.
lmax	I4B	IN	maximum $\ell$ value of the window function.
beam(0:lmax,1:p)	DP	OUT	beam window function generated. The second index runs form 1:1 for temperature only, and 1:3 for polarisation. In the latter case, 1=T, 2=E, 3=B.
<i>beam_file</i> (LEN=filenamelen) (OPTIONAL)	CHR	IN	name of the file containing the (non necessarily gaussian) window function $B_\ell$ of a circular beam. If present, it will override the argument <a href="#">fwhm_arcmin</a> . If fewer columns than requested are found in the file, missing columns will duplicate the existing ones (based on the assumption that $B_\ell$ is the same in T, E and B). Supports the <code>fitsio</code> 'Extended Filename Syntax' (see <a href="#">examples below</a> ).

---

**EXAMPLE:**

```

use healpix_modules
real(dp), dimension(0:1024, 1:3) :: gb0, b1, b2, b3
call generate_beam(5.0_dp, 1024, gb0)
call generate_beam(0_dp, 1024, b1, beam_file='file.fits')
call generate_beam(0_dp, 1024, b2, beam_file='file.fits[col 1]')
call generate_beam(0_dp, 1024, b3, beam_file='file.fits[col 1; 2=0; 3=0]')

```

gb0 will contain the window function of a gaussian beam of FWHM = 5 arcmin, for  $\ell \leq 1024$ .

b1 will contain the first 3 columns (if available) of file.fits. If the file contains only two columns, then  $b1(:,3) = b1(:,2)$ , and if it contains a single column, then  $b1(:,3) = b1(:,2) = b1(:,1)$ .

b2 will be based on a virtual FITS file containg only the first column of file.fits, and we will have  $b2(:,3) = b2(:,2) = b2(:,1)$ .

Finally b3 will read a virtual FITS file in which the first column is the same as in file.fits, while the columns 2 and 3 are set to 0. Therefore  $b3(:,3) = b3(:,2) = 0$ .

---

**MODULES & ROUTINES**

This section lists the modules and routines used by **generate\_beam**.

<b>alm_tools</b>	module, containing:
<b>gaussbeam</b>	routine to generate a gaussian beam

---

**RELATED ROUTINES**

This section lists the routines related to **generate\_beam**.

<b>create_alm</b>	Routine to create $a_{\ell m}$ coefficients using generate_beam.
<b>alter_alm</b>	Routine to alter $a_{\ell m}$ coefficients using generate_beam.
<b>pixel_window</b>	Routine returning a pixel window function.

---

# get\_card

Location in HEALPix directory tree: [src/f90/mod/head\\_fits.F90](#)

This routine reads a keyword of any kind from a FITS header. It is a wrapper to other routines that read keywords of different kinds.

---

**FORMAT**                      call get\_card([header](#), [kwd](#), [value](#), [comment](#))

---

## ARGUMENTS

---

name & dimensionality	kind	in/out	description
header(LEN=80) DIMENSION(:)	CHR	IN	The header to read the keyword from.
kwd(LEN=8)	CHR	IN	the FITS keyword to read (NOT case sensitive).
value	any	OUT	the value read for the keyword. The type of the fortran variable 'value' (double, real, integer, logical or character) should match the type under which the value is written in the FITS file, except if 'value' is a character string, in which case it can read any keyword value, or if 'value' is real or double, in which case it can read any numerical value. Note that long string values (more than 68 characters in length) are supported.
comment(LEN=*)	CHR	OUT	comment read for the keyword.

---

## EXAMPLE:

```
call get_card(header,'NSIDE',nside,comment)
```

if `nside` is defined as an integer, it will contain on output the value of `NSIDE` (say 256) found in header

---

### EXAMPLE:

```
call get_card(header,'ORDERING',ordering,comment)
```

if `ordering` is defined as an character string, it will contain on output the value of `ORDERING` (say 'RING') found in header

---

## MODULES & ROUTINES

This section lists the modules and routines used by `get_card`.

<code>cfitsio</code>	library for FITS file handling.
----------------------	---------------------------------

---

## RELATED ROUTINES

This section lists the routines related to `get_card`.

<code>add_card</code>	general purpose routine to write any keywords into a FITS file header
<code>del_card</code>	routine to discard a keyword from a FITS header
<code>read_par, number_of_alms</code>	routines to read specific keywords from a header in a FITS file.
<code>getsize_fits</code>	function returning the size of the data set in a fits file and reading some other useful FITS keywords
<code>merge_headers</code>	routine to merge two FITS headers

---

## get\_healpix\_main\_dir, ...

---

Location in HEALPix directory tree: **src/f90/mod/paramfile\_io.F90**

A few functions are available to return the full path to **HEALPix** main directory and its **data** and **test** subdirectories. This allow those paths to be controlled by preprocessing macros or environment variables in case of non-standard installation of the **HEALPix** directory structure.

---

### FUNCTIONS:

`hmd = get_healpix_main_dir()`

returns the full path to the main **HEALPix** directory. It will be determined, in this order, from the value of the preprocessing macros **HEALPIX** and **HEALPIXDIR** if they are defined or the environment variable **\$HEALPIX** otherwise.

`hdd = get_healpix_data_dir()`

returns the full path to **HEALPix data** subdirectory. It will be determined from the preprocessing macro **HEALPIXDATA** or the environment variable **\$HEALPIXDATA**. If both fail, it will return the list of directories `{. ../data ./data .. $HEALPIX $HEALPIX/data $HEALPIX/../data $HEALPIX\data}` separated by LineFeed.

`htd = get_healpix_test_dir()`

returns the full path to **HEALPix test** subdirectory. It will be determined, in this order, from the preprocessing macro **HEALPIXTEST**, the environment variable **\$HEALPIXTEST** or **\$HEALPIX/test**.

## get\_healpix\_weight\_file, ...

---

Location in HEALPix directory tree: [src/f90/mod/paramfile\\_io.F90](#)

Functions are available to return the standardized name of pre-generated FITS files required by some **HEALPix** facilities and subroutines. Those files are available in the **data** subdirectory, whose full path is returned by companion functions such as [get\\_healpix\\_data\\_dir](#).

---

### FUNCTIONS:

`winfile = get_healpix_pixel_window_file(nside)`

returns, for the `nside` provided, the name of the FITS file containing the window function associated with the **HEALPix** pixel (of the form `pixel_window_n*.fits`).

`w8file = get_healpix_weight_file(nside, won)`

returns the name of the FITS file containing the ring-based or pixel-based weights for the given **HEALPix** resolution parameter `nside`. If `won = 1`, the output of [get\\_healpix\\_ring\\_weight\\_file\(nside\)](#) is returned, while if `won = 2`, it is the output of [get\\_healpix\\_pixel\\_weight\\_file\(nside\)](#). `won = 0` will return an empty string, while other choices of `won` will result in errors.

`w8rfile = get_healpix_ring_weight_file(nside)`

returns, for the `nside` provided, the name of the FITS file containing the ring-based weights (of the form `weight_ring_n*.fits`).

`w8pfile = get_healpix_pixel_weight_file(nside)`

returns, for the `nside` provided, the name of the FITS file containing the pixel-based weights (of the form `weight_pixel_n*.fits`). Some of them (for power of 2 `nside` in [16, 2048]) are located in the **data** directory, the other ones can be generated with the `compute_weights` C++ facility.

---



---

# getArgument

---

Location in HEALPix directory tree: [src/f90/mod/extension.F90](#)

This subroutine emulates the C routine `getarg`, which returns the value of a given command line argument.

---

**FORMAT**                      call `getArgument(index, value)`

---

---

## ARGUMENTS

---

name & dimensionality	kind	in/out	description
index	I4B	IN	index of the command line argument (where the first argument has index 1)
value	CHR	OUT	value of the argument

---

---

## RELATED ROUTINES

This section lists the routines related to `getArgument`.

<a href="#">getEnvironment</a>	returns value of environment variable
<a href="#">nArguments</a>	returns number of command line arguments

---

# getEnvironment

Location in HEALPix directory tree: **src/f90/mod/extension.F90**

This subroutine emulates the C routine `getenv`, which returns the value of an environment variable.

---

**FORMAT**                      call getEnvironment(**name**, **value**)

---

## ARGUMENTS

---

name & dimensionality	kind	in/out	description
name	CHR	IN	name of the environment variable
value	CHR	OUT	value of the environment variable

---

## EXAMPLE:

```
use extension
character(len=128) :: healpixdir
call getEnvironment('HEALPIX', healpixdir)
print*,healpixdir
```

Will return the value of the `$HEALPIX` system variable (if it is defined)

---

## RELATED ROUTINES

This section lists the routines related to `getEnvironment`.

<b>getArgument</b>	returns list of command line arguments
<b>nArguments</b>	returns number of command line arguments

---

## getdisc\_ring

---

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

This routine is obsolete, use [query\\_disc](#) instead

---

## getnumext\_\_fits

Location in HEALPix directory tree: **src/f90/mod/fitstools.F90**

This routine returns the number of extensions present in a given FITS file.

---

**FORMAT**                      var=getnumext\_\_fits(**filename**)

---

### ARGUMENTS

name & dimensionality	kind	in/out	description
var	I4B	OUT	number of extensions in the FITS file (excluding the primary unit). According to the current format, <b>HEALPix</b> files have at least one extension.
filename(LEN=filename <sub>len</sub> )	CHR	IN	filename of the FITS file.

---

**EXAMPLE:**

```
next = getnumext_fits('map.fits')
```

Returns in **next** the number of extensions present in the FITS file 'map.fits'.

---

**MODULES & ROUTINES**

This section lists the modules and routines used by **getnumext\_fits**.

<b>fitstools</b>	module, containing:
<b>printerror</b>	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.

---

**RELATED ROUTINES**

This section lists the routines related to **getnumext\_fits**.

<b>getsize_fits</b>	routine returning the number of data points in a FITS file, as well as much more information on the file.
<b>input_map</b>	routine to read a <b>HEALPix</b> FITS file

---

## getsize\_fits

---

Location in HEALPix directory tree: **src/f90/mod/fitstools.F90**

This routine reads the number of maps and/or the pixel ordering of a FITS file containing a **HEALPix** map.

---

**FORMAT**                      var=getsize\_fits(*filename* [, *nmaps*, *ordering*,  
                                  *obs\_npix*, *nside*, *mlpol*, *type*, *polarisation*,  
                                  *fwhm\_arcmin*, *beam\_leg*, *coordsys*, *poleconv*,  
                                  *extno*])

Arguments appearing in *italic* are optional.

---

## ARGUMENTS

name & dim.	kind	in/out	description
var	I8B	OUT	number of pixels or time samples in the chosen extension of the FITS file
filename(LEN=*)	CHR	IN	filename of the FITS-file containing <b>HEALPix</b> map(s).

name & dim.	kind	in/out	description
<i>nmaps</i> (OPTIONAL)	I4B	OUT	number of maps in the extension.
<i>ordering</i> (OPTIONAL)	I4B	OUT	pixel ordering, 0=unknown, 1=RING, 2=NESTED
<i>obs_npix</i> (OPTIONAL)	I4B	OUT	number of non blank pixels. It is set to -1 if it can not be determined from header information alone
<i>nside</i> (OPTIONAL)	I4B	OUT	Healpix resolution parameter Nside. Returns a negative value if not found.
<i>mlpol</i> (OPTIONAL)	I4B	OUT	maximum multipole used to generate the map (for simulated map). Returns a negative value if not found.
<i>type</i> (OPTIONAL)	I4B	OUT	Healpix/FITS file type <0 : file not found, or not valid 0 : image only fits file, deprecated Healpix format (var = 12 * nside * nside) 1 : ascii table, generally used for C(1) storage 2 : binary table : with implicit pixel indexing (full sky) (var = 12 * nside * nside) 3 : binary table : with explicit pixel indexing (generally cut sky) (var ≤ 12 * nside * nside) 999 : unable to determine the type
<i>polarisation</i> (OPTIONAL)	I4B	OUT	presence of polarisation data in the file <0 : can not find out 0 : no polarisation 1 : contains polarisation (Q,U or G,C)
<i>fwhm_arcmin</i> (OPTIONAL)	DP	OUT	returns the beam FWHM read from FITS header, translated from Deg (hopefully) to arcmin. Returns a negative value if not found.
<i>beam_leg</i> (LEN=*) (OPTIONAL)	CHR	OUT	filename of beam or filtering window function applied to data (FITS keyword BEAM_LEG). Returns a empty string if not found.
<i>coordsys</i> (LEN=20) (OPTIONAL)	CHR	OUT	string describing the pixelation astrophysical coordinates. 'G' = Galactic, 'E' = ecliptic, 'C' = celestial = equatorial. Returns a empty string if not found.
<i>polconv</i> (OPTIONAL)	I4B	OUT	polarisation coordinate convention (see Healpix primer for details) 0=unknown, 1=COSMO, 2=IAU, 3=neither COSMO nor IAU
<i>extno</i> (OPTIONAL)	I4B	IN	extension number (0 based) for which information is provided. Default = 0 (first extension).

---

**EXAMPLE:**

```
npix= getsize_fits('map.fits', nmaps=nmaps, ordering=ordering,
obs_npix=obs_npix, nside=nside, mlpol=mlpol, type=type,
polarisation=polarisation)
```

Returns 1 or 3 in nmaps, dependent on whether 'map.fits' contain only temperature or both temperature and polarisation maps. The pixel ordering number is found by reading the keyword ORDERING in the FITS file. If this keyword does not exist, 0 is returned.

---

**MODULES & ROUTINES**

This section lists the modules and routines used by **getsize\_\_fits**.

<b>fitstools</b>	module, containing:
<b>printerror</b>	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.

---

**RELATED ROUTINES**

This section lists the routines related to **getsize\_\_fits**.

<b>getnumext__fits</b>	routine returning the number of extension in a FITS file
<b>input__map</b>	routine to read a <b>HEALPix</b> FITS file

---



# healpix\_modules

---

Location in HEALPix directory tree: [src/f90/mod/healpix\\_modules.f90](#)

This module is a meta module containing most of the HEALPix modules. It currently includes

- alm\_tools,
- bit\_manipulation,
- coord\_v\_convert,
- extension,
- fitstools,
- head\_fits,
- healpix\_fft,
- [healpix\\_types](#),
- long\_intrinsic,
- mask\_tools,
- misc\_utils,
- num\_rec,
- obsolete,
- paramfile\_io,
- pix\_tools,
- ran\_tools,
- rngmod,
- statistics,
- udgrade\_nr,
- utilities.

Note that `mpi_alm_tools` is not included since it requires the MPI library for compilation.

---

## EXAMPLE:

```
use healpix_modules
print*, ' pi = ', PI
print*, ' number of pixels in a Nside=64 map:', nside2npix(64)
```

Invoking `healpix_modules` gives access to all **HEALPix** routines and parameters.

---

# healpix\_types

Location in HEALPix directory tree: [src/f90/mod/healpix\\_types.F90](#)

This module defines a set of parameters used by most other HEALPix modules.

The parameters defined in `healpix_types` include

- 'kind' parameters, used when defining the type of a variable,

name	type	value <sup>a</sup>	definition
I1B	integer	1	number of bytes in the hardware-supported signed integers covering the range -99 to 99 with the least margin
I2B	integer	2	same as above for the range -9999 to 9999 (ie, 4 digits)
I4B	integer	4	same as above for 9 digits
I8B	integer	8	same as above for 16 digits <sup>b</sup>
SP	integer	4	number of bytes in the hardware-supported floating-point numbers covering the range $10^{-30}$ to $10^{30}$ with the least margin (hereafter single precision)
DP	integer	8	same as above for the range $10^{-200}$ to $10^{200}$ (double precision)
SPC	integer	4	number of bytes in real ( <i>or</i> imaginary) part of single precision complex numbers
DPC	integer	8	same as above for double precision complex numbers
LGT	integer	4	number of bytes in logical variables

<sup>a</sup>actual value may depend on hardware or compiler

<sup>b</sup>may not be supported by some hardware or compiler; on those systems, the user should set the preprocessing variable `N064BITS` to 1 during compilation to demote automatically `I8B` to `I4B`

- largest accessible numbers,

name	type or kind	value <sup>a</sup>	definition
MAX_I1B	integer	127	largest number accessible to integers of kind I1B
MAX_I2B	integer	32767	same as above for I2B integers
MAX_I4B	integer	$2^{31} - 1 \simeq 2.1 \cdot 10^9$	same as above for I4B integers
MAX_I8B	I8B	$2^{63} - 1 \simeq 9.2 \cdot 10^{18}$	same as above for I8B integers
MAX_SP	SP	$\simeq 3.40 \cdot 10^{38}$	same as above for SP floating-point
MAX_DP	DP	$\simeq 1.80 \cdot 10^{308}$	same as above for DP floating-point

<sup>a</sup>actual value may depend on hardware or compiler

- mathematical definitions,

name	kind	value	definition
QUARTPI	DP	$\pi/4$	
HALFPI	DP	$\pi/2$	
PI	DP	$\pi \simeq 3.14159\dots$	
TWOPI	DP	$2\pi$	
FOURPI	DP	$4\pi$	
SQRT2	DP	$\sqrt{2}$	
EULER	DP	$\gamma \simeq 0.577\dots$	Euler constant
SQ4PI_INV	DP	$1/\sqrt{4\pi}$	
TWOTHIRD	DP	$2/3$	
DEG2RAD	DP	$\pi/180$	Degrees to Radians conversion factor
RAD2DEG	DP	$180/\pi$	Radians to Degrees conversion factor

- and **HEALPix** specific definitions,

name	type or kind	value	definition
HPX_SBADVAL	SP	$-1.6375 \cdot 10^{30}$	default sentinel value given to missing pixels in single precision data sets
HPX_DBADVAL	DP	$-1.6375 \cdot 10^{30}$	same as above for double precision data sets
FILENAMELEN	integer	1024	default length in character of file names.
HEALPIX_VERSION	character	“3.50”	current <b>HEALPix</b> package version.

---

### EXAMPLE:

```
use healpix_types
real(kind=DP) :: dx
print*, ' pi = ', PI
```

The value of PI, as well as all other healpix\_types parameters are made known to the code

---

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

```

FORMAT      call in_ring(nside, iz, phi0, dphi, listir, nir,
nest)

```

name & dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\text{side}}$ parameter of the map.
iz	I4B	IN	ring number, counted southwards from the north pole.
phi0	DP	IN	central $\phi$ position in the slice.
dphi	DP	IN	defines the size of the slice. The slice has length $2 \times dphi$ along the ring with center at $phi0$ .
listir(0:4*nside-1)	I4B/ I8B	OUT	The pixel indexes in the slice.
nir	I4B	OUT	the number of pixels in the slice. $nir \leq 4N_{\text{side}}$
nest (OPTIONAL)	I4B	IN	The pixel indexes are in the NESTED numbering scheme if <b>nest</b> =1, and in RING scheme otherwise.

```
call in ring(256, 10, 0, 0.1, listir, nir, nest=1)
```

HEALPix 3.50

---

## MODULES & ROUTINES

This section lists the modules and routines used by **in\_ring**.

<b>ring2nest</b>	conversion from RING scheme pixel index to NESTED scheme pixel index
<b>next_in_line_nest</b>	returns NESTED index of pixel lying to the East of the current pixel and on the same ring

---

## RELATED ROUTINES

This section lists the routines related to **in\_ring**.

<b>pix2ang, ang2pix</b>	convert between angle and pixel number.
<b>pix2vec, vec2pix</b>	convert between a cartesian vector and pixel number.
<b>getdisc_ring</b>	find all pixels within a certain radius.

---

## input\_map\*

---

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine reads a **HEALPix** map from a FITS file. This can deal with full sky as well as cut sky maps.

---

FORMAT	call input_map*( <i>filename</i> , <i>map</i> , <i>npixtot</i> , <i>nmaps</i> [, <i>fmissval</i> , <i>header</i> , <i>units</i> , <i>extno</i> , <i>ig-</i> <i>nore_polconv</i> ])
--------	--

Arguments appearing in *italic* are optional.

---

## ARGUMENTS

name & dimensionality	kind	i/o	description
filename(len=filenamen)	CHR	IN	FITS file to be read from, containing a full sky or cut sky map
map(0:npixtot-1,1:nmaps)	SP/ DP	OUT	full sky map(s) constructed from the data present in the file, missing pixels are filled with fmissval
npixtot	I4B/ I8B	IN	number of pixels in the full sky map
nmaps	I4B	IN	number of maps in the file
fmissval	SP/ DP	IN	value to be given to missing pixels, ( <b>default:</b> 0)
header(LEN=80)(1:)	CHR	OUT	FITS extension header
units(LEN=20)(1:nmaps)	CHR	OUT	maps units
extno	I4B	IN	extension number to read the data from (0 based).( <b>default:</b> 0) (the first extension is read)
ignore_polcconv	LGT	IN	by default ( <b>ignore_polcconv=.false.</b> ) the output of this routine depends on the value of the FITS keyword POLCCONV found in <b>filename</b> , as described in the <a href="#">note on POLCCONV</a> in <a href="#">The HEALPix Primer</a> . Setting <b>ignore_polcconv=.true.</b> will force input_map to ignore that keyword.

---

## EXAMPLE:

```

use pix_tools, only:  nside2npix
use fitstools, only:  getsize_fits, input_map
...
npixtot = getsize_fits('map.fits',nmaps=nmaps, nside=nside)
npix = nside2npix(nside)
allocate(map(0:npix-1,1:nmaps))
call input_map('map.fits', map, npix, nmaps, fmissval=0.)

```

Reads into **map** the content of the FITS file 'map.fits'. If there are missing pixels in the input file (ie, having value NaN (Not of Number),  $\pm$  Infinity or matching the FITS keyword BAD\_DATA) they will take on output the value provided in optional **fmissval** (here 0, which also is its default value).

---

## MODULES & ROUTINES

This section lists the modules and routines used by **input\_map\***.

<b>fitstools</b>	module, containing:
printerror	routine for printing FITS error messages.
read_bintab	routine to read a binary table from a FITS file
read_fits_cut4	routine to read cut sky map from a FITS file
cfitsio	library for FITS file handling.

---

## RELATED ROUTINES

This section lists the routines related to **input\_map\***.

anafast	executable that reads a <b>HEALPix</b> map and analyses it.
synfast	executable that generate full sky <b>HEALPix</b> maps
getsize_fits	subroutine to know the size of a FITS file.
output_map	subroutine to write a FITS file from a <b>HEALPix</b> map
write_bintabh	subroutine to write a large array into a FITS file piece by piece
input_tod*	subroutine to read an arbitrary subsection of a large binary table

---



# input\_tod\*

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine reads a large binary table (for instance a Time Ordered Data set) from a FITS file. The user can choose to read only a section of the table, starting from an arbitrary position. The data can be read into a single or double precision array.

**FORMAT**                    call input\_tod\*(*filename*, *tod*, *npix*, *ntods* [, *header*, *firstpix*, *fmissval*])

Arguments appearing in *italic* are optional.

## ARGUMENTS

name & dimensionality	kind	in/out	description
<i>filename</i> (LEN= <i>filename</i> len)	CHR	IN	FITS file to be read from
<i>tod</i> (0: <i>npix</i> -1,1: <i>ntods</i> )	SP/ DP	OUT	array constructed from the data present in the file (from the sample <b>firstpix</b> to <b>firstpix</b> + <b>npix</b> - 1. Missing pixels or time samples are filled with <b>fmissval</b> .
<i>npix</i>	I8B	IN	number of pixels or samples to be read. See Note below.
<i>ntods</i>	I4B	IN	number of columns to read
<i>header</i> (LEN=80)(1:)	CHR	OUT	FITS extension header
<i>firstpix</i>	I8B	IN	first pixel (or time sample) to read from (0 based). ( <b>default:</b> 0). See Note below.
<i>fmissval</i>	SP/ DP	IN	value to be given to missing pixels, its default value is 0. Should be of the same type as <b>tod</b> .

**Note :** Indices and number of data elements larger than  $2^{31}$  are only accessible in FITS files on computers with 64 bit enabled compilers and with some specific compilation options of cfitsio (see cfitsio documentation).

## MODULES & ROUTINES

This section lists the modules and routines used by **input\_tod\***.

<b>fitstools</b>	module, containing:
<b>printerror</b>	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.

---

## RELATED ROUTINES

This section lists the routines related to **input\_tod\***.

<b>anafast</b>	executable that reads a <b>HEALPix</b> map and analyses it.
<b>synfast</b>	executable that generate full sky <b>HEALPix</b> maps
<b>getsize_fits</b>	subroutine to know the size of a FITS file.
<b>write_bintabh</b>	subroutine to write large arrays into FITS files
<b>output_map</b>	subroutine to write a FITS file from a <b>HEALPix</b> map
<b>input_map</b>	subroutine to read a <b>HEALPix</b> map (either full sky or cut sky) from a FITS file

---

# long\_count, long\_size

Location in HEALPix directory tree: [src/f90/mod/long\\_intrinsic.F90](#)

The Fortran90 module `long_intrinsic` contains a subset of intrinsic functions (currently `count` and `size`) compiled so that they return **I8B** variables instead of the default integer (generally **I4B**), therefore allowing the handling of arrays with more than  $2^{31} - 1$  elements.

## FUNCTIONS:

**cnt** = long\_count(**mask1**)

returns the I8B integer value that is the number of elements of the logical array **mask1** that have the value `true`.

**sz** = long\_size(**array1** [,**dim**])

**sz** = long\_size(**array2** [,**dim**])

returns the I8B integer value that is the size of the 1D array **array1** or 2D array **array2** or their extent along the dimension **dim** if the scalar integer **dim** is provided.

## ARGUMENTS

name & dimensionality	kind	in/out	description
<b>cnt</b>	I8B	OUT	number of elements with value <code>true</code>
<b>sz</b>	I8B	OUT	size or extent of array
<b>mask1(:)</b>	LGT	IN	1D logical array
<b>array1(:)</b>	I4B/ I8B/ SP/ DP	IN	1D integer or real array
<b>array2(:, :)</b>	I4B/ I8B/ SP/ DP	IN	2D integer or real array
<b>dim</b> (OPTIONAL)	I4B	IN	dimension (starting at 1) along which the array extent is measured.

---

**EXAMPLE:**

```
use healpix_modules
real(SP), dimension(:,:), allocatable :: bigarray
allocate(bigarray(2_i8b**31+5, 3))
print*, size(bigarray), size(bigarray,1), size(bigarray,dim=2)
print*, long_size(bigarray), long_size(bigarray,1),
long_size(bigarray,dim=2)
deallocate(bigarray)
```

Will return (with default compilation options)

```
-2147483633 -2147483643 3
6442450959 2147483653 3
```

meaning that `long_size` handles correctly this large array while  
by default `size` does not.

---

# map2alm\*

Location in HEALPix directory tree: [src/f90/mod/alm\\_tools.F90](#)

This routine is a wrapper to 5 internal routines: `map2alm_sc`, `map2alm_sc_pre`, `map2alm_pol`, `map2alm_pol_pre1`, `map2alm_pol_pre2`. These routines analyse a **HEALPix** *RING ordered* map and return  $a_{\ell m}^T$  (and if specified  $a_{\ell m}^E$  and  $a_{\ell m}^B$ ) values up to the desired order in  $\ell$  (maximum  $3*N_{\text{side}}$ ). The different routines are called depending on what parameters are passed. Some routines analyse with or without precomputed harmonics and some with or without polarisation.

---

**FORMAT**                      call   map2alm\*([nsmax](#),   [nlmax](#),   [nmmax](#),  
                                  [map\\_TQU](#),        [alm\\_TGC](#)[,        [zbounds](#)=,  
                                  [w8ring\\_TQU](#)=, [plm](#)=])

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	the $N_{\text{side}}$ value of the map to analyse.
nlmax	I4B	IN	the maximum $\ell$ value for the analysis.
nmmax	I4B	IN	the maximum $m$ value for the analysis.
map_TQU(0:12*nsmax**2-1)	SP/ DP	IN	if only the temperature map is to be analyse, the map-array should be passed with this rank.
map_TQU(0:12*nsmax**2-1, 1:3)	SP/ DP	IN	if both temperature an polarisation maps are to be analysed, the map array should have this rank, where the second index is (1,2,3) corresponding to (T,Q,U).

---

alm_TGC(1:p, 0:nlmax, 0:nmmax)	SPC/ DPC	OUT	The $a_{\ell m}$ values output from the analysis. $p$ is 1 or 3 dependent on whether polarisation is included or not. In the former case, the first index is (1,2,3) corresponding to (T,E,B).
zbounds(1:2), OPTIONAL	DP	IN	section of the map on which to perform the $a_{\ell m}$ analysis, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$ . If $\text{zbounds}(1) < \text{zbounds}(2)$ , it is performed <i>on</i> the strip $\text{zbounds}(1) < z < \text{zbounds}(2)$ ; if not, it is performed <i>outside</i> the strip $\text{zbounds}(2) \leq z \leq \text{zbounds}(1)$ . If absent, the whole map is processed.
w8ring_TQU(1:2*nsmax, 1:p), OPTIONAL	DP	IN	ring weights for quadrature corrections. If ring weights are not used, this array should be 1 everywhere. $p$ is 1 for a temperature analysis and 3 for (T,Q,U).
plm(0:(nlmax+1)*(nlmax+2)*nsmax-1), OPTIONAL	DP	IN OP-	If this optional matrix is passed with this rank, precomputed $P_{\ell m}(\theta)$ are used instead of recursion. Note that since version 2.20 this feature has become obsolete because of algorithm optimizations.
plm(0:(nlmax+1)*(nlmax+2)*nsmax-1, 1:3), OPTIONAL	DP	IN	If this optional matrix is passed with this rank, precomputed $P_{\ell m}(\theta)$ AND precomputed tensor harmonics are used instead of recursion.

---

## EXAMPLE:

```

use healpix_modules
integer(i4b) :: nside, lmax
real(dp), allocatable, dimension(:, :) :: dw8
real(dp), dimension(2) :: z
real(sp), allocatable, dimension(:, :) :: map
complex(spc), allocatable, dimension(:, :, :) :: alm
nside = 256
lmax = 512
allocate(dw8(1:2*nside, 1:3))

```

```

allocate(map(0:nside2npix(nside)-1,1:3))
allocate(alm(1:3, 0:lmax, 0:lmax))
dw8 = 1.0_dp
z = sin(10.0_dp * DEG2RAD)
call map2alm(nside, lmax, lmax, map, alm, (\ z, -z \) , dw8)

```

Analyses temperature and polarisation maps passed in map. The map has an  $N_{\text{side}}$  of 256, and the analysis is performed up to 512 in  $\ell$  and  $m$ . The resulting  $a_{\ell m}$  coefficients for temperature and polarisation are returned in alm. A  $10^\circ$  cut on each side of the equator is applied. Uniform weights are used.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **map2alm\***.

ring_analysis	Performs FFT for the ring analysis.
misc_util	module, containing:
assert_alloc	routine to print error message when an array is not properly allocated

Note: Starting with **version 3.10**, libsharp routines will be called when precomputed  $P_{\ell m}$  are not provided.

---

## RELATED ROUTINES

This section lists the routines related to **map2alm\***.

anafast	executable using map2alm* to analyse maps.
alm2map	routine performing the inverse transform of map2alm*.
dump_als	write $a_{\ell m}$ coefficients computed by map2alm* into a FITS file
map2alm_iterative	similar to map2alm* with iterative scheme.

---

## map2alm\_iterative

Location in HEALPix directory tree: [src/f90/mod/alm\\_tools.F90](#)

This routine covers and extends the functionalities of [map2alm](#): it analyzes a (polarised) **HEALPix** *RING ordered* map and returns its  $a_{\ell m}$  coefficients for temperature (and polarisation) up to a specified multipole, and use precomputed harmonics if those are provided, but it also can also perform an iterative (Jacobi) determination of the  $a_{\ell m}$ , and apply a pixel mask if one is provided.

Denoting **A** and **S** the analysis ([map2alm](#)) and synthesis ([alm2map](#)) operators and **a**, **m** and **w**, the  $a_{\ell m}$ , map and pixel mask vectors, the Jacobi iterative process reads

$$\mathbf{a}^{(n)} = \mathbf{a}^{(n-1)} + \mathbf{A} \cdot (\mathbf{w} \cdot \mathbf{m} - \mathbf{S} \cdot \mathbf{a}^{(n-1)}), \quad (10)$$

with

$$\mathbf{a}^{(0)} = \mathbf{A} \cdot \mathbf{w} \cdot \mathbf{m}. \quad (11)$$

During the processing, the standard deviation of the input map (**w.m**) and the current residual map  $(\mathbf{w} \cdot \mathbf{m} - \mathbf{S} \cdot \mathbf{a}^{(n-1)})$  is printed out, with the latter expected to get smaller and smaller as  $n$  increases.

The standard deviation of map  $x$  has the usual definition  $\sigma \equiv \sqrt{\sum_{p=1}^N \frac{(x(p) - \bar{x})^2}{N-1}}$ , where the mean is  $\bar{x} \equiv \sum_{p=1}^N \frac{x(p)}{N}$ , and the index  $p$  runs over all pixels.

In [version 3.50](#) a bug affecting previous versions of `map2alm_iterative` has been fixed. (It occurred when `iter_order > 0` was used in conjunction with a `mask` and/or a restrictive `zbounds`, with a magnitude that depended on each of those factors and was larger for non-boolean masks (ie,  $\mathbf{w}^2 \neq \mathbf{w}$ ). To assess the impact of this bug on previous results, the old implementation remains available in `map2alm_iterative_old`). The result was correct when the mask (if any) was applied to the map prior to the `map2alm_iterative` calling, or when no iteration was requested.



**FORMAT**            call map2alm\_iterative(nsmax, nlmax, nm-  
                         max, iter\_order, map\_TQU, alm\_TGC[,  
                         zbounds, w8ring\_TQU, plm, mask])

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	the $N_{\text{side}}$ value of the map to analyse.
nlmax	I4B	IN	the maximum $\ell$ value ( $\ell_{\text{max}}$ ) for the analysis.
nmmax	I4B	IN	the maximum $m$ value for the analysis.
iter_order	I4B	IN	the order of Jacobi iteration. Increasing that order improves the accuracy of the final $a_{\ell m}$ but increases the computation time $T_{\text{CPU}} \propto 1 + 2 \times \text{iter\_order}$ . <code>iter_order = 0</code> is a straight analysis, while <code>iter_order = 3</code> is usually a good compromise.
map_TQU(0:12*nsmax**2-1, 1:p)	SP/ DP	INOUT	input map. $p$ is 1 or 3 depending if temperature (T) only or temperature and polarisation (T, Q, U) are to be analysed. It will be altered on output if a <b>mask</b> is provided and/or if <b>iter_order</b> > 0 and <b>zbounds</b> is provided.
alm_TGC(1:p, 0:nlmax, 0:nm- max)	SPC/ DPC	OUT	The $a_{\ell m}$ values output from the analysis. $p$ is 1 or 3 depending on whether polarisation is included or not. In the former case, the first index is (1,2,3) corresponding to (T,E,B).
zbounds(1:2), OPTIONAL	DP	IN	section of the map on which to perform the $a_{\ell m}$ analysis, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$ . If <code>zbounds(1) &lt; zbounds(2)</code> , it is performed <i>on</i> the strip <code>zbounds(1) &lt; z &lt; zbounds(2)</code> ; if not, it is performed <i>outside</i> the strip <code>zbounds(2) ≤ z ≤ zbounds(1)</code> . If absent, the whole map is processed.

---

w8ring_TQU(1:2*nsmax,1:p), OPTIONAL	DP	IN	ring weights for quadrature corrections. $p$ is 1 for a temperature analysis and 3 for (T,Q,U). If absent, the ring weights are all set to 1.
plm(0:,1:p), OPTIONAL	DP	IN	If this optional matrix is passed, pre-computed scalar (and tensor) $P_{\ell m}(\theta)$ are used instead of recursion.
mask(0:12*nsmax**2-1,1:q), OPTIONAL	SP/ DP	IN	pixel mask, assumed to have the same resolution (and RING ordering) as the map. The map <code>map_TQU</code> is multiplied by that mask before being analyzed, and will therefore be altered on output. $q$ should be in $\{1, 2, 3\}$ . If $p = q = 3$ , then each of the 3 masks is applied to the respective map. If $p = 3$ and $q = 2$ , the first mask is applied to the first map, and the second mask to the second (Q) and third (U) map. If $p = 3$ and $q = 1$ , the same mask is applied to the 3 maps. Note: the output $a_{\ell m}$ are computed directly on the masked map, and are <i>not</i> corrected for the loss of power, correlation or leakage created by the mask.

---

**EXAMPLE:**

```

use healpix_types
use alm_tools
use pix_tools
integer(i4b) :: nside, lmax, npix, iter
real(sp), allocatable, dimension(:, :) :: map
real(sp), allocatable, dimension(:) :: mask
complex(spc), allocatable, dimension(:, :, :) :: alm

nside = 256
lmax = 512
iter = 2
npix = nside2npix(nside)
allocate(map(0:npix-1, 1:3))
allocate(mask(0:npix-1))

```

```

mask(0:) = 0. ! set unvalid pixels to 0
mask(0:10000-1) = 1. ! valid pixels
allocate(alm(1:3, 0:lmax, 0:lmax))
call map2alm_iterative(nside, lmax, lmax, iter, map, alm, mask=mask)

```

Analyses temperature and polarisation signals in the first 10000 pixels of `map` (as determined by `mask`). The map has an  $N_{\text{side}}$  of 256, and the analysis is supposed to be performed up to 512 in  $\ell$  and  $m$ . The resulting  $a_{\ell m}$  coefficients for temperature and polarisation are returned in `alm`. Uniform weights are assumed. In order to improve the  $a_{\ell m}$  accuracy, 2 Jacobi iterations are performed.

---

## MODULES & ROUTINES

This section lists the modules and routines used by `map2alm_iterative`.

<code>map2alm</code>	Performs the alm analysis
<code>alm2map</code>	Performs the map synthesis
<code>misc_util</code>	module, containing:
<code>assert_alloc</code>	routine to print error message when an array is not properly allocated

---

## RELATED ROUTINES

This section lists the routines related to `map2alm_iterative`.

<code>anafast</code>	executable using <code>map2alm_iterative</code> to analyse maps.
<code>alm2map</code>	routine performing the inverse transform of <code>map2alm_iterative</code> .
<code>alm2map_spin</code>	synthesize spin weighted maps.
<code>dump_alms</code>	write $a_{\ell m}$ coefficients computed by <code>map2alm_iterative</code> into a FITS file
<code>map2alm_spin</code>	analyze spin weighted maps.

---

# map2alm\_spin\*

Location in HEALPix directory tree: [src/f90/mod/alm\\_tools.F90](#)

This routine extracts the alm coefficients out of maps of spin  $s$  and  $-s$ . A (complex) map  $S$  of spin  $s$  is a linear combination of the spin weighted harmonics  ${}_sY_{\ell m}$

$${}_sS(p) = \sum_{\ell m} {}_s a_{\ell m} {}_s Y_{\ell m}(p) \quad (12)$$

for  $\ell \geq |m|$ ,  $\ell \geq |s|$ , and is such that  ${}_sS^* = -{}_sS$ .

The [usual phase convention for the spin weighted harmonics](#) is  ${}_sY_{\ell m}^* = (-1)^{s+m} {}_{-s}Y_{\ell -m}$  and therefore  ${}_s a_{\ell m}^* = (-1)^{s+m} {}_{-s} a_{\ell -m}$ . The two (real) input maps for map2alm\_spin\* are defined respectively as

$$|s|S^+ = (|s|S + |s|S)/2 \quad (13)$$

$$|s|S^- = (|s|S - |s|S)/(2i). \quad (14)$$

map2alm\_spin\* outputs the alm coefficients defined as

$$|s|a_{\ell m}^+ = -(|s|a_{\ell m} + (-1)^s {}_{-s}a_{\ell m})/2 \quad (15)$$

$$|s|a_{\ell m}^- = -(|s|a_{\ell m} - (-1)^s {}_{-s}a_{\ell m})/(2i) \quad (16)$$

for  $m \geq 0$ , knowing that, just as for spin 0 maps, the coefficients for  $m < 0$  are given by

$$|s|a_{\ell -m}^+ = (-1)^m |s|a_{\ell m}^{+*}, \quad (17)$$

$$|s|a_{\ell -m}^- = (-1)^m |s|a_{\ell m}^{-*}. \quad (18)$$

With these definitions,  ${}_2a^+$ ,  ${}_2a^-$ ,  ${}_2S^+$  and  ${}_2S^-$  match **HEALPix** polarization  $a^E$ ,  $a^B$ ,  $Q$  and  $U$  respectively. However, for  $s = 0$ ,  ${}_0a_{\ell m}^+ = -a_{\ell m}^T$ ,  ${}_0a_{\ell m}^- = 0$ ,  ${}_0S^+ = T$ ,  ${}_0S^- = 0$ .

---

**FORMAT**                      call map2alm\_spin\*([nsmax](#), [nlmax](#), [nmmax](#),  
                                  [spin](#), [map](#), [alm](#)[, [zbounds](#)=, [w8ring\\_TQU](#)=])

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	the $N_{\text{side}}$ value of the map to analyse.
nlmax	I4B	IN	the maximum $\ell$ value for the analysis.
nmmax	I4B	IN	the maximum $m$ value for the analysis.
spin	I4B	IN	the spin $s$ of the maps to be analysed (only its absolute value is relevant).
map(0:12*nsmax**2-1, 1:2)	SP/ DP	IN	$ _s S^+$ and $ _s S^-$ input maps
alm(1:2, 0:nlmax, 0:nmmax)	SPC/ DPC	OUT	The $ _s a_{\ell m}^+$ and $ _s a_{\ell m}^-$ output values.
zbounds(1:2), OPTIONAL	DP	IN	section of the map on which to perform the $a_{\ell m}$ analysis, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$ . If $\text{zbounds}(1) < \text{zbounds}(2)$ , it is performed <i>on</i> the strip $\text{zbounds}(1) < z < \text{zbounds}(2)$ ; if not, it is performed <i>outside</i> the strip $\text{zbounds}(2) \leq z \leq \text{zbounds}(1)$ . If absent, the whole map is processed.
w8ring_TQU(1:2*nsmax,1:2), OPTIONAL	DP	IN	ring weights for quadrature corrections. If ring weights are not used, this array should be 1 everywhere.

---

### EXAMPLE:

```

use healpix_types
use alm_tools
use pix_tools
integer(i4b) :: nside, lmax, spin
real(sp), allocatable, dimension(:, :) :: map
complex(spc), allocatable, dimension(:, :, :) :: alm

nside = 256
lmax = 512
spin = 5
allocate(map(0:nside2npix(nside)-1, 1:2))
allocate(alm(1:2, 0:lmax, 0:lmax))

```

```
...
call map2alm_spin(nside, lmax, lmax, spin, map, alm)
```

Analyses spin 5 and -5 maps. The maps have an  $N_{\text{side}}$  of 256, and the analysis is performed up to 512 in  $\ell$  and  $m$ . The resulting  $a_{\ell m}$  coefficients for are returned in alm.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **map2alm\_spin\***.

ring_analysis	Performs FFT for the ring analysis.
compute_lam_mm, get_pixel_layout,	
gen_lamfac_der, gen_mfac,	
gen_recfac, init_rescale, l_min_ylm	Ancillary routines used for ${}_s Y_{\ell m}$ recursion
<b>misc_util</b>	module, containing:
<b>assert_alloc</b>	routine to print error message when an array is not properly allocated

Note: Starting with **version 3.10**, libsharp routines will be called if  $0 < |s| \leq 100$ .

---

## RELATED ROUTINES

This section lists the routines related to **map2alm\_spin\***.

<b>alm2map_spin</b>	routine performing the inverse transform of map2alm_spin*.
<b>map2alm</b>	routine analyzing temperature and polarization maps

---

## maskborder\_\_nest

Location in HEALPix directory tree: [src/f90/mod/mask\\_tools.F90](#)

For a input binary mask in NESTED ordering, maskborder\_\_nest identifies the pixels located on the inner boundary of *invalid* regions

---

**FORMAT**                    call    maskborder\_\_nest([nside](#),        [mask\\_in](#),  
                                 [mask\\_out](#), nbordpix, [ [border\\_value](#)])

Arguments appearing in *italic* are optional.

---

### ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	The $N_{\text{side}}$ value of the input mask.
mask_in(0:Npix-1)	I4B	IN	Input binary NESTED-ordered mask. Npix = 12*nside*nside
mask_out(0:Npix-1)	I4B	OUT	Output NESTED-ordered mask, in which inner border pixels (defined as 0-valued and adjacent to 1-valued pixels) take the value 2 (or <a href="#">border_value</a> ). Can be the same array as mask_in.
nbordpix	I4B	OUT	Number of border pixels found
<i>border_value</i>	I4B	IN	value to be given to border pixels in output mask. ( <b>default: 2</b> ).

---

### EXAMPLE:

```
use healpix_types
use healpix_modules
...
call maskborder__nest(nside, mask_in, mask_in, nbordpix)
```



For a binary input mask `mask_in`, it will look for border pixels and output their number in `nborpix`. In this example the `mask_in` will be modified so that border pixels take value 2 on output.

---

## MODULES & ROUTINES

This section lists the modules and routines used by `maskborder_nest`.

<code>mask_tools</code>	mask processing module (see related routines below)
-------------------------	---

---

## RELATED ROUTINES

This section lists the routines related to `maskborder_nest`.

<code>dist2holes_nest</code>	angular distance to closest invalid pixel of the given mask
<code>fill_holes_nest</code>	turn to <i>valid</i> all pixels located in 'holes' containing fewer pixels than the given threshold
<code>maskborder_nest</code>	identify inner boundary pixels of 'holes' for given mask
<code>size_holes_nest</code>	returns size (in pixels) of holes found in input mask

---

# medfiltmap\*

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

This routine performs the median filtering of a **HEALPix** full sky map for a given neighborhood radius

---

**FORMAT**                    call medfiltmap\*(in\_map, radius, med\_map  
                              [, nest, fmissval, fill\_holes])

---

## ARGUMENTS

name & dimensionality		kind	in/out	description
in_map(0:npix-1)		SP/ DP	IN	Full sky <b>HEALPix</b> map to filter. <b>npix</b> should be valid <b>HEALPix</b> pixel number.
radius		DP	IN	Radius in RADIANS of the disk on which the median is computed.
med_map(0:npix-1)		SP/ DP	OUT	Median filtered map: each pixel is the median of the input map valid neighboring pixels contained within a distance <b>radius</b>
nest	OPTIONAL	I4B	IN	set to 1 if the map ordering is NESTED, set to 0 if it is RING.
fmissval	OPTIONAL	SP/ DP	IN	sentinel value given to missing or non-valid pixels. Default: <b>HPX_SBADVAL</b> or <b>HPX_DBADVAL</b> = $-1.6375 \cdot 10^{30}$
fill_holes	OPTIONAL	OP- LGT	IN	if set to <b>.true.</b> will replace non-valid pixels by median of neighbors; if set to <b>.false.</b> will leave non-valid pixels unchanged. Default: <b>.false.</b>

---

## EXAMPLE:

```
use healpix_types
use pix_tools
...
call medfiltmap(map, 0.5*DEG2RAD, med)
```

Output in **med** the median filter of **map**, using a filter radius of 0.5 Deg

---

## MODULES & ROUTINES

This section lists the modules and routines used by **medfiltmap\***.

<b>statistics</b>	module, containing:
<b>median</b>	routine to compute the median of a data set
<b>pix_tools</b>	module, containing:
<b>pix2vec_ring, pix2vec_nest</b>	routines to find the location of a pixel on the sky
<b>query_disc</b>	routine to find pixels lying within a radius of a given point

---

# median\*

Location in HEALPix directory tree: [src/f90/mod/statistics.f90](#)

This function computes the median of a data set

---

**FORMAT**                      var=median\*(data[, badval, even])

---

## ARGUMENTS

---

name & dimensionality	kind	in/out	description
var	SP/ DP	OUT	median of the data set, defined as the middle number (or the average of the 2 middle numbers) once the valid data points are sorted in monotonous order
data(:)	SP/ DP	IN	data set
badval (OPTIONAL)	SP/ DP	IN	sentinel value given to bad data points. Data points with this value will be ignored during calculation of the median. If not set, all points will be considered. <b>Do not set to 0!</b>
even (OPTIONAL)	LGT	IN	if set to <code>.true.</code> and the number of valid data points is even, will output the average of the 2 middle points (which doubles the calculation time). If the number of points is odd, the single middle point is output and this keyword is ignored.

---

## EXAMPLE:

```
use statistics, only: median
...
med = median(map, even=.true.)
```

Outputs in med the median of map

---

## MODULES & ROUTINES

This section lists the modules and routines used by **median\***.

<b>m_indmed</b>	module of the Orderpack 2.0 package, written by: Michel Ollagnon, <a href="http://www.fortran-2000.com/rank/">http://www.fortran-2000.com/rank/</a> , also available in <a href="src/f90/mod/indmed.f90">src/f90/mod/indmed.f90</a>
indmed	routine of <b>m_indmed</b> to output rank of median

---

## RELATED ROUTINES

This section lists the routines related to **median\***.

<b>compute_statistics</b>	routine min, max, absolute deviation, and first four order moments of a data set
---------------------------	--

---

# merge\_headers

Location in HEALPix directory tree: [src/f90/mod/head\\_fits.F90](#)

This routine merges two FITS headers.

---

**FORMAT**                      call merge\_headers([header1](#), [header2](#))

---

## ARGUMENTS

---

name&dimensionality	kind	in/out	description
header1(LEN=80) DIMENSION(:)	CHR	IN	First header.
header2(LEN=80) DIMENSION(:)	CHR	INOUT	Second header. On output, will contain the concatenation of (in that order) header2 and header1. If header2 is too short to allow the merging the output will be truncated

---

## EXAMPLE:

```
call merge_headers(header1, header2)
```

On output header2 will contain the original header2, followed by the content of header1

---

## MODULES & ROUTINES

This section lists the modules and routines used by **merge\_headers**.

<b>write_hl</b>	more general routine for adding a keyword to a header.
<b>cfitsio</b>	library for FITS file handling.

---

## RELATED ROUTINES

This section lists the routines related to **merge\_headers**.

<code>add_card</code>	general purpose routine to write any keywords into a FITS file header
<code>get_card</code>	general purpose routine to read any keywords from a header in a FITS file.
<code>del_card</code>	routine to discard a keyword from a FITS header
<code>read_par</code> , <code>number_of_alms</code>	routines to read specific keywords from a header in a FITS file.
<code>getsize_fits</code>	function returning the size of the data set in a fits file and reading some other useful FITS keywords

---

## mpi\_alm\_tools\*

---

Location in HEALPix directory tree: [src/f90/mod/mpi\\_alm\\_tools.f90](#)

This module implements MPI parallelization of the alm2map and map2alm routines. It is not compiled by default during installation, but rather intended for users who need massive parallelization in their own programming. Typical applications are Monte Carlo simulations and Markov chain type analyses.

The routines can be called in two modes, either simple or advanced. The former mimics the interface of the standard routines, but with an additional MPI handle as a first argument, and is intended for applications which requires only one or a few transforms. The latter interface provides both more flexibility (in particular the option of pre-computation of the Legendre polynomials) and a simpler interface when multiple transforms are required. This interface is particularly well suited for Monte Carlo simulations and Markov chain type analyses.

---

### EXAMPLE:

- Simple one-line interfaces:
  - `mpi_map2alm_simple`
  - `mpi_alm2map_simple`
- Three-step advanced interfaces:
  1. Initialization:  
`mpi_initialize_alm_tools`
  2. Execution of spherical harmonics transforms
    - `mpi_map2alm (root processor)`
    - `mpi_alm2map (root processor)`
    - `mpi_map2alm_slave (slave processor)`
    - `mpi_alm2map_slave (slave processor)`
  3. Finalizing:  
`mpi_cleanup_alm_tools`





## mpi\_alm2map\*

Location in HEALPix directory tree: [src/f90/mod/mpi\\_alm\\_tools.f90](#)

This subroutine implements MPI parallelization of the serial alm2map routine. It supports both temperature and polarization inputs in both single and double precision. It must only be run by the root node of the MPI communicator.

---

**FORMAT**                      call mpi\_alm2map\*([alms](#), [map](#))

---

### ARGUMENTS

name & dimensionality	kind	in/out	description
alms(1:nmaps,0:lmax,0:nmax)	SPC or DPC	IN	Input alms. If nmaps=1, only temperature information is included; if nmaps=3, polarization information is included
map(0:npix,1:nmaps)	SP or DP	OUT	Output map. nmaps must match that of the input alms array.

---

### EXAMPLE:

```
call mpi_comm_rank(comm, myid, ierr)
if (myid == root) then
    call mpi_initialize_alm_tools(comm, nsmax, nlmax, nmmax,
                                zbounds,polarization, precompute_plms)
    call mpi_alm2map(alms, map)
else
    call mpi_initialize_alm_tools(comm)
    call mpi_alm2map_slave
end
call mpi_cleanup_alm_tools
```

This example 1) initializes the `mpi_alm_tools` module (i.e., allocates internal arrays and defines required parameters), 2) executes a parallel `alm2map` operation, and 3) frees the previously allocated memory.

---

## MODULES & ROUTINES

This section lists the modules and routines used by `mpi_alm2map*`.

`alm_tools`      module

---

## RELATED ROUTINES

This section lists the routines related to `mpi_alm2map*`.

<code>mpi_cleanup_alm_tools</code>	Frees memory that is allocated by the current routine.
<code>mpi_initialize_alm_tools</code>	Allocates memory and defines variables for the <code>mpi_alm_tools</code> module.
<code>mpi_alm2map_slave</code>	Routine for executing a parallel inverse spherical harmonics transform (slave processor interface)
<code>mpi_map2alm</code>	Routine for executing a parallel spherical harmonics transform (root processor interface)
<code>mpi_map2alm_slave</code>	Routine for executing a parallel spherical harmonics transform (slave processor interface)
<code>mpi_alm2map_simple</code>	One-line interface to the parallel inverse spherical harmonics transform
<code>mpi_map2alm_simple</code>	One-line interface to the parallel spherical harmonics transform

---

Location in HEALPix directory tree: `src/f90/mod/mpi_alm_tools.f90`

This subroutine provides a simplified (one-line) interface to the MPI version of `alm2map`. It supports both temperature and polarization inputs in both single and double precision. It must only be run by all nodes in the MPI communicator.

```
FORMAT      call  mpi_alm2map_simple*(comm,  alms,
map)
```

## ARGUMENTS

name & dimensionality	kind	in/out	description
comm	I4B	IN	MPI communicator.
alms(1:nmaps,0:lmax,0:nmax)	SPC or DPC	IN	Input alms. If nmaps=1, only temperature information is included; if nmaps=3, polarization information is included
map(0:npix,1:nmaps)	SP or DP	OUT	Output map. nmaps must match that of the input alms array.

**EXAMPLE:**

```
call mpi_alm2map_simple(comm, map, alms)
```

This example executes a parallel `map2alm` operation through the one-line interface. Although all processors must supply allocated arrays to the routine, only the root processor's information will be used as input, and only the root processor's alms will be complete after execution.

## MODULES & ROUTINES

This section lists the modules and routines used by **mpi\_alm2map\_simple\***.

**alm\_tools**                      module

---

## RELATED ROUTINES

This section lists the routines related to **mpi\_alm2map\_simple\***.

<b>mpi_cleanup_alm_tools</b>	Frees memory that is allocated by the current routine.
<b>mpi_initialize_alm_tools</b>	Allocates memory and defines variables for the mpi_alm_tools module.
<b>mpi_alm2map</b>	Routine for executing a parallel inverse spherical harmonics transform (root processor interface)
<b>mpi_alm2map_slave</b>	Routine for executing a parallel inverse spherical harmonics transform (slave processor interface)
<b>mpi_map2alm</b>	Routine for executing a parallel spherical harmonics transform (root processor interface)
<b>mpi_map2alm_slave</b>	Routine for executing a parallel spherical harmonics transform (slave processor interface)
<b>mpi_map2alm_simple</b>	One-line interface to the parallel spherical harmonics transform

---

---

## mpi\_alm2map\_slave

---

Location in HEALPix directory tree: [src/f90/mod/mpi\\_alm\\_tools.f90](#)

This subroutine complements the master routine `mpi_alm2map`, and should be run by all slaves in the current MPI communicator. It is run without arguments, but after an appropriate call to `initialize_mpi_alm_tools`.

---

**FORMAT**                      `call mpi_alm2map_slave()`

---

### ARGUMENTS

---

None.

---

### EXAMPLE:

```
call mpi_comm_rank(comm, myid, ierr)
if (myid == root) then
    call mpi_initialize_alm_tools(comm, nsmax, nlmax, nmmax,
                                zbounds,polarization, precompute_plms)
    call mpi_alm2map(alms, map)
else
    call mpi_initialize_alm_tools(comm)
    call mpi_alm2map_slave
end
call mpi_cleanup_alm_tools
```

This example 1) initializes the `mpi_alm_tools` module (i.e., allocates internal arrays and defines required parameters), 2) executes a parallel `alm2map` operation, and 3) frees the previously allocated memory.

---

### MODULES & ROUTINES

This section lists the modules and routines used by `mpi_alm2map_slave`.

alm\_tools

module

---

## RELATED ROUTINES

This section lists the routines related to **mpi\_alm2map\_slave**.

<b>mpi_cleanup_alm_tools</b>	Frees memory that is allocated by the current routine.
<b>mpi_initialize_alm_tools</b>	Allocates memory and defines variables for the mpi_alm_tools module.
<b>mpi_alm2map</b>	Routine for executing a parallel inverse spherical harmonics transform (root processor interface)
<b>mpi_map2alm</b>	Routine for executing a parallel spherical harmonics transform (root processor interface)
<b>mpi_map2alm_slave</b>	Routine for executing a parallel spherical harmonics transform (slave processor interface)
<b>mpi_alm2map_simple</b>	One-line interface to the parallel inverse spherical harmonics transform
<b>mpi_map2alm_simple</b>	One-line interface to the parallel spherical harmonics transform

---

## mpi\_cleanup\_alm\_tools

---

Location in HEALPix directory tree: [src/f90/mod/mpi\\_alm\\_tools.f90](#)

This subroutine deallocates any private arrays previously allocated in the `mpi_alm_tools` module. It should be run (without arguments) by all processors in the current communicator after the last call to any of the working routines.

---

**FORMAT**                      call mpi\_cleanup\_alm\_tools()

---

### ARGUMENTS

---

None.

---

### EXAMPLE:

```
call mpi_comm_rank(comm, myid, ierr)
if (myid == root) then
    call mpi_initialize_alm_tools(comm, nsmax, nlmax, nmmax,
                                   zbounds, polarization, precompute_plms)
    call mpi_map2alm(map, alms)
else
    call mpi_initialize_alm_tools(comm)
    call mpi_map2alm_slave
end
call mpi_cleanup_alm_tools
```

This example 1) initializes the `mpi_alm_tools` module (i.e., allocates internal arrays and defines required parameters), 2) executes a parallel `map2alm` operation, and 3) frees the previously allocated memory.

---

### RELATED ROUTINES

This section lists the routines related to `mpi_cleanup_alm_tools`.



---

<code>mpi_initialize_alm_tools</code>	Allocates memory and defines variables for the <code>mpi_alm_tools</code> module.
<code>mpi_alm2map</code>	Routine for executing a parallel inverse spherical harmonics transform (root processor interface)
<code>mpi_alm2map_slave</code>	Routine for executing a parallel inverse spherical harmonics transform (slave processor interface)
<code>mpi_map2alm</code>	Routine for executing a parallel spherical harmonics transform (root processor interface)
<code>mpi_map2alm_slave</code>	Routine for executing a parallel spherical harmonics transform (slave processor interface)
<code>mpi_alm2map_simple</code>	One-line interface to the parallel inverse spherical harmonics transform
<code>mpi_map2alm_simple</code>	One-line interface to the parallel spherical harmonics transform

---

## mpi\_initialize\_alm\_tools

Location in HEALPix directory tree: [src/f90/mod/mpi\\_alm\\_tools.f90](#)

This subroutine initializes the mpi\_alm\_tools module, and must be run prior to any of the advanced interface working routines by all processors in the MPI communicator. The root processor must supply all arguments, while it is optional for the slaves. However, the information is disregarded if they do.

A major advantage of MPI parallelization is large quantities of memory, allowing for pre-computation of the Legendre polynomials even with high  $N_{\text{side}}$  and  $\ell_{\text{max}}$ , since each processor only needs a fraction ( $1/N_{\text{procs}}$ ) of the complete table. This feature is controlled by the “precompute\_plms” parameter. In general, the CPU time can be expected to decrease by roughly 50% using pre-computed Legendre polynomials for temperature calculations, and by about 30% for polarization calculations.

---

**FORMAT**                      call mpi\_initialize\_alm\_tools(comm, [nsmax],  
                                 [nlmax], [nmmax], [zbounds], [polarization],  
                                 [precompute\_plms], [w8ring])

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
comm	I4B	IN	MPI communicator.
nsmax	I4B	IN	the $N_{\text{side}}$ value of the HEALPix map. (OPTIONAL)
nlmax	I4B	IN	the maximum $\ell$ value used for the $a_{\ell m}$ . (OPTIONAL)
nmmax	I4B	IN	the maximum $m$ value used for the $a_{\ell m}$ . (OPTIONAL)

---

zbounds(1:2)	DP	IN	section of the map on which to perform the $a_{\ell m}$ analysis, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$ . If $\text{zbounds}(1) < \text{zbounds}(2)$ , it is performed <i>on</i> the strip $\text{zbounds}(1) < z < \text{zbounds}(2)$ ; if not, it is performed <i>outside</i> the strip $\text{zbounds}(2) \leq z \leq \text{zbounds}(1)$ . If absent, the whole map is processed. (OPTIONAL)
polarization	LGT	IN	if polarization is required, this should be set to true, else it should be set to false. (OPTIONAL)
precompute_plms	I4B	IN	0 = do not pre-compute any $P_{\ell m}$ 's; 1 = pre-compute $P_{\ell m}^T$ ; 2 = pre-compute $P_{\ell m}^T$ and $P_{\ell m}^P$ . (OPTIONAL)
w8ring_TQU(1:2*nsmax, 1:p)	DP	IN	ring weights for quadrature corrections. If ring weights are not used, this array should be 1 everywhere. p is 1 for a temperature analysis and 3 for (T,Q,U). (OPTIONAL)

---

### EXAMPLE:

```

call mpi_comm_rank(comm, myid, ierr)
if (myid == root) then
    call mpi_initialize_alm_tools(comm, nsmax, nlmax, nmmax,
        zbounds, polarization, precompute_plms)
    call mpi_map2alm(map, alms)
else
    call mpi_initialize_alm_tools(comm)
    call mpi_map2alm_slave
end
call mpi_cleanup_alm_tools

```

This example 1) initializes the mpi\_alm\_tools module (i.e., allocates internal arrays and defines required parameters), 2) executes a parallel map2alm operation, and 3) frees the previously allocated memory.

---

## RELATED ROUTINES

This section lists the routines related to `mpi_initialize_alm_tools`.

<code>mpi_cleanup_alm_tools</code>	Frees memory that is allocated by the current routine.
<code>mpi_alm2map</code>	Routine for executing a parallel inverse spherical harmonics transform (root processor interface)
<code>mpi_alm2map_slave</code>	Routine for executing a parallel inverse spherical harmonics transform (slave processor interface)
<code>mpi_map2alm</code>	Routine for executing a parallel spherical harmonics transform (root processor interface)
<code>mpi_map2alm_slave</code>	Routine for executing a parallel spherical harmonics transform (slave processor interface)
<code>mpi_alm2map_simple</code>	One-line interface to the parallel inverse spherical harmonics transform
<code>mpi_map2alm_simple</code>	One-line interface to the parallel spherical harmonics transform

---

## mpi\_map2alm\*

Location in HEALPix directory tree: [src/f90/mod/mpi\\_alm\\_tools.f90](#)

This subroutine implements MPI parallelization of the serial map2alm routine. It supports both temperature and polarization inputs in both single and double precision. It must only be run by the root node of the MPI communicator.

---

**FORMAT**                      call mpi\_map2alm\*([map](#), [alms](#))

---

### ARGUMENTS

name & dimensionality	kind	in/out	description
map(0:npix,1:nmaps)	SP or DP	IN	map to analyse. If nmaps=1, only temperature information is included; if nmaps=3, polarization information is included
alms(1:nmaps,0:lmax,0:nmax)	SPC or DPC	OUT	output alms. nmaps must equal that of the input map

---

### EXAMPLE:

```
call mpi_comm_rank(comm, myid, ierr)
if (myid == root) then
    call mpi_initialize_alm_tools(comm, nsmax, nlmax, nmmax,
                                zbounds,polarization, precompute_plms)
    call mpi_map2alm(map, alms)
else
    call mpi_initialize_alm_tools(comm)
    call mpi_map2alm_slave
end
call mpi_cleanup_alm_tools
```

This example 1) initializes the `mpi_alm_tools` module (i.e., allocates internal arrays and defines required parameters), 2) executes a parallel `map2alm` operation, and 3) frees the previously allocated memory.

---

## MODULES & ROUTINES

This section lists the modules and routines used by `mpi_map2alm*`.

`alm_tools`            module

---

## RELATED ROUTINES

This section lists the routines related to `mpi_map2alm*`.

<code>mpi_cleanup_alm_tools</code>	Frees memory that is allocated by the current routine.
<code>mpi_initialize_alm_tools</code>	Allocates memory and defines variables for the <code>mpi_alm_tools</code> module.
<code>mpi_alm2map</code>	Routine for executing a parallel inverse spherical harmonics transform (root processor interface)
<code>mpi_alm2map_slave</code>	Routine for executing a parallel inverse spherical harmonics transform (slave processor interface)
<code>mpi_map2alm_slave</code>	Routine for executing a parallel spherical harmonics transform (slave processor interface)
<code>mpi_alm2map_simple</code>	One-line interface to the parallel inverse spherical harmonics transform
<code>mpi_map2alm_simple</code>	One-line interface to the parallel spherical harmonics transform

---

## mpi\_map2alm\_simple\*

Location in HEALPix directory tree: [src/f90/mod/mpi\\_alm\\_tools.f90](#)

This subroutine provides a simplified (one-line) interface to the MPI version of map2alm. It supports both temperature and polarization inputs in both single and double precision. It must only be run by all processors in the MPI communicator.

---

**FORMAT**                      call    mpi\_map2alm\_simple\*(comm,    map,  
                                 alms, [zbounds], [w8ring])

---

### ARGUMENTS

name & dimensionality	kind	in/out	description
comm	I4B	IN	MPI communicator.
map(0:npix-1,1:nmaps)	SP or DP	IN	input map. If nmaps=1, only temperature information is included; if nmaps=3, polarization information is included
alms(1:nmaps,0:lmax,0:nmax)	SPC or DPC	IN	output alms. nmaps must equal that of the input map
zbounds(1:2)	DP	IN	section of the map on which to perform the $a_{lm}$ analysis, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$ . If $zbounds(1) < zbounds(2)$ , it is performed <i>on</i> the strip $zbounds(1) < z < zbounds(2)$ ; if not, it is performed <i>outside</i> the strip $zbounds(2) \leq z \leq zbounds(1)$ . If absent, the whole map is processed. (OPTIONAL)

---

w8ring_TQU(1:2*nsmax, 1:p)	DP	IN	ring weights for quadrature corrections. If ring weights are not used, this array should be 1 everywhere. p is 1 for a temperature analysis and 3 for (T,Q,U). (OPTIONAL)
----------------------------	----	----	---

---

**EXAMPLE:**

```
call mpi_map2alm_simple(comm, map, alms)
```

This example executes a parallel map2alm operation through the one-line interface. Although all processors must supply allocated arrays to the routine, only the root processor's information will be used as input, and only the root processor's alms will be complete after execution.

**MODULES & ROUTINES**

This section lists the modules and routines used by **mpi\_map2alm\_simple\***.

**alm\_tools**                      module

**RELATED ROUTINES**

This section lists the routines related to **mpi\_map2alm\_simple\***.

<b>mpi_cleanup_alm_tools</b>	Frees memory that is allocated by the current routine.
<b>mpi_initialize_alm_tools</b>	Allocates memory and defines variables for the mpi_alm_tools module.
<b>mpi_alm2map</b>	Routine for executing a parallel inverse spherical harmonics transform (root processor interface)
<b>mpi_alm2map_slave</b>	Routine for executing a parallel inverse spherical harmonics transform (slave processor interface)
<b>mpi_map2alm</b>	Routine for executing a parallel spherical harmonics transform (root processor interface)
<b>mpi_map2alm_slave</b>	Routine for executing a parallel spherical harmonics transform (slave processor interface)



mpi\_alm2map\_simple

One-line interface to the parallel inverse spherical harmonics transform



## mpi\_map2alm\_slave

---

Location in HEALPix directory tree: [src/f90/mod/mpi\\_alm\\_tools.f90](#)

This subroutine complements the master routine `mpi_map2alm`, and should be run by all slaves in the current MPI communicator. It is run without arguments, but after an appropriate call to `initialize_mpi_alm_tools`.

---

**FORMAT**                    `call mpi_map2alm_slave()`

---

### ARGUMENTS

---

None.

---

### EXAMPLE:

```
call mpi_comm_rank(comm, myid, ierr)
if (myid == root) then
    call mpi_initialize_alm_tools(comm, nsmax, nlmax, nmmax,
                                zbounds, polarization, precompute_plms)
    call mpi_map2alm(map, alms)
else
    call mpi_initialize_alm_tools(comm)
    call mpi_map2alm_slave
end
call mpi_cleanup_alm_tools
```

This example 1) initializes the `mpi_alm_tools` module (i.e., allocates internal arrays and defines required parameters), 2) executes a parallel `map2alm` operation, and 3) frees the previously allocated memory.

---

### MODULES & ROUTINES

This section lists the modules and routines used by `mpi_map2alm_slave`.

---

alm\_tools                      module

---

## RELATED ROUTINES

This section lists the routines related to **mpi\_map2alm\_slave**.

<b>mpi_cleanup_alm_tools</b>	Frees memory that is allocated by the current routine.
<b>mpi_initialize_alm_tools</b>	Allocates memory and defines variables for the mpi_alm_tools module.
<b>mpi_alm2map</b>	Routine for executing a parallel inverse spherical harmonics transform (root processor interface)
<b>mpi_alm2map_slave</b>	Routine for executing a parallel inverse spherical harmonics transform (slave processor interface)
<b>mpi_map2alm</b>	Routine for executing a parallel spherical harmonics transform (root processor interface)
<b>mpi_alm2map_simple</b>	One-line interface to the parallel inverse spherical harmonics transform
<b>mpi_map2alm_simple</b>	One-line interface to the parallel spherical harmonics transform

---

# nArguments

Location in HEALPix directory tree: [src/f90/mod/extension.F90](#)

This function emulates the C routine `iargc`, which returns the number of command line arguments provided.

---

**FORMAT**                      `var=nArguments( )`

---

## ARGUMENTS

---

name&dimensionality	kind	in/out	description
var	I4B	OUT	number of command line arguments

---

## RELATED ROUTINES

This section lists the routines related to **nArguments**.

<a href="#">getEnvironment</a>	returns value of environment variable
<a href="#">getArgument</a>	returns list of command line arguments

---

# neighbours\_nest

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

This subroutine returns the number and locations (in terms of pixel numbers) of the topological neighbours of a central pixel. The pixels are ordered in a clockwise sense about the central pixel with the southernmost pixel in first element. For the 4 pixels in the southern corners of the equatorial faces which have two equally southern neighbours the routine returns the southwestern pixel first and proceeds clockwise.

---

**FORMAT**                      call neighbours\_nest([nside](#), [ipix](#), [list](#), [nneigh](#))

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	The $N_{\text{side}}$ parameter of the map.
ipix	I4B/ I8B	IN	The NESTED pixel index of the central pixel.
list(8)	I4B/ I8B	OUT	On exit, the vector of neighbouring pixels. This contains <b>nneigh</b> relevant elements.
nneigh	I4B	OUT	The number of neighbours (mostly 8, except at 8 sites, where there are only 7 neighbours).

---

## EXAMPLE:

```
use pix_tools
integer :: nneigh, list(1:8)
call neighbours_nest(4, 1, list, nneigh)
print*,nneigh
print*,list(1:nneigh)
```

This returns `nneigh`= 8 and a vector `list`, which contains the pixel numbers ( 90, 0, 2, 3, 6, 4, 94, 91).

---

## MODULES & ROUTINES

This section lists the modules and routines used by `neighbours__nest`.

<code>mk_xy2pix, mk_pix2xy</code>	precomputing arrays for the conversion of NESTED pixel numbers to Cartesian coords in each face.
<code>pix2xy__nest, xy2pix__nest</code>	Conversion between NESTED pixel numbers to Cartesian coords in each face.
<b>bit_manipulation</b>	module, containing:
<code>invMSB, invLSB, swapLSBMSB, invswapLSBMSB</code>	functions which manipulate the bit vector which represents the NESTED pixel numbers. They correspond to NorthWest<->SouthEast, SouthWest<->NorthEast, East<->West and North-South flips of the diamond faces, respectively.

---

## RELATED ROUTINES

This section lists the routines related to `neighbours__nest`.

<code>pix2ang, ang2pix</code>	convert between angle and pixel number.
<code>pix2vec, vec2pix</code>	convert between a cartesian vector and pixel number.

---

# nest2uniq

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

This F90 facility turns the parameter  $N_{\text{side}}$  (a power of 2) and the pixel index  $p$  into the Unique ID number  $u = p + 4N_{\text{side}}^2$ . See ["The Unique Identifier scheme"](#) section in ["HEALPix Introduction Document"](#) for more details.

---

**FORMAT**                      call nest2uniq([nside](#), [pnest](#), [puniq](#))

---

## ARGUMENTS

name	kind	in/out	description
nside	I4B	IN	The <b>HEALPix</b> $N_{\text{side}}$ parameter.
pnest	I4B/I8B	IN	(NESTED scheme) pixel identification number over the range $\{0, 12N_{\text{side}}^2 - 1\}$ .
puniq	I4B/I8B	OUT	The <b>HEALPix</b> Unique pixel identifier.

---

## EXAMPLE:

```
use healpix_modules
integer(I4B) :: puniq
call nest2uniq(1, 0, puniq)
print*,puniq
```

returns

4

since the first pixel ( $p = 0$ ) at  $N_{\text{side}} = 1$  is the pixel with Unique ID number 4.

---

## RELATED ROUTINES

This section lists the routines related to **nest2uniq**.

[uniq2nest](#)

] Transforms Unique **HEALPix** pixel ID number into Nside and Nested pixel number

`pix2xxx, ...`

to turn NESTED pixel index into sky coordinates  
and back

---



# npix2nside

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Function to provide the resolution parameter  $N_{\text{side}}$  corresponding to  $N_{\text{pix}}$  pixels over the full sky.

**FORMAT** `var=npix2nside(npix)`

## ARGUMENTS

name & dimensionality	kind	in/out	description
npix	I4B/ I8B	IN	the number $N_{\text{pix}}$ of pixels over the whole sky.
var	I4B	OUT	the parameter $N_{\text{side}}$ . If $N_{\text{pix}}$ is valid (12 times a power of 2 in $\{1, \dots, 2^{28}\}$ ), $N_{\text{side}} = \sqrt{N_{\text{pix}}/12}$ is returned; if not, an error message is issued and -1 is returned.

## EXAMPLE:

```
use healpix_modules
integer(I4B) :: nside
nside= npix2nside(786432)
```

Returns the resolution parameter  $N_{\text{side}}$  (256) corresponding to 786432 pixels on the sky.

## RELATED ROUTINES

This section lists the routines related to **npix2nside**.

[nside2npix](#) returns the number of pixels  $N_{\text{pix}}$  corresponding to resolution parameter  $N_{\text{side}}$



# nside2npix

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Function to provide the number of pixels  $N_{\text{pix}}$  over the full sky corresponding to resolution parameter  $N_{\text{side}}$ .

**FORMAT** `var=nside2npix(nside)`

## ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\text{side}}$ parameter of the map.
var	I8B	OUT	the number of pixels $N_{\text{pix}}$ of the map. If $N_{\text{side}}$ is valid (a power of 2 in $\{1, \dots, 2^{28} = 268435456\}$ ), $N_{\text{pix}} = 12N_{\text{side}}^2$ is returned; if not, an error message is issued and -1 is returned.

## EXAMPLE:

```
use healpix_modules
integer(I8B) :: npix
npix= nside2npix(256)
```

Returns the number of **HEALPix** pixels (786432) for the resolution parameter 256.

## RELATED ROUTINES

This section lists the routines related to **nside2npix**.

[npix2nside](#) returns resolution parameter corresponding to the number of pixels.



# nside2npweights

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Function to return the number of pixel-based weights (in compact form) for a given Nside:

$$N_w = \frac{(N_{\text{side}} + 1)(3N_{\text{side}} + 1)}{4} \simeq \frac{N_{\text{pix}}}{16}$$

---

**FORMAT**                      var=nside2npweights([nside](#))

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\text{side}}$ parameter.
var	I8B	OUT	the number of template pixels $N_w$ .

## EXAMPLE:

```
use healpix\_modules
integer(I8B) :: nw8
nw8 = nside2npweights(256)
```

Returns in **nw8** the number of non-redundant **HEALPix** pixel-based weights (49408) for the resolution parameter 256.

---

## RELATED ROUTINES

This section lists the routines related to **nside2npweights**.

<a href="#">unfold_weightsfile</a>	reads of FITS file containing a list ring-based or pixel-based weights into a full sky map
------------------------------------	--

---

## nside2ntemplates

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Function to provide the number of template pixels

$$N_{\text{template}} = \frac{1 + N_{\text{side}}(N_{\text{side}} + 6)}{4}$$

corresponding to resolution parameter  $N_{\text{side}}$ . Each template pixel has a different shape that *can not* be matched (by rotation or reflexion) to that of any of the other templates.

---

**FORMAT**                      var=nside2ntemplates([nside](#))

---

### ARGUMENTS

---

name & dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\text{side}}$ parameter.
var	I8B	OUT	the number of template pixels $N_{\text{template}}$ .

### EXAMPLE:

```
use healpix_modules
integer(I8B) :: ntpl
ntpl= nside2ntemplates(256)
```

Returns in `ntpl` the number of **HEALPix** template pixels (16768) for the resolution parameter 256.

---

### RELATED ROUTINES

This section lists the routines related to **nside2ntemplates**.

[template\\_pixel\\_ring](#)

---

<code>template_pixel_nest</code>	return the template pixel associated with any <b>HEALPix</b> pixel
<code>same_shape_pixels_ring</code>	
<code>same_shape_pixels_nest</code>	return the ordered list of pixels having the same shape as a given pixel template

---

## number\_\_of\_\_alms

Location in HEALPix directory tree: **src/f90/mod/fitstools.F90**

This function returns the number of  $a_{\ell m}$  values stored in each FITS extension in a FITS file containing  $a_{\ell m}$

---

**FORMAT**                      var=number\_\_of\_\_alms(filename[, extnum])

---

### ARGUMENTS

---

name & dimensionality	kind	in/out	description
filename(LEN=filenamelen)	CHR	IN	filename of the FITS-file containing $a_{\ell m}$ .
extnum	I4B	OUT	number of extensions in the file

---

### EXAMPLE:

---

```
print*,number_of_alms('alms.fits')
```

Prints the number of  $a_{\ell m}$  stored in each extension of the file 'alms.fits'

---

### MODULES & ROUTINES

This section lists the modules and routines used by **number\_\_of\_\_alms**.

<b>fitstools</b>	module, containing:
<b>printerror</b>	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.



---

## RELATED ROUTINES

This section lists the routines related to **number\_of\_alms**.

**fits2alms**, **read\_conbintab**      routines that read  $a_{\ell m}$  values from FITS files.

---

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine writes a full sky **HEALPix** map into a FITS file. The map can be either single or double precision real. It *has* to be 2-dimensional.

```

FORMAT      call  output_map*(map,  header,  file-
               name[,extno])

```

## ARGUMENTS

name & dimensionality	kind	in/out	description
map(0:,1:)	SP/ DP	IN	full sky map(s) to be output
header(LEN=80)(1:)	CHR	IN	string array containing the FITS header to be included in the file
filename(LEN=*)	CHR	IN	filename of the FITS-file to contain <b>HEALPix</b> map(s).
<i>extno</i>	I4B	IN	extension number in which to write the data (0 based). ( <b>default:</b> 0)

**EXAMPLE:**

```
use healpix_types
use fits_tools, only : output_map
real(sp), dimension(0:12*16**2-1, 1:1) :: map
character(len=80), dimension(1:10) :: header
header(:) = "
map(:,:) = 1.
call output_map(map, header, 'map.fits')
```

generates a simple map (made of 1s) and outputs it into the FITS file `map.fits`

---

## MODULES & ROUTINES

This section lists the modules and routines used by **output\_map\***.

<b>fitstools</b>	module, containing:
printererror	routine for printing FITS error messages.
write_bintab	routine to write a binary table into a FITS file.
<b>cfitsio</b>	library for FITS file handling.

---

## RELATED ROUTINES

This section lists the routines related to **output\_map\***.

anafast	executable that reads a <b>HEALPix</b> map from a FITS file and analyses it.
synfast	executable that generate full sky <b>HEALPix</b> maps
<input_map< input=""></input_map<>	subroutine to read a <b>HEALPix</b> map from a a FITS file
write_bintabh	subroutine to write a large array into a FITS file piece by piece
<input_tod*< input=""></input_tod*<>	subroutine to read an arbitrary subsection of a large binary table
<input_minimal_header< input=""></input_minimal_header<>	routine to write minimal FITS header

---

## **parse\_\_init, parse\_\_int, ..., parse\_\_finish**

---

Location in HEALPix directory tree: [src/f90/mod/paramfile\\_io.F90](#)

The Fortran90 module paramfile\_io contains functions to obtain parameters from parameter files or interactively

---

## **ARGUMENTS**

name&dimensionality	kind	in/out	description
fname	CHR	IN	file containing the simulation parameters. If empty, parameters are obtained interactively.
handle	PMF	INOUT	Object of type (paramfile_handle) used to store parameter information
keyname	CHR	IN	name of the required parameter
default	XXX	IN	optional argument containing the default value for a given simulation parameter; must be of appropriate type.
vmin	XXX	IN	optional argument containing the minimum value for a given simulation parameter; must be of appropriate type.
vmax	XXX	IN	optional argument containing the maximum value for a given simulation parameter; must be of appropriate type.
descr	CHR	IN	optional argument containing a description of the required simulation parameter
filestatus	CHR	IN	optional argument. If present, the parameter must be a valid filename. If filestatus=='new', the file must not exist; if filestatus=='old', the file must exist.
code	CHR	IN	optional argument. Contains the name of the executable.
silent	LGT	IN	optional argument. If set to <code>.true.</code> the parsing routines will run silently in non-interactive mode (except for warning or error messages, which will always appear). This is mainly intended for MPI usage where many processors parse the same parameter file: <code>silent</code> can be set to <code>.true.</code> on all CPUs except one.

---

**ROUTINES:**

```
handle = parse_init (fname [,silent])
```

initializes the parser to work on the file fname, or interactively, if fname is empty

```
intval = parse_int (handle, keyname [, default, vmin, vmax, descr])
```

```
longval = parse_long (handle, keyname [, default, vmin, vmax, descr])
```

```
realval = parse_real (handle, keyname [, default, vmin, vmax, descr])
```

```
doubleval = parse_double (handle, keyname [, default, vmin, vmax, descr])
```

```
stringval = parse_string (handle, keyname [, default, descr, filestatus])
```

```
logicval = parse_lgt (handle, keyname [, default, descr])
```

These routines obtain integer(i4b), integer(i8b), real(sp), real(dp), character(len=\*) and logical values, respectively.

Note: `parse_string` will expand all environment variables of the form `${XXX}` (eg: `${HOME}`). It will also replace a *leading ~/* by the user's home directory.

```
call parse_summarize (handle [, code])
```

if the parameters were set interactively, this routine will print out a parameter file performing the same settings.

```
call parse_check_unused (handle [, code])
```

if a parameter file was read, this routine will print out all the parameters found in the file but not used by the code. Intended at detecting typos in parameter names.

```
call parse_finish (handle)
```

frees the memory

---

**EXAMPLE:**

```
program who_r_u
use healpix_types
use paramfile_io
use extension
```

```
implicit none
type(paramfile_handle) :: handle
character(len=256) :: parafilename, name
real(DP) :: age
```

```
parafilename = "
if (nArguments() == 1) call getArguments(1, parafilename)
handle = parse_init(parafilename)
name = parse_string(handle, 'name',descr='Enter your last name: ')
age = parse_double(handle, 'age', descr='Enter your age in years: ', &
```

```
& default=18.d0,vmin=0.d0)
call parse_summarize(handle, 'who_r_u')
end program who_r_u
```

If a file is provided as command line argument when running the executable `who_r_u`, that file will be parsed in search of the lines starting with 'name =' and 'age =', otherwise the same questions will be asked interactively.

---

## RELATED ROUTINES

This section lists the routines related to `parse_init`, `parse_int`, ..., `parse_finish`.

<code>concatnl</code>	generates from a set of strings the multi-line description
<code>nArguments</code>	returns the number of command line arguments
<code>getArgument</code>	returns the list of command line arguments

---

## pixel\_window

Location in HEALPix directory tree: [src/f90/mod/alm\\_tools.F90](#)

This routine returns the *averaged*  $\ell$ -space window function  $w_{\text{pix}}(\ell)$  (for temperature and polarisation) associated to **HEALPix** pixels of resolution parameter  $N_{\text{side}}$ . Because of the integration of the signal over the pixel area, the  $a_{\ell m}^{(\text{pix})}$  coefficients of a pixelated map are related to the unpixelated underlying  $a_{\ell m}$  by  $a_{\ell m}^{(\text{pix})} = a_{\ell m} w_{\text{pix}}(\ell)$ .

Unless specified otherwise, the  $w_{\text{pix}}(\ell)$  are read from the files \$HEALPIX/data/pixel\_window\_n????.fits.

---

**FORMAT**                      call pixel\_window([pixlw](#), [nside](#)[, [windowfile](#)])

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
pixlw(0:lmax,1:p)	DP	OUT	pixel window function(s) $w_{\text{pix}}(\ell)$ generated. The first index must be $\ell_{\text{max}} \leq 4N_{\text{side}}$ . The second index runs from 1:1 for temperature only, and 1:3 for polarisation. In the latter case, 1=T, 2=E, 3=B.
nside	I4B	IN	<b>HEALPix</b> $N_{\text{side}}$ resolution parameter. Unless <a href="#">windowfile</a> is set, the file associated with $N_{\text{side}}$ and shipped with the package is read by default. If <a href="#">nside</a> = 0, the pixel is assumed infinitely small and <a href="#">pixlw</a> is returned with value 1.
windowfile (OPTIONAL)	CHR	IN	FITS file containing the pixel window to be read instead of the default.

---

## EXAMPLE:

```
call pixel_window(pixlw, 64)
```



returns in `pixlw` the pixel window function for  $N_{\text{side}} = 64$ .

---

## MODULES & ROUTINES

This section lists the modules and routines used by **pixel\_window**.

<b>misc_utils</b>	module, containing:
<code>assert</code> , <code>fatal_error</code>	interrupt code in case of error
<b>extension</b>	module, containing:
<code>getEnvironment</code>	read environment variable
<b>fitstools</b>	module, containing:
<code>read_dbintab</code>	reads double precision binary table

---

## RELATED ROUTINES

This section lists the routines related to **pixel\_window**.

<code>gaussbeam</code>	routine to generate a gaussian beam window function
<code>generate_beam</code>	returns a beam window function
<code>alter_alm</code> , <code>rotate_alm</code>	modifies $a_{\ell m}$ to emulate effect of real space filtering and coordinate rotation respectively
<code>alm2map</code>	synthetize a <b>HEALPix</b> map from its $a_{\ell m}$ (or $a_{\ell m}^{(\text{pix})}$ ).
<code>alm2map_der</code>	synthetize a map and its derivatives from its $a_{\ell m}$ (or $a_{\ell m}^{(\text{pix})}$ ).

---

# **pix2xxx,ang2xxx,vec2xxx, nest2ring,ring2nest**

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

The Fortran90 module `pix_tools` contains some subroutines to convert between pixel number in the **HEALPix** map and  $(\theta, \phi)$  or  $(x, y, z)$  coordinates on the sphere. Some of these routines are listed here.

## **ARGUMENTS**

name & dimensionality	kind	in/out	description
nside	I4B	IN	$N_{\text{side}}$ parameter for the <b>HEALPix</b> map.
ipnest	I4B/ I8B	—	pixel identification number in NESTED scheme over the range $\{0, N_{\text{pix}} - 1\}$ .
ipring	I4B/ I8B	—	pixel identification number in RING scheme over the range $\{0, N_{\text{pix}} - 1\}$ .
theta	DP	—	colatitude in radians measured southward from north pole in $\{0, \pi\}$ .
phi	DP	—	longitude in radians, measured eastward in $[0, 2\pi]$ .
vector(3)	DP	—	three dimensional cartesian position vector $(x, y, z)$ . The north pole is $(0, 0, 1)$ . An output vector is normalised to unity.
vertex(3,4) OPTIONAL	DP	OUT	three dimensional cartesian position vectors $(x, y, z)$ (normalised to unity) pointing to the 4 vertices of a given pixel. The four vertices are listed in the order North, West, South, East.

---

**ROUTINES:**

call pix2ang\_ring(nside, ipring, theta, phi)

renders **theta** and **phi** coordinates of the nominal pixel center given the pixel number **ipring** and a map resolution parameter **nside**.

call pix2vec\_ring(nside, ipring, vector [,vertex])

renders cartesian vector coordinates of the nominal pixel center given the pixel number **ipring** and a map resolution parameter **nside**. Optionally renders cartesian vector coordinates of the considered pixel four vertices.

call ang2pix\_ring(nside, theta, phi, ipring)

renders the pixel number **ipring** for a pixel which, given the map resolution parameter **nside**, contains the point on the sphere at angular coordinates **theta** and **phi**.

call vec2pix\_ring(nside, vector, ipring)

renders the pixel number **ipring** for a pixel which, given the map resolution parameter **nside**, contains the point on the sphere at cartesian coordinates **vector**.

call pix2ang\_nest(nside, ipnest, theta, phi)

renders **theta** and **phi** coordinates of the nominal pixel center given the pixel number **ipnest** and a map resolution parameter **nside**.

call pix2vec\_nest(nside, ipnest, vector [,vertex])

renders cartesian vector coordinates of the nominal pixel center given the pixel number **ipnest** and a map resolution parameter **nside**. Optionally renders cartesian vector coordinates of the considered pixel four vertices.

call ang2pix\_nest(nside, theta, phi, ipnest)

renders the pixel number **ipnest** for a pixel which, given the map resolution parameter **nside**, contains the point on the sphere at angular coordinates **theta** and **phi**.

call vec2pix\_nest(nside, vector, ipnest)

renders the pixel number **ipnest** for a pixel which, given the map resolution parameter **nside**, contains the point on the sphere at cartesian coordinates **vector**.

call nest2ring(nside, ipnest, ipring)

performs conversion from NESTED to RING pixel number.

call ring2nest(nside, ipring, ipnest)

performs conversion from RING to NESTED pixel number.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **pix2xxx,ang2xxx,vec2xxx,nest2ring,ring2nest**.

<code>mk_pix2xy, mk_xy2pix</code>	routines used in the conversion between pixel values and “cartesian” coordinates on the Healpix face.
-----------------------------------	---

---

## RELATED ROUTINES

This section lists the routines related to **pix2xxx,ang2xxx,vec2xxx,nest2ring,ring2nest**.

<code>neighbours_nest</code>	find neighbouring pixels.
<code>ang2vec</code>	convert $(\theta, \phi)$ spherical coordinates into $(x, y, z)$ cartesian coordinates.
<code>vec2ang</code>	convert $(x, y, z)$ cartesian coordinates into $(\theta, \phi)$ spherical coordinates.
<code>convert_inplace</code>	in-place conversion between RING and NESTED for integer/real/double maps.
<code>convert_nest2ring</code>	convert from NESTED to RING scheme using a temporary array.
<code>nest2uniq, uniq2nest</code>	conversion of standard pixel index to/from Unique ID number

---

# planck\_rng

Location in HEALPix directory tree: [src/f90/mod/rngmod.f90](#)

The derived type `planck_rng` is used by the Random Number Generation routines `rand_init`, `rand_uni`, `rand_gauss` to describe fully the current RNG sequence.

Most users do not need to know about the `planck_rng` definition. It may be useful for those wanting to take a snapshot of the RNG sequence they are using (by eg, dumping the latest values of `planck_rng` structure on disk) so that the same sequence can be resumed later on from that same point.

The type `planck_rng` is a structure defined as

name	type	definition
x, y, z, w	I4B	internal variables of uniform RNG
gset	DP	internal variable for Gaussian RNG
empty	LGT	flag used by Gaussian RNG

## RELATED ROUTINES

This section lists the routines related to `planck_rng`.

<code>rand_gauss</code>	function which returns a random normal deviate.
<code>rand_uni</code>	function which returns a random uniform deviate.
<code>rand_init</code>	subroutine to initiate a random number sequence.

## plm\_gen

Location in HEALPix directory tree: [src/f90/mod/alm\\_tools.F90](#)

This routine computes the latitude dependent part  $\lambda_{\ell m}$  of the spherical harmonics ( $Y_{\ell m}(\theta, \phi) = \lambda_{\ell m}(\theta)e^{im\phi}$ ) of spin 0 and 2 (see **HEALPix** primer) used to synthesize or analyze **HEALPix** maps of temperature and polarisation. Since version 2.20, which introduced optimized algorithms for spherical harmonic transforms, it has become obsolete and should no longer be used.

**FORMAT**                      call plm\_gen([nsmax](#), [nlmax](#), [nmmax](#), [plm](#))

## ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	The $N_{\text{side}}$ value for which to compute the $\lambda_{\ell m}$ .
nlmax	I4B	IN	The maximum multipole order $\ell$ of the generated $\lambda_{\ell m}$ .
nmmax	I4B	IN	The maximum degree $m$ of the generated $\lambda_{\ell m}$ .
plm(0:n_plm-1, 1:p)	DP	OUT	The $\lambda_{\ell m}$ values, either for temperature only ( $p = 1$ ) or temperature and polarisation ( $p = 3$ ). The number of $\lambda_{\ell m}$ is $n\_plm = nsmax*(nmmax+1)*(2*nlmax-nmmax+2)$ . They are stored in the order of increasing order $\ell$ , increasing degree $m$ , for all the <b>HEALPix</b> ring colatitudes $\theta$ from North Pole to Equator, ie $\lambda_{00}(\theta_1)$ , $\lambda_{10}(\theta_1)$ , $\lambda_{20}(\theta_1)$ , $\dots$ , $\lambda_{11}(\theta_1)$ , $\lambda_{21}(\theta_1)$ ; $\dots$ , $\lambda_{00}(\theta_2) \dots$

---

**EXAMPLE:**

```

use healpix_types
use alm_tools, only : plm_gen
integer(I4B) :: nside, lmax, mmax, n_plm
real(DP), dimension(:,:), allocatable :: plm
...
nside=256 ; lmax=512 ; mmax=lmax
npix=nside2npix(nside)
n_plm=nside*(mmax+1)*(2*lmax-mmax+2)
allocate(plm(0:n_plm-1,1:3))
...
call plm_gen(nside, lmax, mmax, plm)

```

Computes the spherical harmonics for temperature and polarisation for  $N_{\text{side}} = 256$ , up to 512 in  $\ell$  and  $m$ .

---

**MODULES & ROUTINES**

This section lists the modules and routines used by **plm\_gen**.

compute_lam_mm, get_pixel_layout,	
gen_lamfac, gen_mfac, gen_normpol,	
gen_recfac, init_rescale, l_min_ylm	Ancillary routines used for $\lambda_{\ell m}$ recursion
<b>misc_utils</b>	module, containing:
assert_alloc	routine to print error message, when an array can not be allocated properly

---

**RELATED ROUTINES**

This section lists the routines related to **plm\_gen**.

<b>alm2map</b>	routine generating maps of temperature and polarisation from their $a_{\ell m}$ that can use precomputed $\lambda_{\ell m}$ generated by plm_gen
<b>map2alm</b>	routine analysing maps of temperature and polarisation that can use precomputed $\lambda_{\ell m}$ generated by plm_gen

plngen	executable using plm_gen to compute the $\lambda_{\ell m}$ and writting them on disc
--------	---

---



# query\_disc

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Routine to find the index of all pixels within an angular distance radius from a defined center. The output indices can be either in the RING or NESTED scheme

---

**FORMAT**                      call query\_disc([nside](#), [vector0](#), [radius](#), [listpix](#),  
                                  [nlist](#)[, [nest](#), [inclusive](#)])

---

## ARGUMENTS

name & dimensional- ity	kind	in/out	description
nside	I4B	IN	the $N_{\text{side}}$ parameter of the map.
vector0(3)	DP	IN	cartesian vector pointing at the disc center.
radius	DP	IN	disc radius in radians.
listpix(0:*)	I4B/ I8B	OUT	the index for all pixels within <a href="#">radius</a> . Make sure that the size of the array is big enough to contain all pixels.
nlist	I4B/ I8B	OUT	The number of pixels listed in <a href="#">listpix</a> .
nest (OPTIONAL)	I4B	IN	The pixel indices are in the NESTED num- bering scheme if <a href="#">nest</a> =1, and in RING scheme otherwise.
inclusive (OPTIONAL)	I4B	IN	If set to 1, all the pixels overlapping (even partially) with the disc are listed, other- wise only those whose center lies within the disc are listed.

---

## EXAMPLE:

```
use healpix_modules
call query_disc(256,(/0,0,1/),pi/2,listpix,nlist,nest=1)
```

Returns the NESTED pixel index of all pixels north of the equatorial line in a  $N_{\text{side}} = 256$  map.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **query\_disc**.

<b>in_ring</b>	routine to find the pixels in a certain slice of a given ring.
<b>ring_num</b>	function to return the ring number corresponding to the coordinate $z$

---

## RELATED ROUTINES

This section lists the routines related to **query\_disc**.

<b>pix2ang, ang2pix</b>	convert between angle and pixel number.
<b>pix2vec, vec2pix</b>	convert between a cartesian vector and pixel number.
<b>query_disc, query_polygon, query_strip, query_triangle</b>	render the list of pixels enclosed respectively in a given disc, polygon, latitude strip and triangle
<b>surface_triangle</b>	computes the surface in steradians of a spherical triangle defined by 3 vertices

---

# query\_polygon

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Routine to find the index of all pixels enclosed in a polygon. The polygon should be convex, or have only one concave vertex. The edges should not intersect each other. The output indices can be either in the RING or NESTED scheme

---

**FORMAT**                      call query\_polygon([nside](#), [vlist](#), [nv](#), [listpix](#),  
                                  [nlist](#)[, [nest](#), [inclusive](#)])

---

## ARGUMENTS

name & dimension-ality	kind	in/out	description
nside	I4B	IN	the $N_{\text{side}}$ parameter of the map.
vlist(1:3,0:*)	DP	IN	cartesian vector pointing at polygon respective vertices.
nv	I4B	IN	number of vertices, should be equal to 3 or larger.
listpix(0:*)	I4B/ I8B	OUT	the index for all pixels enclosed in the triangle. Make sure that the size of the array is big enough to contain all pixels.
nlist	I4B/ I8B	OUT	The number of pixels listed in <code>listpix</code> .
nest (OPTIONAL)	I4B	IN	The pixel indices are in the NESTED numbering scheme if <code>nest=1</code> , and in RING scheme otherwise.
inclusive (OPTIONAL)	I4B	IN	If set to 1, all the pixels overlapping (even partially) with the polygon are listed, otherwise only those whose center lies within the polygon are listed.

---

## EXAMPLE:

```

use healpix_modules
real(dp), dimension(1:3,0:9) :: vertices
vertices(:,0) = (/0.,0.,1./) ! +z
vertices(:,1) = (/1.,0.,0./) ! +x
vertices(:,2) = (/1.,1.,-1./) ! x+y-z
vertices(:,3) = (/0.,1.,0./) ! +y

call query_polygon(256,vertices,4,listpix,nlist,nest=0)

```

Returns the RING pixel index of all pixels in the polygon with vertices of cartesian coordinates (0,0,1), (1,0,0), (1,1,-1) and (0,1,0) in a  $N_{\text{side}} = 256$  map.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **query\_polygon**.

isort	routine to sort integer number
query_triangle	render the list of pixels enclosed in a given triangle
surface_triangle	computes the surface of a spherical triangle defined by 3 vertices
vect_prod	routine to compute the vectorial product of two 3D vectors

---

## RELATED ROUTINES

This section lists the routines related to **query\_polygon**.

pix2ang, ang2pix	convert between angle and pixel number.
pix2vec, vec2pix	convert between a cartesian vector and pixel number.
query_disc, query_polygon, query_strip, query_triangle	render the list of pixels enclosed respectively in a given disc, polygon, latitude strip and triangle
surface_triangle	computes the surface in steradians of a spherical triangle defined by 3 vertices

---

# query\_strip

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Routine to find the index of all pixels enclosed in a latitude strip.  
The output indices can be either in the RING or NESTED scheme

---

**FORMAT**                    call query\_strip([nside](#), [theta1](#), [theta2](#), [listpix](#),  
                              [nlist](#)[, [nest](#), [inclusive](#)])

---

## ARGUMENTS

name&dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\text{side}}$ parameter of the map.
theta1	DP	IN	colatitude lower bound in radians measured from North Pole (between 0 and $\pi$ ).
theta2	DP	IN	colatitude upper bound in radians measured from North Pole (between 0 and $\pi$ ). If $\text{theta1} < \text{theta2}$ , the pixels lying in $[\text{theta1}, \text{theta2}]$ are output, otherwise, the pixel lying in $[0, \text{theta2}]$ and those lying in $[\text{theta1}, \pi]$ are output.
listpix(0:*)	I4B/ I8B	OUT	the index for all pixels enclosed in the strip(s). Make sure that the size of the array is big enough to contain all pixels.
nlist	I4B/ I8B	OUT	The number of pixels listed in <code>listpix</code> .
nest (OPTIONAL)	I4B	IN	The pixel indices are in the NESTED numbering scheme if <code>nest=1</code> , and in RING scheme otherwise.
inclusive (OPTIONAL)	I4B	IN	If set to 1, all the pixels overlapping (even partially) with the strip are listed; otherwise only those whose center lies within the strip are listed.

---

**EXAMPLE:**

```
call query_strip(256,0.75*PI,0.2*PI,listpix,nlist,nest=1)
```

Returns the NESTED pixel index of all pixels with colatitude in  $[0, \pi/5]$  and those with colatitude in  $[3\pi/4, \pi]$

---

**MODULES & ROUTINES**

This section lists the modules and routines used by **query\_strip**.

<b>in_ring</b>	routine to find the pixels in a certain slice of a given ring.
<b>intrs_intrv</b>	routine to compute the intersection of 2 intervals on a circle
<b>ring_num</b>	function to return the ring number corresponding to the coordinate $z$
<b>vect_prod</b>	routine to compute the vectorial product of two 3D vectors

---

**RELATED ROUTINES**

This section lists the routines related to **query\_strip**.

<b>pix2ang, ang2pix</b>	convert between angle and pixel number.
<b>pix2vec, vec2pix</b>	convert between a cartesian vector and pixel number.
<b>query_disc, query_polygon,</b> <b>query_strip, query_triangle</b>	render the list of pixels enclosed respectively in a given disc, polygon, latitude strip and triangle
<b>surface_triangle</b>	computes the surface in steradians of a spherical triangle defined by 3 vertices

---

# query\_triangle

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Routine to find the index of all pixels enclosed in a spherical triangle described by its three vertices. The output indices can be either in the RING or NESTED scheme

---

**FORMAT**                      call query\_triangle([nside](#), [v1](#), [v2](#), [v3](#), [listpix](#),  
                                  [nlist](#)[, [nest](#), [inclusive](#)])

---

## ARGUMENTS

name&dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\text{side}}$ parameter of the map.
v1(3)	DP	IN	cartesian vector pointing at the triangle first vertex.
v2(3)	DP	IN	cartesian vector pointing at the triangle second vertex.
v3(3)	DP	IN	cartesian vector pointing at the triangle third vertex.
listpix(0:*)	I4B/ I8B	OUT	the index for all pixels enclosed in the triangle. Make sure that the size of the array is big enough to contain all pixels.
nlist	I4B/ I8B	OUT	The number of pixels listed in <code>listpix</code> .
nest (OPTIONAL)	I4B	IN	The pixel indices are in the NESTED numbering scheme if <code>nest=1</code> , and in RING scheme otherwise.
inclusive (OPTIONAL)	I4B	IN	If set to 1, all the pixels overlapping (even partially) with the triangle are listed, otherwise only those whose center lies within the triangle are listed.

---

## EXAMPLE:

```
call query_triangle(256, (/1,0,0/), (/0,1,0/), (/0,0,1/), listpix, nlist)
```

Returns the RING pixel index of the (98560) pixels in the octant  $(x, y, z > 0)$  in a  $N_{\text{side}} = 256$  map.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **query\_triangle**.

<b>in_ring</b>	routine to find the pixels in a certain slice of a given ring.
<b>intrs_intrv</b>	routine to compute the intersection of 2 intervals on a circle
<b>ring_num</b>	function to return the ring number corresponding to the coordinate $z$
<b>vect_prod</b>	routine to compute the vectorial product of two 3D vectors

---

## RELATED ROUTINES

This section lists the routines related to **query\_triangle**.

<b>pix2ang, ang2pix</b>	convert between angle and pixel number.
<b>pix2vec, vec2pix</b>	convert between a cartesian vector and pixel number.
<b>query_disc, query_polygon, query_strip, query_triangle</b>	render the list of pixels enclosed respectively in a given disc, polygon, latitude strip and triangle
<b>surface_triangle</b>	computes the surface in steradians of a spherical triangle defined by 3 vertices

---



# rand\_gauss

Location in HEALPix directory tree: [src/f90/mod/rngmod.f90](#)

This routine returns a number out of a pseudo-random normal deviate (ie, its probability distribution is a Gaussian of mean 0 and variance 1).

---

**FORMAT**                      var=rand\_gauss([rng\\_handle](#))

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
rng_handle	planck_rng	INOUT	structure of type <code>planck_rng</code> containing on all information necessary to continue same random sequence.
var	DP	OUT	number belonging to a pseudo-random normal deviate.

---

## EXAMPLE:

```
use healpix_types
use rngmod
type(planck_rng) :: rng_handle
real(dp) :: gauss
```

```
call rand_init(rng_handle, 12345, 6789012)
gauss = rand_gauss(rng_handle)
```

initiates a random sequence with the pair of seeds (12345, 6789012), and generates one number out of the normal deviate.

---

## RELATED ROUTINES

This section lists the routines related to **rand\_gauss**.

<code>planck_rng</code>	derived type describing RNG state
<code>rand_uni</code>	function which returns a random uniform deviate.
<code>rand_init</code>	subroutine to initiate a random number sequence.

---

# rand\_init

Location in HEALPix directory tree: [src/f90/mod/rngmod.f90](#)

This routine initializes, with up to 4 seeds, a random number sequence. The generator being primed is an F90 port of an xorshift generator described in Marsaglia, Journal of Statistical Software 2003, vol 8. It has a theoretical period of  $2^{128} - 1 \approx 3.410^{38}$ . Please refer to the “Comment on Random Number Generator” in the Fortran90 facilities guidelines.

---

**FORMAT**                    call rand\_init([rng\\_handle](#), [[seed1](#), [seed2](#),  
[seed3](#), [seed4](#)])

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
rng_handle	planck_rng	OUT	structure of type <code>planck_rng</code> containing on output all information necessary to continue same random sequence.
seed1 (OPTIONAL)	I4B	IN	first seed of the random sequence. Can be of arbitrary sign. If set to zero or not provided will be replaced internally by a non-zero hard coded value.
seed2 (OPTIONAL)	I4B	IN	second seed. Same properties as above
seed3 (OPTIONAL)	I4B	IN	third seed. Same as above.
seed4 (OPTIONAL)	I4B	IN	fourth seed. Same as above.

## EXAMPLE:

```
use rngmod
type(planck_rng) :: rng_handle
call rand_init(rng_handle, 12345, 6789012)
```

initiates a random sequence with the pair of seeds (12345, 6789012).

---

## RELATED ROUTINES

This section lists the routines related to **rand\_init**.

<code>planck_rng</code>	derived type describing RNG state
<code>rand_gauss</code>	function which returns a random normal deviate.
<code>rand_uni</code>	function which returns a random uniform deviate.

---

# rand\_uni

Location in HEALPix directory tree: [src/f90/mod/rngmod.f90](#)

This routine returns a number out of a pseudo-random uniform deviate (ie, its probability distribution is uniform in the range ]0,1[).

---

**FORMAT**                      var=rand\_uni([rng\\_handle](#))

---

## ARGUMENTS

---

name & dimensionality	kind	in/out	description
rng_handle	planck_rng	INOUT	structure of type <code>planck_rng</code> containing on all information necessary to continue same random sequence.
var	DP	OUT	number belonging to a pseudo-random uniform deviate.

---

## EXAMPLE:

```
use healpix_types
use rngmod
type(planck_rng) :: rng_handle
real(dp) :: uni
```

```
call rand_init(rng_handle, 12345, 6789012)
uni = rand_uni(rng_handle)
```

initiates a random sequence with the pair of seeds (12345, 6789012), and generates one number out of the uniform deviate.

---

## RELATED ROUTINES

This section lists the routines related to **rand\_uni**.

<code>planck_rng</code>	derived type describing RNG state
<code>rand_gauss</code>	function which returns a random normal deviate.
<code>rand_init</code>	subroutine to initiate a random number sequence.

---

## read\_asctab\*

---

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine is obsolete, use [fits2cl](#) instead

---

## read\_\_bintab\*

---

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine reads a **HEALPix** map from a binary FITS-file.  
The routine can read a temperature map or both temperature  
and polarisation maps (T,Q,U)

---

<b>FORMAT</b>	call read__bintab*( <i>filename</i> , <i>map</i> , <i>npixtot</i> , <i>nmaps</i> , <i>nullval</i> , <i>anynull</i> [, <i>header</i> , <i>units</i> , <i>extno</i> ])
---------------	---

Arguments appearing in *italic* are optional.

---

## ARGUMENTS



name & dimensionality	kind	in/out	description
filename(LEN=filenamelen)	CHR	IN	filename of FITS-file containing the map(s).
npixtot	I4B	IN	Number of pixels to be read from map.
nmaps	I4B	IN	number of maps to be read, 1 for temperature only, and 3 for (T,Q,U).
map(0:npixtot-1,1:nmaps)	SP/ DP	OUT	the map read from the FITS-file.
nullval	SP/ DP	OUT	value of missing pixels in the map.
anynull	LGT	OUT	.TRUE., if there are missing pixels, and .FALSE. otherwise.
header(LEN=80)(1:) (OPTIONAL)	CHR	OUT	character string array containing the FITS header read from the file. Its dimension has to be defined prior to calling the routine
units(LEN=*)(1:nmaps)	CHR	OUT	character string array containing the physical units of each map read
extno	I4B	IN	extension number to read the data from (0 based).( <b>default:</b> 0) (the first extension is read)

---

**EXAMPLE:**

```
call read_bintab ('map.fits', map, 12*32**2, 1, nullval, anynull)
```

Reads a **HEALPix** temperature map from the file 'map.fits' to the array `map(0:12*32**2-1,1:1)`. The pixel number  $12*32**2$  is the number of pixels in a  $N_{\text{side}} = 32$  **HEALPix** map. If there are missing pixels in the input file (with value NaN (Not a Number),  $\pm\text{Infinity}$ , or matching the FITS keyword `BAD_DATA`) then `anynull` is `.TRUE.` and these pixels get the value returned in `nullval`.

---

**MODULES & ROUTINES**

This section lists the modules and routines used by `read__bintab*`.

<b>fitstools</b>	module, containing:
<b>prnterror</b>	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.

---

**RELATED ROUTINES**

This section lists the routines related to `read__bintab*`.

<b>input_map</b>	Routine which reads a map using <code>read__bintab*</code> and fills missing pixels with a given value.
<b>map2alm</b>	Routine which analyse a map and returns the $a_{lm}$ coefficients.
<b>read_fits_cut4</b>	Routine to read cut sky <b>HEALPix</b> FITS maps
<b>write_plm, write_bintab</b>	Routines to write <b>HEALPix</b> FITS maps

---

# read\_conbintab\*

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine reads a FITS file containing  $a_{\ell m}$  values for constrained realisation. The FITS file is supposed to contain one integer column with  $index = \ell^2 + \ell + m + 1$  and 2 or 4 single (or double) precision columns with real/imaginary  $a_{\ell m}$  values and real/imaginary standard deviation on these  $a_{\ell m}$ . It is supposed to contain either 1 or 3 extension(s) containing respectively the  $a_{\ell m}$  for T and if applicable E and B.

**FORMAT**                      call read\_conbintab\*([filename](#), [alms](#), [nalms](#),  
[units](#), [extno](#))]

## ARGUMENTS

name&dimensionality	kind	in/out	description
filename(LEN=filenamelen)	CHR	IN	filename of FITS file containing $a_{\ell m}$ .
nalms	I4B	IN	Number of $a_{\ell m}$ values to read from the file.
alms(0:nalms-1,1:6)	SP/ DP	OUT	the $a_{\ell m}$ read from the file. alms(i,1) and alms(i,2) contain the $\ell$ and $m$ values for the ith $a_{\ell m}$ . alms(i,3) and alms(i,4) contain the real and imaginary value of the ith $a_{\ell m}$ . Finally, the standard deviation for the ith $a_{\ell m}$ is contained in alms(i,5) (real) and alms(i,6) (imaginary).
units(len=20)(1:) (OPTIONAL)	CHR	OUT	character string containing the units of the $a_{\ell m}$
extno (OPTIONAL)	I4B	IN	extension (0 based) of the FITS file to be read

---

**EXAMPLE:**

```
call read_conbintab ('alms.fits',alms,65*66/2)
```

Read  $65 \times 66 / 2$  (the number of  $a_{\ell m}$  needed to fill the whole range from  $l=0$  to  $l=64$ )  $a_{\ell m}$  values from the file 'alms.fits' into the array `alms(0:65*66/2-1,1:6)`.

---

**MODULES & ROUTINES**

This section lists the modules and routines used by **read\_conbintab\***.

<b>fitstools</b>	module, containing:
<b>printerror</b>	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.

---

**RELATED ROUTINES**

This section lists the routines related to **read\_conbintab\***.

<b>alms2fits, dump_alms</b>	routines to write $a_{\ell m}$ to a FITS-file
<b>fits2alms</b>	
<b>number_of_alms, getsize_fits</b>	has the same function as <code>read_conbintab</code> but is more general.
	can be used to find out the number of $a_{\ell m}$ available in the file.

---

# read\_dbintab

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine reads a double precision binary array from a FITS-file. It is used by **HEALPix** to read precomputed  $P_{\ell m}(\theta)$  values and pixel window functions.

---

**FORMAT**                      call read\_dbintab(filename, map, npixtot,  
                                 nmaps, nullval, anynull, units)

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=filenamelen)	CHR	IN	filename of FITS-file containing the double precision array.
npixtot	I4B	IN	Number of values to be read from the file.
nmaps	I4B	IN	number of 1-dim. arrays, 1 for scalar $P_{\ell m}$ s and pixel windows, 3 for scalar and tensor $P_{\ell m}$ s.
map(0:npixtot-1,1:nmaps)	DP	OUT	the array read from the FITS-file.
nullval	DP	OUT	value of missing pixels in the array.
anynull	LGT	OUT	TRUE, if there are missing pixels, and FALSE otherwise.
units(len=20)(1:nmaps)	CHR	OUT	respective physical units of the maps in the FITS file.

---

**EXAMPLE:**

```
call read_dbintab ('plm_32.fits',plm,65*66*32,1,nullval,anynull)
```

Reads precomputed scalar  $P_{\ell m}(\theta)$  from the file 'plm\_32.fits'. The values are returned in the array `plm(0:65*66*32,1:1)`. The number of values  $65*66*32$  is the number of precomputed  $P_{\ell m}(\theta)$  for a  $N_{\text{side}} = 32$ ,  $\ell_{\text{max}} = 64$  map. If there are missing values in the file, `anynull` is TRUE and `nullval` contains the values of these pixels.

---

**MODULES & ROUTINES**

This section lists the modules and routines used by `read_dbintab`.

<b>fitstools</b>	module, containing:
<code>prnterror</code>	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.

---

**RELATED ROUTINES**

This section lists the routines related to `read_dbintab`.

<code>plmgen</code>	Executable to create files with precomputed $P_{\ell m}(\theta)$ .
<code>write_plm</code>	Routine to create a file to be read by <code>read_dbintab</code> .

---

# read\_fits\_cut4

Location in HEALPix directory tree: **src/f90/mod/fitstools.F90**

This routine reads a cut sky **HEALPix** map from a FITS file. The format used for the FITS file follows the one used for Boomerang98 and is adapted from COBE/DMR

---

**FORMAT**                      call read\_fits\_cut4(*filename*, *np*, *pixel*, [*signal*,  
                                  *n\_obs*, *error*, *header*, *units*, *extno*])

Arguments appearing in *italic* are optional.

---

## ARGUMENTS

name&dimensionality	kind	in/out	description
<i>filename</i> (LEN= <i>filename</i> len)	CHR	IN	FITS file to be read from, containing a cut sky map
<i>np</i>	I4B	IN	number of pixels to be read from the file
<i>pixel</i> (0:np-1)	I4B	OUT	index of observed (or valid) pixels
<i>signal</i> (0:np-1) (OPTIONAL)	SP	OUT	value of signal in each observed pixel
<i>n_obs</i> (0:np-1)	I4B	OUT	number of observation per pixel
<i>serror</i> (0:np-1)	SP	OUT	<i>rms</i> of signal in pixel. (For white noise, this would be $\propto 1/\sqrt{n\_obs}$ )
<i>header</i> (LEN=80)(1:)	CHR	OUT	FITS extension header
<i>units</i> (LEN=20)	CHR	OUT	maps units (applies only to Signal and Serror, which are assumed to have the same units)
<i>extno</i>	I4B	IN	extension number (0 based) for which map is read. Default = 0 (first extension).

---

## MODULES & ROUTINES

This section lists the modules and routines used by `read_fits_cut4`.

<b>fitstools</b>	module, containing:
<code>prnterror</code>	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.

---

## RELATED ROUTINES

This section lists the routines related to `read_fits_cut4`.

<code>anafast</code>	executable that reads a <b>HEALPix</b> map and analyses it.
<code>synfast</code>	executable that generate full sky <b>HEALPix</b> maps
<code>getsize_fits</code>	routine to know the size of a FITS file and its type (eg, full sky vs cut sky)
<code>input_map</code>	all purpose routine to input a map of any kind from a FITS file
<code>output_map</code>	subroutine to write a FITS file from a <b>HEALPix</b> map
<code>write_fits_cut4</code>	subroutine to write a cut sky map into a FITS file

---



# read\_par

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine reads the ‘NSIDE’, ‘TFIELDS’, ‘MAX-LPOL’, and optionally ‘MAX-MPOL’ keywords from a FITS-file. These keywords describe  $N_{\text{side}}$ , number of datasets (maps) and maximum multipole  $\ell$  (order) and  $m$  (degree) value for the file. If a keyword is not found in the FITS file, a value of -1 is returned instead. The file could eg. be a **HEALPix** map, or a set of  $a_{\ell m}$  or precomputed  $P_{\ell m}(\theta)$

**FORMAT**                      call read\_par( [filename](#), [nside](#), [lmax](#), [tfields](#) [, [mmax](#)] )

## ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=filenamelen)	CHR	IN	filename of the FITS file.
nside	I4B	OUT	‘NSIDE’ keyword value from the FITS header.
lmax	I4B	OUT	‘MAX-LPOL’ keyword value from the FITS header.
tfields	I4B	OUT	‘TFIELDS’ keyword value from the FITS header.
mmax (OPTIONAL)	I4B	OUT	‘MAX-MPOL’ keyword value from the FITS header.

## EXAMPLE:

```
call read_par('plm_128p.fits', nside, lmax, nhar)
```

Checks the  $N_{\text{side}}$  and maximum  $\ell$  value used for the precomputed  $P_{\ell m}(\theta)$  that are stored in the file ‘plm\_128p.fits’. If the file also contains tensor harmonics, nhar is returned with the value 3, otherwise it is 1.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **read\_\_par**.

<b>fitstools</b>	module, containing:
<b>prnterror</b>	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.

---

## RELATED ROUTINES

This section lists the routines related to **read\_\_par**.

synfast, plmgen	executables that produce FITS-files with keywords relevant to this routine.
-----------------	---

---

# real\_fft

Location in HEALPix directory tree: [src/f90/mod/healpix\\_fft.F90](#)

This routine performs a forward or backward Fast Fourier Transformation on its argument `data`.

---

**FORMAT**                      call real\_fft([data](#), [backward](#))

---

## ARGUMENTS

---

name & dimensionality	kind	in/out	description
data(:)	XXX	INOUT	array containing the input and output data. Can be of type real(sp) or real(dp)
backward	LGT	IN	Optional argument. If present and true, perform backward transformation, else forward

---

## EXAMPLE:

```
use healpix_fft
call real_fft (data, backward=.true.)
```

Performs a backward FFT on data.

---

## RELATED ROUTINES

This section lists the routines related to **real\_fft**.

[complex\\_fft](#)                      routine for FFT of complex data

---

## remove\_dipole\*

---

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

This routine provides a means to fit and remove the dipole and monopole from a **HEALPix** map. The fit is obtained by solving the linear system

$$\sum_{j=0}^{d^2-1} A_{ij} f_j = b_i \quad (19)$$

with,  $d = 1$  or  $2$ , and

$$b_i \equiv \sum_{p \in \mathcal{P}} s_i(p) w(p) m(p), \quad (20)$$

$$A_{ij} \equiv \sum_{p \in \mathcal{P}} s_i(p) w(p) s_j(p), \quad (21)$$

where  $\mathcal{P}$  is the set of valid, unmasked pixels,  $m$  is the input map,  $w$  is pixel weighting, while  $s_0(p) = 1$  and  $s_1(p) = x$ ,  $s_2(p) = y$ ,  $s_3(p) = z$  are respectively the monopole and dipole templates. The output map is then

$$m'(p) = m(p) - \sum_{i=0}^{d^2-1} f_i s_i(p). \quad (22)$$

---

**FORMAT**      call remove\_dipole\*([nside](#), [map](#), [ordering](#), [degree](#), [multipoles](#), [zbounds](#)[, [fmissval](#), [mask](#), [weights](#)])

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	value of $N_{\text{side}}$ resolution parameter for input map
map(0:12*nside*nside-1)	SP/ DP	INOUT	<b>HEALPix</b> map from which the monopole and dipole will be removed. Those are removed from <i>all unflagged pixels</i> , even those excluded by the cut <b>zounds</b> or the <b>mask</b> .
ordering	I4B	IN	<b>HEALPix</b> scheme 1:RING, 2:NESTED
degree	I4B	IN	multipoles to fit and remove. It is either 0 (nothing done), 1 (monopole only) or 2 (monopole and dipole).
multipoles(0:degree*degree-1)	DP	OUT	values of best fit monopole and dipole. The monopole is described as a scalar in the same units as the input map, the dipole as a 3D cartesian vector, in the same units.
zbounds(1:2)	DP	IN	section of the map on which to perform the fit, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$ . If $\text{zbounds}(1) < \text{zbounds}(2)$ , it is performed <i>on</i> the strip $\text{zbounds}(1) < z < \text{zbounds}(2)$ ; if not, it is performed <i>outside</i> the strip $\text{zbounds}(2) \leq z \leq \text{zbounds}(1)$ . If absent, the whole map is processed.
fmissval (OPTIONAL)	SP/ DP	IN	value used to flag bad pixel on input ( <b>default:</b> -1.6375e30). Pixels with that value are ignored during the fit, and left unchanged on output.
mask(0:12*nside*nside-1) (OPTIONAL)	SP/ DP	IN	mask of valid pixels. Pixels with $ \text{mask}  < 10^{-10}$ are not used for fit. Note: the map is <i>not</i> multiplied by the mask.
weights(0:12*nside*nside-1) (OPTIONAL)	SP/ DP	IN	weight to be given to each map pixel before doing the fit. By default pixels are given a uniform weight of 1. Note: the output map is <i>not</i> multiplied by the weights.

---

**EXAMPLE:**

```
s = sin(15.0_dp * DEG2RAD)
call remove_dipole*(128, map, 1, 2, multipoles, (\ s, -s \) )
```

Will compute and remove the best fit monopole and dipole from a map with  $N_{\text{side}} = 128$  in RING ordering scheme. The fit is performed on pixels with  $|b| > 15^\circ$ .

---

**MODULES & ROUTINES**

This section lists the modules and routines used by **remove\_dipole\***.

**pix\_tools**            module, containing:

---

**RELATED ROUTINES**

This section lists the routines related to **remove\_dipole\***.

**add\_dipole**            routine to add a dipole and monopole to a map.

---

# ring\_analysis

Location in HEALPix directory tree: [src/f90/mod/alm\\_tools.F90](#)

This subroutine computes the Fast Fourier Transform of a single ring of pixels and extends the computed coefficients up to the maximum  $m$  of the transform.

---

**FORMAT**                    call ring\_analysis([nsmax](#), [nlmax](#), [nmmax](#),  
                              [datain](#), [nph](#), [dataout](#), [kphi0](#))

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	$N_{\text{side}}$ of the map.
nlmax	I4B	IN	Maximum $\ell$ of the analysis.
nmmax	I4B	IN	Maximum $m$ of the analysis.
nph	I4B	IN	The number of points on the ring.
datain(0:nph-1)	DP	IN	Function values on the ring.
dataout(0:nmmax)	DPC	OUT	Fourier components, replicated to $Nmmax$ .
kphi0	I4B	IN	0 if the first pixel on the ring is at $\phi = 0$ ; 1 otherwise.

---

## EXAMPLE:

```
call ring_analysis(64,128,128,datain,8,dataout,0)
```

Returns the periodically extended complex Fourier Transform of datain in dataout. They are returned in the following order: 0 1 2 3 4 5 6 7 6 5 4 3 2 1 0 ... The value  $kphi0 = 0$  specifies that no phase factor needed to be applied, because the ring starts at  $\phi = 0$ .

---

## MODULES & ROUTINES

This section lists the modules and routines used by **ring\_analysis**.

**healpix\_fft**      module.

---

## RELATED ROUTINES

This section lists the routines related to **ring\_analysis**.

**ring\_synthesis**      Inverse transform (complex-to-real), used in  
**alm2map**, **alm2map\_der** and **synfast**

---



# ring\_num

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

This function returns the ring number for a  $z$  coordinate.

---

**FORMAT**                      var=ring\_num([nside](#), [z](#))

---

## ARGUMENTS

name&dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\text{side}}$ parameter of the map.
$z$	DP	IN	the $z$ coordinate to find the ring number for.

---

## EXAMPLE:

```
print*,ring_num(256, 0.5)
```

Prints the ring number of the ring at position  $z = 0.5$ .

---

## MODULES & ROUTINES

This section lists the modules and routines used by **ring\_num**.

None

---

## RELATED ROUTINES

This section lists the routines related to **ring\_num**.

**in\_ring** Returns the pixels in a slice on a given ring.

---

# ring\_synthesis

Location in HEALPix directory tree: [src/f90/mod/alm\\_tools.F90](#)

---

**FORMAT**                      call ring\_synthesis([nsmax](#), [nlmax](#), [nmmax](#),  
                                 [datain](#), [nph](#), [dataout](#), [kphi0](#))

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	$N_{\text{side}}$ of the map.
nlmax	I4B	IN	Maximum $\ell$ of the analysis.
nmmax	I4B	IN	Maximum $m$ of the analysis.
nph	I4B	IN	The number of points on the ring.
datain(0:nmmax)	DPC	IN	Fourier components as computed from the $a_{lm}$ .
dataout(0:nph-1)	DP	OUT	Synthesized function values on the ring.
kphi0	I4B	IN	0 if the first pixel on the ring is at $\phi = 0$ ; 1 otherwise.

---

## EXAMPLE:

```
call ring_synthesis(64,128,128,datain,8,dataout,1)
```

This computes the inverse (complex-to-real) Fast Fourier Transform for the second ring from the pole, containing 8 pixels, for a map resolution of  $N_{\text{side}} = 64$ . 128 complex Fourier components contribute to these 8 pixels. The value  $kphi0 = 1$  specifies that a phase factor needed to be applied to correctly rotate the ring into position on the **HEALPix** grid.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **ring\_synthesis**.

**healpix\_fft**            module.

---

## RELATED ROUTINES

This section lists the routines related to **ring\_synthesis**.

**ring\_analysis**            Forward transform, used in **map2alm** and **anafast**

---

## rotate\_alm\*

Location in HEALPix directory tree: [src/f90/mod/alm\\_tools.F90](#)

This routine transform the scalar (and tensor)  $a_{\ell m}$  coefficients to emulate the effect of an arbitrary rotation of the underlying map. The rotation is done directly on the  $a_{\ell m}$  using the Wigner rotation matrices, computed by recursion. To rotate the  $a_{\ell m}$  for  $\ell \leq \ell_{\max}$  the number of operations scales like  $\ell_{\max}^3$ .

---

**FORMAT**                      call rotate\_alm\*(nlmax, alm\_TGC, psi, theta, phi)

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
nlmax	I4B	IN	maximum $\ell$ value for the $a_{\ell m}$ .
alm_TGC(1:p,0:nlmax,0:nlmax)	SPC/ DPC	INOUT	complex $a_{\ell m}$ values before and after rotation of the coordinate system. The first index here runs from 1:1 for temperature only, and 1:3 for polarisation. In the latter case, 1=T, 2=E, 3=B.
psi	DP	IN	first rotation: angle $\psi$ about the z-axis. All angles are in radians and should lie in $[-2\pi, 2\pi]$ , the rotations are active and the referential system is assumed to be right handed, the routine <a href="#">coordsys2euler_zyz</a> can be used to generate the Euler angles $\psi, \theta, \varphi$ for rotation between standard astronomical coordinate systems;
theta	DP	IN	second rotation: angle $\theta$ about the original (unrotated) y-axis;
phi	DP	IN	third rotation: angle $\varphi$ about the original (unrotated) z-axis;

---

**EXAMPLE:**

```

use alm_tools, only: rotate_alm
...
call rotate_alm(64, alm_TGC, PI/3., 0.5_dp, 0.0_dp)

```

Transforms scalar and tensor  $a_{\ell m}$  for  $\ell_{\max} = m_{\max} = 64$  to emulate a rotation of the underlying map by  $(\psi = \pi/3, \theta = 0.5, \varphi = 0)$ .

---

**EXAMPLE:**

```

use coord_v_convert, only: coordsys2euler_zyz
use alm_tools, only: rotate_alm
...
call coordsys2euler_zyz(2000.0_dp, 2000.0_dp, 'E', 'G', psi, theta, phi)
call rotate_alm(64, alm_TGC, psi, theta, phi)

```

Rotate the  $a_{\ell m}$  from Ecliptic to Galactic coordinates.

---

**RELATED ROUTINES**

This section lists the routines related to **rotate\_alm\***.

<b>coordsys2euler_zyz</b>	can be used to generate the Euler angles $\psi, \theta, \varphi$ for rotation between standard astronomical coordinate systems
<b>create_alm</b>	Routine to create $a_{\ell m}$ coefficients.
<b>alter_alm</b>	Routine to modify $a_{\ell m}$ coefficients to apply or remove the effect of an instrumental beam.
<b>map2alm</b>	Routines to analyze a <b>HEALPix</b> sky map into its $a_{\ell m}$ coefficients.
<b>alm2map</b>	Routines to synthesize a <b>HEALPix</b> sky map from its $a_{\ell m}$ coefficients.
<b>alms2fits, dump_alms</b>	Routines to save a set of $a_{\ell m}$ in a FITS file.
<b>xcc_v_convert</b>	rotates a 3D coordinate vector from one astronomical coordinate system to another.



## same\_shape\_pixels\_nest, same\_shape\_pixels\_ring

---

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

These routines provide the ordered list of all **HEALPix** pixels having the same shape as a given template, for a resolution parameter  $N_{\text{side}}$ . Depending on the template considered the number of such pixels is either 8, 16,  $4N_{\text{side}}$  or  $8N_{\text{side}}$ .

The template pixels are all located in the Northern Hemisphere, or on the Equator. They are chosen to have their center located at

$$\begin{aligned} z = \cos(\theta) &\geq 2/3, & 0 < \phi &\leq \pi/2, \\ 2/3 > z &\geq 0, & \phi &= 0, \quad \text{or} \quad \phi = \frac{\pi}{4N_{\text{side}}}. \end{aligned} \quad (23)$$

They are numbered continuously from 0, starting at the North Pole, with the index increasing in  $\phi$ , and then increasing for decreasing  $z$ .

---

**FORMAT**                      call same\_shape\_pixels\_nest([nside](#), [template](#)  
                                  [, [list](#), [reflexion](#), [nrep](#)])

---

**FORMAT**                      call same\_shape\_pixels\_ring([nside](#), [template](#)  
                                  [, [list](#), [reflexion](#), [nrep](#)])

---

**ARGUMENTS**



name & dimensionality	kind	in/out	description
nside	I4B	IN	the <b>HEALPix</b> $N_{\text{side}}$ parameter.
template	I4B/ I8B	IN	identification number of the template pixel (the numbering scheme of the pixel templates is the same for both routines).
list(0:nrep-1) OPTIONAL	I4B/ I8B	OUT	pointer containing the ordered list of NESTED/RING scheme identification numbers (in $\{0, 12N_{\text{side}}^2 - 1\}$ ) of all pixels having the same shape as the template provided. The routines will allocate the <b>list</b> array if it is not allocated upon calling.
reflexion(0:nrep-1) OPTIONAL	I4B	OUT	pointer containing the transformation(s) (in $\{0, 3\}$ ) to apply to each of the returned pixels to match exactly in shape and position its respective template. 0: rotation around the polar axis only, 1: rotation + East-West swap (ie, reflexion around meridian), 2: rotation + North-South swap (ie, reflexion around Equator), 3: rotation + East-West and North-South swaps. The routines will allocate the <b>list</b> array if it is not allocated upon calling.
nrep OPTIONAL	I4B/ I8B	OUT	number of pixels having the same template (either 8, 16, $4N_{\text{side}}$ or $8N_{\text{side}}$ ).

---

### EXAMPLE:

call same\_shape\_pixels\_ring(256, 1234, list, reflexion, np)

Returns in **list** the RING-scheme index of the all the pixels having the same shape as the template #1234 for  $N_{\text{side}} = 256$ . Upon return **reflexion** will contain the rotation/reflexions to apply to each pixel returned to match the template, and **np** will contain the number of pixels having that same shape (16 in that case).

---

### RELATED ROUTINES

This section lists the routines related to **same\_shape\_pixels\_ring**.

<code>nside2templates</code>	returns the number of template pixel shapes available for a given $N_{\text{side}}$ .
<code>template_pixel_ring</code>	return the template shape matching the pixel provided
<code>template_pixel_nest</code>	

---

## scan\_directories

Location in HEALPix directory tree: [src/f90/mod/paramfile\\_io.F90](#)

Function to scan a set of directories for a given file

---

**FORMAT**                    var=scan\_directories([directories](#),        [filename](#),  
                              [fullpath](#))

---

### ARGUMENTS

name&dimensionality	kind	in/out	description
directories	CHR	IN	contains the set of directories (up to 20), separated by an ASCII character of value < 32 (see <a href="#">concatnl</a> ). During the search, it is assumed that the given directories and filename can be separated by nothing, a / (slash) or a \ (backslash)
filename	CHR	IN	the file to be found.
fullpath	CHR	OUT	returns the full path to the first occurrence of the file among the directories provided. Empty if the file is not found. The search is not recursive.
var	LGT	OUT	set to true if the file is found, to false otherwise.

---

### EXAMPLE:

```
use paramfile_io
character(len=filenamelen) :: dirs, full
logical(lgt) :: found
dirs = concatnl('dir1','/dir2','/dir2/subdir1/') ! build directories
list.
found = scan_directories(dirs, 'myfile', full) ! do the search
if (found) print*,trim(full)
```

Search for 'myfile' in the directories 'dir1', '/dir2',  
'/dir2/subdir1/'

---

## RELATED ROUTINES

This section lists the routines related to **scan\_directories**.

<code>parse_XXX</code>	parse an ASCII file for parameters definition
<code>concatnl</code>	concatenates a set of substrings into one string, interspaced with LineFeed character

---

## size\_holes\_nest

Location in HEALPix directory tree: [src/f90/mod/mask\\_tools.F90](#)

For a input binary mask in NESTED ordering, size\_holes\_nest identifies the pixels located on the inner boundary of *invalid* regions

---

**FORMAT**                      call size\_holes\_nest(*nside*, *mask*, *nholes*, *nph*,  
                                  [*tags*, *sizeholes*, *listpix*])

Arguments appearing in *italic* are optional.

---

## ARGUMENTS

name & dim.	kind	in/out	description
<i>nside</i>	I4B	IN	The <i>nside</i> value of the input mask.
<i>mask</i> (0:Npix-1)	I4B	IN	Input binary NESTED-ordered mask. Npix = 12* <i>nside</i> * <i>nside</i>
<i>nholes</i>	I4B	OUT	Number of holes found
<i>nph</i>	I4B	OUT	Number of pixels in holes
<i>tags</i> (0:Npix-1) (OPTIONAL)	I4B	OUT	Pointer allocated by size_holes_nest, containing a sky map in which <i>invalid</i> pixels belonging to the largest hole have value -1, those belonging to the second largest hole have value -2, and so on, while valid pixels keep value +1.
<i>sizeholes</i> (0:nholes-1)	I4B	OUT	Pointer allocated by size_holes_nest, containing on output the respective size of each hole (in decreasing order). Eg, <i>sizeholes</i> (0) is the number of pixels in the largest hole (taking value -1 in <i>tags</i> ).
<i>listpix</i> (0:nholes+nph)	I4B	OUT	Pointer allocated by size_holes_nest, containing on output the indexed list of pixels in each hole. Pixels located in the first (and largest) hole are given by <i>listpix</i> ( <i>listpix</i> (0): <i>listpix</i> (1)-1)

---

**EXAMPLE:**

```

use healpix_types
use healpix_modules
...
call size_holes_nest(nside, mask, nholes, nph)

      ???

```

---

**MODULES & ROUTINES**

This section lists the modules and routines used by **size\_holes\_nest**.

<b>mask_tools</b>	mask processing module (see related routines below)
-------------------	---

---

**RELATED ROUTINES**

This section lists the routines related to **size\_holes\_nest**.

<b>dist2holes_nest</b>	angular distance to closest invalid pixel of the given mask
<b>fill_holes_nest</b>	turn to <i>valid</i> all pixels located in 'holes' containing fewer pixels than the given threshold
<b>maskborder_nest</b>	identify inner boundary pixels of 'holes' for given mask
<b>size_holes_nest</b>	returns size (in pixels) of holes found in input mask

---

# string, strlowercase, struppercase

Location in HEALPix directory tree: [src/f90/mod/misc\\_utils.F90](#)

The Fortran90 module `misc_utils` contains three functions to create or manipulate character strings.

## ARGUMENTS

name & dimensionality	kind	in/out	description
number	LGT/ I4B/ SP/ DP	IN	number or boolean flag to be turned into a character string.
instring	CHR	IN	arbitrary character string.
outstring	CHR	—	output character string.
format OPTIONAL	CHR	IN	character string describing Fortran format of output.

## FUNCTIONS:

```
outstring = string(number [,format])
```

returns in `outstring` its argument `number` converted to a character string. If `format` is provided it is used to format the output, if not, the fortran default format matching `number`'s type is used.

```
outstring = strlowercase(instring)
```

returns in `outstring` its argument `instring` converted to lowercase. ASCII characters in the [A-Z] range are mapped to [a-z], while all others remain unchanged.

```
outstring = struppercase(instring)
```

returns in `outstring` its argument `instring` converted to uppercase. ASCII characters in the `[a-z]` range are mapped to `[A-Z]`, while all others remain unchanged.

---

**EXAMPLE:**

```
use misc_utils
character(len=24) :: s1
s1 = string(123,'(i5.5)')
print*, trim(s1)
print*,trim(strupcase('*aBcD-123'))
print*,trim(strlowcase('*aBcD-123'))
```

Will printout 00123, \*ABCD-123 and \*abcd-123.

---



# surface\_triangle

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Returns the surface in steradians of the spherical triangle described by its three vertices

---

**FORMAT**                      call surface\_triangle([v1](#), [v2](#), [v3](#), [surface](#))

---

## ARGUMENTS

name&dimensionality	kind	in/out	description
v1(3)	DP	IN	cartesian vector pointing at the triangle first vertex.
v2(3)	DP	IN	cartesian vector pointing at the triangle second vertex.
v3(3)	DP	IN	cartesian vector pointing at the triangle third vertex.
surface	DP	OUT	surface of the triangle in steradians.

---

## EXAMPLE:

```
use healpix_types
use pix_tools, only : surface_triangle
real(DP) :: surface, one = 1.0_dp
call surface_triangle((/1,0,0/)*one, (/0,1,0/)*one, (/0,0,1/)*one,
surface)
print*, surface
```

Returns the surface in steradians of the triangle defined by the octant ( $x, y, z > 0$ ) : 1.5707963267948966

---

## RELATED ROUTINES

This section lists the routines related to **surface\_triangle**.

<code>pix2ang</code> , <code>ang2pix</code>	convert between angle and pixel number.
<code>pix2vec</code> , <code>vec2pix</code>	convert between a cartesian vector and pixel number.
<code>query_disc</code> , <code>query_polygon</code> , <code>query_strip</code> , <code>query_triangle</code>	render the list of pixels enclosed respectively in a given disc, polygon, latitude strip and triangle

---

# template\_pixel\_nest, template\_pixel\_ring

---

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Routines to provide the index of the template pixel associated with a given **HEALPix** pixel, for a resolution parameter  $N_{\text{side}}$ .

Any pixel can be *matched in shape* to a single of these templates by a combination of a rotation around the polar axis with reflexion(s) around a meridian and/or the equator.

The template pixels are all located in the Northern Hemisphere, or on the Equator. They are chosen to have their center located at

$$\begin{aligned} z = \cos(\theta) &\geq 2/3, & 0 < \phi &\leq \pi/2, \\ 2/3 > z &\geq 0, & \phi &= 0, \quad \text{or} \quad \phi = \frac{\pi}{4N_{\text{side}}}. \end{aligned} \quad (24)$$

They are numbered continuously from 0, starting at the North Pole, with the index increasing in  $\phi$ , and then increasing for decreasing  $z$ .

---

<b>FORMAT</b>	call template_pixel_nest( <a href="#">nside</a> , <a href="#">pixel_nest</a> , <a href="#">template</a> , <a href="#">reflexion</a> )
---------------	--

---

<b>FORMAT</b>	call template_pixel_ring( <a href="#">nside</a> , <a href="#">pixel_ring</a> , <a href="#">template</a> , <a href="#">reflexion</a> )
---------------	--

---

## ARGUMENTS

---

name & dimensionality	kind	in/out	description
nside	I4B	IN	the <b>HEALPix</b> $N_{\text{side}}$ parameter.
pixel_nest	I4B/ I8B	IN	NESTED scheme pixel identification number over the range $\{0, 12N_{\text{side}}^2 - 1\}$ .
pixel_ring	I4B/ I8B	IN	RING scheme pixel identification number over the range $\{0, 12N_{\text{side}}^2 - 1\}$ .
template	I4B/ I8B	OUT	identification number of the template matching in shape the pixel provided (the numbering scheme of the pixel templates is the same for both routines).
reflexion	I4B	OUT	in $\{0, 3\}$ encodes the transformation(s) to apply to each pixel provided to match exactly in shape and position its respective template. 0: rotation around the polar axis only, 1: rotation + East-West swap (ie, reflexion around meridian), 2: rotation + North-South swap (ie, reflexion around Equator), 3: rotation + East-West and North-South swaps

---

### EXAMPLE:

```
call template_pixel_ring(256, 500000, template, reflexion)
```

Returns in **template** the index of the template pixel (16663) whose shape matches that of the pixel #500000 for  $N_{\text{side}} = 256$ . Upon return **reflexion** will contain 2, meaning that the template must be reflected around a meridian and around the equator (and then rotated around the polar axis) in order to match the pixel.

---

### RELATED ROUTINES

This section lists the routines related to **template\_pixel\_ring**.

<b>nside2templates</b>	returns the number of template pixel shapes available for a given $N_{\text{side}}$ .
<b>same_shape_pixels_ring</b>	
<b>same_shape_pixels_nest</b>	return the ordered list of pixels having the same shape as a given pixel template



## udgrade\_\_nest\*

---

Location in HEALPix directory tree: [src/f90/mod/udgrade\\_nr.f90](#)

Routine to degrade or prograde the pixel size of a **HEALPix** map indexed with the NESTED scheme. The degradation/progradation is done assuming an intensive quantity (like temperature) that does NOT scale with surface area.

In case of degradation, a big pixel that contains one or several bad pixels will take the average of the valid small pixels, unless a 'pessimistic' behavior is assumed in which case the big pixel will take the bad pixel sentinel value. In case of progradation, a bad pixel only spawns bad pixels.

The routine accepts both mono and bi-dimensional maps.

---

<b>FORMAT</b>	call    udgrade__nest*( <a href="#">map_in</a> , <a href="#">nside_in</a> , <a href="#">map_out</a> , <a href="#">nside_out</a> [, <i>fmissval</i> , <i>pessimistic</i> ])
---------------	---

Arguments appearing in *italic* are optional.

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
map_in(0:12*nside_in**2-1)	SP/ DP	IN	mono-dimensional full sky map to be prograded or degraded.
map_in (0:12*nside_in**2-1,1:nd)	SP/ DP	IN	bi-dimensional full sky map to be prograded or degraded. The routine finds the second dimension (nd) by itself.
nside_in	I4B	IN	the $N_{\text{side}}$ resolution parameter of the input map. Must be a power of 2.
map_out(0:12*nside_out**2-1)	SP/ DP	OUT	mono-dimensional full sky map after degradation or progradation.
map_out (0:12*nside_out**2-1,1:nd)	SP/ DP	OUT	bi-dimensional full sky map after degradation or progradation. The second dimension (nd) should match that of the input map.
nside_out	I4B	IN	the $N_{\text{side}}$ resolution parameter of the output map. Must be a power of 2. If <code>nside_out &gt; nside_in</code> , the map is prograded (ie, more and smaller pixels) with each pixel having the same value as its parent; otherwise, the map is degraded (ie, fewer larger pixels), with each pixel being the average of its $(\text{nside\_in}/\text{nside\_out})^2$ components.
<i>fmissval</i>	SP/ DP	IN	sentinel value given to bad pixels in input and output maps. ( <b>default:</b> <span style="color: red;">HPX_SBADVAL</span> or <span style="color: red;">HPX_DBADVAL</span> )
<i>pessimistic</i>	LGT	IN	if set to <code>.true.</code> , during a degradation, a big pixel containing at least a small bad pixel will be returned as bad as well, instead of taking the average of the remaining valid pixels. ( <b>default:</b> <code>.false.</code> )

---

### EXAMPLE:

```
use udgrade_nr
call udgrade_nest(map_hi, 256, map_low, 64)
```

Degrades a NESTED ordered map with  $N_{\text{side}} = 256$  into a NESTED map with  $N_{\text{side}} = 64$

---

## RELATED ROUTINES

This section lists the routines related to **udgrade\_nest\***.

**udgrade\_ring**            prograde or degrade a RING ordered map.

---



## udgrade\_ring\*

---

Location in HEALPix directory tree: [src/f90/mod/udgrade\\_nr.f90](#)

Routine to degrade or prograde the pixel size of a **HEALPix** map indexed with the RING scheme. The degradation/progradation is done assuming an intensive quantity (like temperature) that does NOT scale with surface area.

In case of degradation, a big pixel that contains one or several bad pixels will take the average of the valid small pixels, unless a 'pessimistic' behavior is assumed in which case the big pixel will take the bad pixel sentinel value. In case of progradation, a bad pixel only spawns bad pixels.

The routine accepts both mono and bi-dimensional maps.

---

<b>FORMAT</b>	call	udgrade_ring*(	map_in,	nside_in,
			map_out,	nside_out[, <i>fmissval</i> , <i>pessimistic</i> ])

Arguments appearing in *italic* are optional.

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
map_in(0:12*nside_in**2-1)	SP/ DP	INOUT	mono-dimensional full sky map to be prograded or degraded. The routine finds the second dimension (nd) by itself. <b>Note that the map is modified on output (reordered into NESTED scheme).</b>
map_in (0:12*nside_in**2-1,1:nd)	SP/ DP	INOUT	bi-dimensional full sky map to be prograded or degraded. <b>Note that the map is modified on output (reordered into NESTED scheme).</b>
nside_in	I4B	IN	the $N_{\text{side}}$ resolution parameter of the input map. Must be a power of 2.
map_out(0:12*nside_out**2-1)	SP/ DP	OUT	mono-dimensional full sky map after degradation or progradation.
map_out (0:12*nside_out**2-1,1:nd)	SP/ DP	OUT	bi-dimensional full sky map after degradation or progradation. The second dimension (nd) should match that of the input map.
nside_out	I4B	IN	the $N_{\text{side}}$ resolution parameter of the output map. Must be a power of 2. If nside_out > nside_in, the map is prograded (ie, more and smaller pixels) with each pixel having the same value as its parent; otherwise, the map is degraded (ie, fewer larger pixels), with each pixel being the average of its (nside_in/nside_out) <sup>2</sup> components.
fmissval	SP/ DP	IN	sentinel value given to bad pixels in input and output maps. (default: <b>HPX_SBADVAL</b> or <b>HPX_DBADVAL</b> )
pessimistic	LGT	IN	if set to <b>.true.</b> , during a degradation, a big pixel containing at least a small bad pixel will be returned as bad as well, instead of taking the average of the remaining valid pixels. (default: <b>.false.</b> )

---

### EXAMPLE:

```
use udgrade_nr
```

```
call udgrade_ring(map_hi, 256, map_low, 64)
```

Degrades a RING ordered map with  $N_{\text{side}} = 256$  into a RING map with  $N_{\text{side}} = 64$

---

## RELATED ROUTINES

This section lists the routines related to **udgrade\_ring\***.

**udgrade\_nest**      prograde or degrade a NESTED ordered map.

---

# unfold\_\_weightsfile

Location in HEALPix directory tree: **src/f90/mod/fitstools.F90**

This routine read a ring-based or pixel-based quadrature weight file and turn it into a full sky (ring-ordered) **HEALPix** map.

---

**FORMAT**                      call unfold\_\_weightsfile(**w8file**, **w8map**)

---

## ARGUMENTS

---

name & dimensionality	kind	in/out	description
w8file(LEN=filenamelen)	CHR	IN	filename of FITS-file containing a list of ring-ordered or pixel-ordered quadrature weights for some Nside.
w8map(0:12*Nside**2-1)	SP/ DP	OUT	an array containing a full sky map of weights for the same Nside.

---

## EXAMPLE:

---

```

use healpix_modules
real(DP), allocatable(:) :: w8map
character(len=FILENAMELEN) :: dirname, filename
integer(I4B) :: nside, won
won = 2
nside = 128
npix = nside2npix(nside)
allocate(w8map(0:npix-1))
dirname = get_healpix_data_dir()
filename = get_healpix_weights_file(nside, won)
filename = trim(dirname)////'/trim(filename)
call unfold_weightsfile(filename, w8map)

```

This code snippet looks for a pixel-based (**won=2**) weights file in standard location, with a standard name, for **nside=128**, reads it and unfolds it into a full sky ring-ordered map named **w8map**.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **unfold\_weightsfile**.

<b>pixtools</b>	module, containing:
<b>nside2npweights</b>	function returning the number of pixel-based weights expected for a given Nside when stored in compact form
<b>nside2npix</b>	function returning the number of pixels on the full sky, for a given Nside
<b>fitstools</b>	module, containing:
<b>printerror</b>	routine for printing FITS error messages.
<b>getsize_fits</b>	function returning the size of a FITS file, and parsing its header.
<b>input_map</b>	routine to read FITS file.
<b>cfitsio</b>	library for FITS file handling.

---

## RELATED ROUTINES

This section lists the routines related to **unfold\_weightsfile**.

<b>get_healpix_data_dir</b>	routine returning actual path to <b>data</b> directory
<b>get_healpix_weights_file</b>	routine returning the standard name of pre-computed weights file
<b>anafast</b> , <b>smoothing</b>	these two facilities use <b>unfold_weightsfile</b> to apply quadrature weighting to the maps they respectively analyze and smooth

---

## uniq2nest

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

This F90 facility turns the Unique Identifier  $u = p + 4N_{\text{side}}^2$ , into the parameter  $N_{\text{side}}$  (a power of 2) and the pixel index  $p$ . See ["The Unique Identifier scheme"](#) section in ["HEALPix Introduction Document"](#) for more details.

---

**FORMAT**                      call uniq2nest([puniq](#), [nside](#), [pnest](#))

---

### ARGUMENTS

---

name	kind	in/out	description
puniq	I4B/I8B	IN	The <b>HEALPix</b> Unique pixel identifier. Must be $\geq 4$ .
nside	I4B	OUT	The <b>HEALPix</b> $N_{\text{side}}$ parameter.
pnest	I4B/I8B	OUT	(NESTED scheme) pixel identification number over the range $\{0, 12N_{\text{side}}^2 - 1\}$ .

---

### EXAMPLE:

```
use healpix_modules
integer(I4B) :: nside, pnest
call uniq2nest(4, nside, pnest)
print*,nside,pnest
```

returns

1        0

since the pixel with Unique ID number 4 is the first pixel ( $p = 0$ ) at  $N_{\text{side}} = 1$ .

---

### RELATED ROUTINES

This section lists the routines related to **uniq2nest**.

[nest2uniq](#)

Transforms Nside and Nested pixel number into Unique **HEALPix** pixel ID number

`pix2xxx, ...` to turn NESTED pixel index into sky coordinates  
and back

---

## vec2ang

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Routine to convert the 3D position vector  $(x, y, z)$  of point into its position angles  $(\theta, \phi)$  on the sphere with  $x = \sin \theta \cos \phi$ ,  $y = \sin \theta \sin \phi$ ,  $z = \cos \theta$ .

---

**FORMAT**                      call vec2ang([vector](#), [theta](#), [phi](#))

---

### ARGUMENTS

name&dimensionality	kind	in/out	description
vector(3)	DP	IN	three dimensional cartesian position vector $(x, y, z)$ . The north pole is $(0, 0, 1)$
theta	DP	OUT	colatitude in radians measured southward from north pole (in $[0, \pi]$ ).
phi	DP	OUT	longitude in radians measured eastward (in $[0, 2\pi]$ ).

---

### RELATED ROUTINES

This section lists the routines related to **vec2ang**.

<a href="#">ang2vec</a>	converts the position angles of a point on the sphere into its 3D position vector.
<a href="#">angdist</a>	computes the angular distance between 2 vectors
<a href="#">vect_prod</a>	computes the vector product between two 3D vectors

---



# vect\_prod

Location in HEALPix directory tree: [src/f90/mod/pix\\_tools.F90](#)

Returns the vectorial product of two vectors.

---

**FORMAT**                      call vect\_prod([v1](#), [v2](#), [v3](#))

---

## ARGUMENTS

---

name & dimensionality	kind	in/out	description
v1(3)	DP	IN	cartesian vector $\mathbf{v}_1$ .
v2(3)	DP	IN	cartesian vector $\mathbf{v}_2$ .
v3(3)	DP	OUT	cartesian vector $\mathbf{v}_3 = \mathbf{v}_1 \times \mathbf{v}_2$

## EXAMPLE:

---

```
use healpix_types
use pix_tools, only : vect_prod
real(DP), dimension(3) :: vec
real(DP) :: one = 1.0_dp
call vect_prod((/2,0,0/)*one, (/0,1,0/)*one, vec)
print*, vec
```

will return : 0.00E+000 0.00E+000 2.00

---

## RELATED ROUTINES

This section lists the routines related to **vect\_prod**.

<a href="#">ang2vec</a>	converts the position angles of a point on the sphere into its 3D position vector.
<a href="#">angdist</a>	computes the angular distance between 2 vectors

**vec2ang**

converts the 3D position vector of point into its position angles on the sphere.

---

# write\_asctab\*

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine stores a power spectrum in an ascii FITS-file. The routine can store temperature coefficients  $C_\ell^T$  or both temperature and polarisation coefficients  $C_\ell^T, C_\ell^E, C_\ell^B, C_\ell^{T \times E}$ .

**FORMAT**                      call write\_asctab\*([clout](#), [lmax](#), [ncl](#), [header](#), [nlheader](#), [filename](#)[, [extno](#)])

Arguments appearing in *italic* are optional.

## ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=filenamelen)	CHR	IN	the FITS file to which the power spectrum is written.
lmax	I4B	IN	Maximum $\ell$ value to be written.
ncl	I4B	IN	1 for temperature coefficients only, 4 for polarisation.
clout(0:lmax,1:ncl)	SP/ DP	IN	the powerspectrum to be saved in the file.
nlheader	I4B	IN	number of header lines to write to the file.
header(LEN=80) (1:nlheader)	CHR	IN	the header to the FITS-file.
<i>extno</i>	I4B	IN	extension number in which to write the data (0 based). ( <b>default: 0</b> )

## EXAMPLE:

```
use healpix\_modules
real(SP), allocatable, dimension(:, :) :: cl
character(len=80), dimension(1:100) :: header
allocate(cl(0:64,1:1))
call write\_minimal\_header(header, 'cl', nlmax=64)
```

```
call write_asctab (cl,64,1,header,100,'cl.fits')
```

Writes a power spectrum in the array `cl(0:64,1:1)` to a FITS-file called 'cl.fits'. The `cl` array contains the temperature power spectrum  $C_\ell^T$  up to an  $\ell$  value of 64. 100 header lines are written to the file from the array `header(1:100)` which was previously filled the minimal required information for a power spectrum file.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **write\_asctab\***.

<b>fitstools</b>	module, containing:
<b>printerror</b>	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.

---

## RELATED ROUTINES

This section lists the routines related to **write\_asctab\***.

<b>alm2cl</b>	Routine computing the power spectrum from spherical harmonics coefficients $a_{\ell m}$
<b>fits2cl</b>	Routine to read a FITS file created by <code>write_asctab</code> .
<b>write_minimal_header</b>	routine to write minimal FITS header

---

## write\_bintab\*

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine creates a binary FITS-file from a **HEALPix** map. The routine can save a temperature map or both temperature and polarisation maps (T,Q,U) to the file.

---

**FORMAT**                      call write\_bintab\*([map](#), [npix](#), [nmap](#), [header](#),  
                                 [nlheader](#), [filename](#)[, [extno](#)])

Arguments appearing in *italic* are optional.

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
map(0:npix-1,1:nmap)	SP/ DP	IN	the map to write to the FITS-file.
npix	I4B/ I8B	IN	Number of pixels in the map.
nmap	I4B	IN	number of maps to be written, 1 for temperature only, and 3 for (T,Q,U).
header(LEN=80)      (1:nl- header)	CHR	IN	The header for the FITS-file.
nlheader	I4B	IN	number of header lines to write to the file.
filename(LEN=*)	CHR	IN	the map(s) is (are) written to a FITS-file with this filename.
<i>extno</i>	I4B	IN	extension number in which to write the data (0 based). ( <b>default:</b> 0)

---

## EXAMPLE:

call write\_bintab (map,12\*32\*\*2,3,header,120,'map.fits')

Makes a binary FITS-file called ‘map.fits’ from the **HEALPix** maps (T,Q,U) in the array `map(0:12*32**2-1,1:3)`. The number of pixels  $12*32**2$  corresponds to the number of pixels in a  $N_{\text{side}} = 32$  **HEALPix** map. The header for the FITS-file is given in the string array header and the number of lines in the header is 120.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **write\_bintab\***.

<b>fitstools</b>	module, containing:
<b>prnterror</b>	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.

---

## RELATED ROUTINES

This section lists the routines related to **write\_bintab\***.

<b>input_map, read_bintab</b>	routines which read a file created by <b>write_bintab*</b> .
<b>map2alm</b>	subroutine which analyse a map and returns the $a_{\ell m}$ coefficients.
<b>output_map</b>	subroutine which calls <b>write_bintab*</b>
<b>write_bintabh</b>	subroutine to write a large array into a FITS file piece by piece
<b>input_tod*</b>	subroutine to read an arbitrary subsection of a large binary table
<b>write_minimal_header</b>	routine to write minimal FITS header

---

## write\_bintabh

---

Location in HEALPix directory tree: **src/f90/mod/fitstools.F90**

This routine is designed to write large (or huge) arrays into a binary table extension of a FITS file. The user can choose to write the array piece by piece. This is designed to deal with Time Ordered Data set (tod).

---

**FORMAT**                    call write\_bintabh(**tod**, **npix**, **ntod**, **header**, **nl-**  
                             **header**, **filename**, [*extno*, *firstpix*, *repeat*])

Arguments appearing in *italic* are optional.

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
tod(0:npix-1,1:ntod)	SP/ DP	IN	The map or tod to write to the FITS file. It will be written in the file at the location corresponding to pixels (or time samples) <b>firstpix</b> to <b>firtpix + npix -1</b> .
npix	I8B	IN	Number of pixels or time samples in the map or TOD. See Note below.
ntod	I4B	IN	Number of maps or tods to be written. Each of them will be in a different column of the FITS binary table.
header(LEN=80) (1:nlheader)	CHR	IN	The header for the FITS file.
nlheader	I4B	IN	number of header lines to write to the file.
filename(LEN=filename <sub>len</sub> )	CHR	IN	The array is written into a FITS file with this filename.
extno	I4B	IN	extension number in which to write the data (0 based). ( <b>default:</b> 0)
firstpix	I8B	IN	0 Location in the FITS file of the first pixel (or time sample) to be written (0 based). ( <b>default:</b> 0). See Note below.
repeat	I4B	IN	Length of the element vector used in the binary table. ( <b>default:</b> 1024 if <b>npix</b> $\propto$ 1024; 12000 if <b>npix</b> > 12000 and 1 otherwise). Choosing a large <b>repeat</b> for multi-column tables ( <b>ntod</b> > 1) generally speeds up the I/O. It also helps bringing the number of rows of the table under $2^{31}$ , which is a hard limit of cfitsio. If the number of samples or pixels of each map or TOD is not a multiple of <b>repeat</b> , then the last element vector will be padded with sentinel values <b>HPX_SBADVAL</b> or <b>HPX_DBADVAL</b> .

**Note :** Indices and number of data elements larger than  $2^{31}$  are only accessible in FITS files on computers with 64 bit enabled compilers and with some specific compilation options of cfitsio (see cfitsio documentation).

---

## EXAMPLE:



```

use healpix_types
use fitstools, only : write_bintabh
character(len=80), dimension(1:128) :: hdr
real(SP), dimension(0:49,1) :: tod
character(len=FILENAMELEN) :: fname='tod.fits'
hdr(:) = ' '
tod(:,1) = 1.
call write_bintabh(tod, 50_i8b, 1, hdr, 128, fname, firstpix=0_i8b,
repeat=10)
tod = tod * 3.
call write_bintabh(tod, 20_i8b, 1, hdr, 128, fname, firstpix=40_i8b)

```

Writes into the FITS file 'tod.fits' a 1 column binary table, where the first 40 data samples have the value 1. and the next 20 have the value 3. (Note that in this example the second call to write\_bintabh overwrites some of the pixels written by the first call). The samples will be written in element vectors of length 10. The header for the FITS file is given in the string array `hdr` and its number of lines is 128.

---

## MODULES & ROUTINES

This section lists the modules and routines used by `write_bintabh`.

<code>fitstools</code>	module, containing:
<code>prnterror</code>	routine for printing FITS error messages.
<code>cfitsio</code>	library for FITS file handling.

---

## RELATED ROUTINES

This section lists the routines related to `write_bintabh`.

<code>input_tod*</code>	routine that reads a file created by write_bintabh.
<code>input_map</code> , <code>read_bintab</code>	routines to read <b>HEALPix</b> sky map,
<code>write_minimal_header</code>	routine to write minimal FITS header

---

## write\_\_dbintab

---

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine is obsolete.

To write  $P_{lm}$  polynoms into a FITS file, use [write\\_\\_plm](#) instead.

To write a Healpix map into a FITS file, use [write\\_\\_bintab](#) or [output\\_\\_map](#).

---

---

## write\_fits\_cut4

---

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine writes a cut sky **HEALPix** map into a FITS file. The format used for the FITS file follows the one used for Boomerang98 and is adapted from COBE/DMR. This routine can be used to store polarized maps, where the information relative to the Stokes parameters I, Q and U are placed in extension 0, 1 and 2 respectively by successive invocation of the routine.

---

<b>FORMAT</b>	call write_fits_cut4( <a href="#">filename</a> , <a href="#">np</a> , <a href="#">pixel</a> , <a href="#">signal</a> , <a href="#">n_obs</a> , <a href="#">serror</a> [, <a href="#">header</a> , <a href="#">coord</a> , <a href="#">nside</a> , <a href="#">order</a> , <a href="#">units</a> , <a href="#">extno</a> , <a href="#">polarisation</a> ])
---------------	---

Arguments appearing in *italic* are optional.

---

## ARGUMENTS

name&dimensionality	kind	in/out	description
filename(LEN=filenamelen)	CHR	IN	FITS file to be read from, containing a cut sky map
np	I4B	IN	number of pixels to be written in the file
pixel(0:np-1)	I4B	IN	index of observed (or valid) pixels
signal(0:np-1)	SP	IN	value of signal in each observed pixel
n_obs(0:np-1)	I4B	IN	number of observation per pixel
serror(0:np-1)	SP	IN	<i>rms</i> of signal in pixel, for white noise, this is $\propto 1/\sqrt{n\_obs}$ .
header(LEN=80)(1:) (OPTIONAL)	CHR	IN	FITS extension header to be included in the FITS file
coord(LEN=1)	CHR	IN	astrophysical coordinates ('C' or 'Q' Celestial/eQuatorial, 'G' for Galactic, 'E' for Ecliptic)
nside	I4B	IN	<b>HEALPix</b> resolution parameter of data set
order	I4B	IN	<b>HEALPix</b> ordering scheme, 1: RING, 2: NESTED
units(LEN=20)	CHR	IN	maps units (applies only to Signal and Serror)
extno	I4B	IN	(0 based) extension number in which to write data. ( <b>default:</b> 0). If set to 0 (or not set) <i>a new file is written from scratch</i> . If set to a value larger than 1, the corresponding extension is added or updated, as long as all previous extensions already exist. All extensions of the same file should use the same Nside, Order and Coord
polarisaton	I4B	IN	if set to a non zero value, specifies that file will contain the I, Q and U polarisation Stokes parameter in extensions 0, 1 and 2 respectively, and sets the FITS header keywords accordingly. If not set, the keywords found in <b>header</b> will prevail. Note: the information relative to Nside, Order and Coord <i>has</i> to be given, either thru these keyword or via the FITS Header.

---

## MODULES & ROUTINES

This section lists the modules and routines used by **write\_fits\_cut4**.

<b>fitstools</b>	module, containing:
prnterror	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.

---

## RELATED ROUTINES

This section lists the routines related to **write\_fits\_cut4**.

anafast	executable that reads a <b>HEALPix</b> map and analyses it.
synfast	executable that generate full sky <b>HEALPix</b> maps
<b>getsize_fits</b>	routine to know the size of a FITS file and its type (eg, full sky vs cut sky)
<b>input_map</b>	all purpose routine to input a map of any kind from a FITS file
<b>output_map</b>	subroutine to write a FITS file from a <b>HEALPix</b> map
<b>read_fits_cut4</b>	subroutine to read a <b>HEALPix</b> cut sky map from a FITS file
<b>write_minimal_header</b>	routine to write minimal FITS header

---

## write\_\_minimal\_\_header

Location in HEALPix directory tree: [src/f90/mod/head\\_fits.F90](#)

This routine writes the baseline FITS header for the most common **HEALPix** data sets: (cut sky or full sky) map,  $C(\ell)$  power spectra and  $a_{\ell m}$  coefficients.

---

**FORMAT**                      call write\_\_minimal\_\_header(*header*, *dtype*, [*append*, *nside*, *order*, *ordering*, *coordsys*, *creator*, *version*, *randseed*, *beam\_leg*, *fwhm\_degree*, *units*, *nlmax*, *polar*, *nmmax*, *bcross*, *deriv*, *asym\_cl*] )

Arguments appearing in *italic* are optional.

---

## ARGUMENTS

name & dimensionality	kind	in/out	description
header(LEN=80)	CHR	INOUT	The FITS header to fill in.
DIMENSION(:)			
dtype(LEN=*)	CHR	IN	data to be put in the FITS file, must be one of 'ALM', 'CL', 'MAP', 'CUTMAP' (case un-sensitive).

name & dimensionality	kind	in/out	description
<i>append</i>	LGT	IN	if set to TRUE, the keywords will be appended to the content of <b>header</b> instead of written from scratch
<i>nside</i>	I4B	IN	map resolution parameter; required for dtype='MAP' and dtype='CUTMAP'
<i>order</i>	I4B	IN	map ordering, either 1 (=ring) or 2 (=nested); see <b>ordering</b>
<i>ordering(LEN=*)</i>	CHR	IN	map ordering, either 'RING' or 'NESTED' (case un-sensitive); either <b>order</b> or <b>ordering</b> is required for dtype='MAP' and dtype='CUTMAP'
<i>coordsys(LEN=*)</i>	CHR	IN	map coordinate system; Valid choices are 'G' = Galactic, 'E' = Ecliptic, 'C'/'Q' = Celestial = eQuatorial
<i>creator(LEN=*)</i>	CHR	IN	name of software generating the data set
<i>version(LEN=*)</i>	CHR	IN	version of <b>creator</b> software
<i>randseed</i>	I4B	IN	random number generator seed used to generate the data
<i>beam_leg(LEN=*)</i>	CHR	IN	File containing Legendre transform of symmetric beam
<i>fwhm_degree</i>	DP	IN	FWHM in degrees of gaussian symmetric beam (FITS keyword: <b>FWHM</b> )
<i>units(LEN=*)</i>	CHR	IN	physical units of the data set (FITS keyword: <b>TUNIT*</b> )
<i>nlmax</i>	I4B	IN	maximum multipole order $l$ of the data set (FITS keyword: <b>MAX-LPOL</b> )
<i>polar</i>	LGT	IN	if set to .TRUE., the file to be written contains polarized data
<i>nmmax</i>	I4B	IN	maximum degree $m$ of data set (FITS keyword: <b>MAX-MPOL</b> )
<i>bcross</i>	LGT	IN	if set to .TRUE., the magnetic cross terms power spectra (TB and EB) are included; only applies to dtype='CL'
<i>deriv</i>	I4B	IN	order of derivatives to included in FITS file (0, 1 or 2); only applies to dtype='MAP'
<i>asym_cl</i>	LGT	IN	if set to .TRUE., the asymmetric power spectra (ET, BT and BE on top of TE, TB and EB) are included; only applies to dtype='CL'

**EXAMPLE:**

```

use healpix_types
use head_fits
character(len=80), dimension(1:60) :: header
call write_minimal_header(header, 'MAP', nside=256, ordering='Nested')
call add_card(header, 'HISTORY', 'Dummy map')

```

Writes in `header` a **HEALPix** compliant FITS header for a  $N_{\text{side}} = 256$  map with NESTED ordering. Further HISTORY information is added with `add_card`

---

**MODULES & ROUTINES**

This section lists the modules and routines used by `write_minimal_header`.

<code>write_hl</code>	more general routine for adding a keyword to a header.
<code>cfitsio</code>	library for FITS file handling.

---

**RELATED ROUTINES**

This section lists the routines related to `write_minimal_header`.

<code>add_card</code>	general purpose routine to write/edit an arbitrary keyword into a FITS file header.
<code>get_card</code>	general purpose routine to read any keywords from a header in a FITS file.
<code>del_card</code>	routine to discard a keyword from a FITS header
<code>read_par, number_of_alms</code>	routines to read specific keywords from a header in a FITS file.
<code>getsize_fits</code>	function returning the size of the data set in a fits file and reading some other useful FITS keywords
<code>merge_headers</code>	routine to merge two FITS headers

---



# write\_plm

Location in HEALPix directory tree: [src/f90/mod/fitstools.F90](#)

This routine creates a double precision binary FITS-file from a given array. The routine is used by the **HEALPix** facility `plmgen` to store precomputed  $P_{\ell m}(\theta)$ .

---

**FORMAT**                      call write\_plm([plm](#), [nplm](#), [nhar](#), [header](#), [nlheader](#), [filename](#), [nsmax](#), [nlmax](#))

---

## ARGUMENTS

name&dimensionality	kind	in/out	description
plm(0:nplm-1,1:nhar)	DP	IN	the array with the precomputed $P_{\ell m}(\theta)$ values.
nplm	I4B	IN	Number of $P_{\ell m}$ values to store.
nhar	I4B	IN	1 for scalar $P_{\ell m}$ only and 3 for tensor harmonics.
header(LEN=80) (1:nlheader)	CHR	IN	The header for the FITS-file.
nlheader	I4B	IN	number of header lines to write to the file.
filename(LEN=filenamelen)	CHR	IN	the precomputed $P_{\ell m}(\theta)$ values are written to this file.
nsmax	I4B	IN	$N_{\text{side}}$ for the precomputed $P_{\ell m}$ s.
nlmax	I4B	IN	maximum $\ell$ value for the precomputed $P_{\ell m}$ s.

---

**EXAMPLE:**

```
call write_plm (plm, 65*66*32, 1, header, 120, 'plm_32.fits', 32, 64)
```

Makes a double precision binary FITS-file called 'plm\_32.fits' from the precomputed  $P_{\ell m}(\theta)$  in the array `plm(0:65*66*32-1,1:1)`. The number 65\*66\*32 corresponds to the number of precomputed  $P_{\ell m}$ s needed for a  $N_{\text{side}} = 32$  **HEALPix** map synthesis/analysis. The header for the FITS-file is given in the string array `header` and the number of lines in the header is 120.

---

**MODULES & ROUTINES**

This section lists the modules and routines used by **write\_plm**.

<b>fitstools</b>	module, containing:
<b>prnterror</b>	routine for printing FITS error messages.
<b>cfitsio</b>	library for FITS file handling.

---

**RELATED ROUTINES**

This section lists the routines related to **write\_plm**.

<b>read_dbintab, read_bintab</b>	routines which reads a file created by <b>write_plm</b> .
<b>map2alm, alm2map</b>	routines using precomputed $P_{\ell m}(\theta)$ .

---

# xccc\_v\_convert

Location in HEALPix directory tree: [src/f90/mod/coord\\_v\\_convert.f90](#)

This routine rotates a 3D coordinate vector from one astronomical coordinate system to another.

---

**FORMAT**                    call xccc\_v\_convert(*ivector*, *iePOCH*, *oePOCH*,  
                              *isys*, *osys*, *ovector*)

---

## ARGUMENTS

name & dimension- ality	kind	in/out	description
ivector(1:3)	DP	IN	3D coordinate vector of one astronomical object, in the input coordinate system.
iePOCH	DP	IN	epoch of the input astronomical coordinate system.
oePOCH	DP	IN	epoch of the output astronomical coordinate system.
isys(len=*)	CHR	IN	input coordinate system, should be one of 'E'=Ecliptic, 'G'=Galactic, 'C'/'Q'=Celestial/eQuatorial.
osys(len=*)	CHR	IN	output coordinate system, same choice as above.
ovector(1:3)	DP	IN	3D coordinate vector of the same object, in the output coordinate system.

---

## EXAMPLE:

```
use healpix_types
use coord_v_convert, only: xccc_v_convert
real(dp) :: vecin(1:3), vecout(1:3)
vecin = (/ 0_dp, 0_dp, 1_dp /)
call xccc_v_convert(vecin, 2000.0_dp, 2000.0_dp, 'g', 'c', vecout)
```

Will produce in `vecout` the location in Celestial coordinates (2000 epoch) of the North Galactic Pole (defined in `vecin`)

---

## RELATED ROUTINES

This section lists the routines related to `xcc_v_convert`.

<code>coordsys2euler_zyz</code>	produces the Euler angles $\psi, \theta, \varphi$ in (Z,Y,Z) convention for rotation between standard astronomical coordinate systems.
<code>ang2vec, vec2ang</code>	Routine to convert spherical coordinates (co-latitude and longitude) into 3D vector coordinates and vice-versa.

---