An introduction to RPN standard files

Yves Chartier
(yves.chartier@ec.gc.ca)
Section informatique, Recherche en Prévision Numérique
Atmospheric Environment Service, Environment Canada

Last Revision: June 29th, 1995

Introduction

This document gives an overview of the RPN standard file format, which has been in use at the CID/CMC/RPN[1] complex since 1980. This file format is used to store gridded data from numerical weather prediction models, objective analyses and geophysical fields.

This document is intended as an introduction for people having to work with this data format. It is assumed that the reader has a working knowledge of FORTRAN, and has coded algorithms manipulating gridded data (1, 2, or 3 dimensional arrays).

After having read this document, the reader should be able to create and manipulate RPN standard files using FORTRAN.

Historical perspective

The RPN standard file format was created by Michel Valin, from the “Section Informatique” of RPN, in the late 1970’s. At that time, the CID/CMC/RPN super-computer was a CYBER 7600 (2 Mbytes RAM, 36 Mips), to which users where submitting batch jobs from a CYBER-171 front-end (512K memory, 1 Mips). Punched cards were the most common way to communicate with the computer, and 300 Baud was the standard communication speed for the lucky few who had access to a terminal.

The production system of numerical forecasts was already quite complex. It was composed of dozens of applications, most of which had their own, non-standard and undocumented data format. The majority of data files were without internal structure, a record number being often the only clue that one had to locate a given data item (e.g. to compute the height thickness between 1000 and 500 mb, one had to know that records #78 and #89 from file “xyz” had to be accessed). It is left to the reader to imagine the amount of work involved in modifying the horizontal, vertical or temporal resolutions of the models, which could alter the distribution of the data fields in the files (e.g. use rec. #117 instead of #78 and rec. #125 instead of #89).

The first release of the RPN standard file software appeared in 1980. The advantages of the new data format over the previous ones were numerous:

- each meteorological record stored in a standard file had several non-ambiguous identifiers (name, a pressure level, a forecast hour, a date of origin, precise geographical location, etc...).
- files were sequential or direct access, allowing fast positioning and retrieval of data.
- any application dealing with gridded data only had one single format to deal with, thereby eliminating the cost of using dozens of other non-descriptive formats.
- record manipulation was done through high-level subroutines; the underlying structure of the files could then change without requiring users to modify their applications.

A second release of the software came in 1982, offering new enhancements. The most significant was the total transportability of files, bit for bit, to almost any type of computer.

Some features of the original standard file software needed 60 or 64-bit machines (e.g. packing 8 Hollerith characters or storing a 10-digit integer in an integer variable). But in 1989, the CID/CMC/RPN complex acquired numerous UNIX workstations and servers, all of which were 32-bit machines. A third release of the software was therefore needed to accomodate the new platforms. At the same time several other enhancements were added to this release, such as a streamlined interface and an improved internal management.

General structure

From an end-user point of view, the RPN standard file package can be compared to commercial flat-file database software. The basic unit of storage is a file, containing numerous records of gridded meteorological fields. A record contains a 1, 2 or 3-D grid representing the values of a field, with various attributes that give information about the field.

[1] CID:Centre Informatique de Dorval/Dorval Computer Center, CMC: Canadian Meteorological Center, RPN: Recherche en Prévision Numérique

The RPN standard file flavors

The RPN standard files come in three distinct flavors: random access (RND - **which is not equivalent** to direct access in FORTRAN), indexable sequential (SEQ) and FORTRAN sequential (SEQ/FTN). These flavors define the way records are accessed within the file. In SEQ and SEQ/FTN files, an application that wants to access the 50th record of a file needs to read or skip the first 49 records before accessing it, whereas in a RND file, that record can be accessed directly.

An important concept that comes with these different file types is the pointer position; this has an impact on the behavior of some of the standard file functions. RND type files have a catalog of the contents of the file, which is always resident in computer memory. For these, the file pointer is an index that is attached to the catalog only. In SEQ and SEQ/FTN file type, there is no catalog of the contents of a file, and the file pointer is physically tied to the file. It points either to the beginning of file, to the current record or to the end of file.

Here are some of the differences found between RND and SEQ files.

- In RND files, all the attributes of a given record can be retrieved directly if the file pointer contains a valid position. In SEQ or SEQ/FTN type files, changing the value of the file pointer has no effect; the file pointer is restricted to the record currently selected (or to the end of the file).

- In RND files, the value of the file pointer becomes undefined after having written a record; in SEQ or SEQ/FTN files, the file pointer points to the end of the file after that operation.

- In RND files, most record searching functions start from the beginning of the catalog, and multiple searches do not require an explicit rewinding of the file. In SEQ or SEQ/FTN files, the same functions start their search from the current record, and multiple searches may require an explicit rewinding of the file.

The choice of the file type really depends on the application. A numerical model that reads a sequential stream of records at start-up and writes another stream of records at completion will get the most efficient input/output with SEQ/FTN or SEQ files. However, the vast majority of RPN standard files are of random access (RND) type, and this is the type recommended for beginners. SEQ type files are recommended over SEQ/FTN because they are transportable between heterogeneous systems.

The RPN standard file attributes

The RPN standard file attributes can be roughly divided in 5 categories, which are:

- **Field identification attributes**
    - name of variable
    - type of field (analysis, forecast, climatological field...)
    - label (model identification)
    - user defined index
- **Time attributes**
    - date of original analysis
    - forecast hour (e.g. 12 hour forecast)
    - length of time step used in model integration (0 if analysis)
    - time step number
- **Spatial attributes**
    - vertical level (pressure/height/sigma)
    - dimension along the X axis
    - dimension along the Y axis
    - dimension along the Z axis
    - type of geographical projection
    - parameters describing the geographical projection
- **Internal representation attributes**
    - data type (real, signed or unsigned integer, character, etc)
    - packing ratio (number of bits used to represent each element of the field)
- **Internal storage attributes**
    - erased field flag
    - length of record in host machine words
    - starting address of record in host machine words
    - unused number of bits in the last word
    - other parameters reserved for future use

The usage of these attributes varies with the operation (reading, writing, querying) that is done on the file. Generally, only a subset of the attributes described above is needed for a given operation.

I - Classification by category

The following is a list of the attributes found in RPN standard files, as they are used in FORTRAN programs, classified by category.

Field identification attributes

Data element	Suggested name	Data type	Range
Variable name	NOMVAR	CHARACTER*2	UPPER CASE
Type of field	TYPVAR	CHARACTER*1	UPPER CASE
Label	ETIKET	CHARACTER*8	UPPER CASE
User defined index	IP3	INTEGER	0-4095

Time attributes

Data element	Data type	Suggested name	Range
Date of orig. analysis	INTEGER	DATEO	MMDDYYHHR
Date of validity	INTEGER	DATEV	MMDDYYHHR
Forecast hour	INTEGER	IP2	0-32767
Length of time step	INTEGER	DEET	0-32767
Time step number	INTEGER	NPAS	0-32767

Spatial attributes

Data element	Data type	Suggested name	Range
Vertical level	INTEGER	IP1	0-32767
# of points along X	INTEGER	I	1-32767
# of points along Y	INTEGER	NJ	1-32767
# of points along Z	INTEGER	NK	1-4095
Type of geographical projection	CHARACTER*1	GRTYP	UPPER CASE
1st grid parameter	INTEGER	IG1	0-2047
2nd grid parameter	INTEGER	IG2	0-2047
3rd grid parameter	INTEGER	IG3	0-6553
4th grid parameter	INTEGER	IG4	0-6553

Internal representation attributes

Data element	Data type	Suggested name	Range
Data type	INTEGER	DATYP	0-5
Packing ratio/ # of bits	INTEGER	NPAK	0-32 / -1 to -48

Internal storage attributes

Data element	Data type	Suggested name
Erased field flag	INTEGER	DLTF
Key	INTEGER	KEY
Length of record in host machine words	INTEGER	LNG
Starting address of rec. in host machine words	INTEGER	SWA
Unused number of bits in the last word	INTEGER	UBC
reserved for future use	INTEGER	EXTRA1
reserved for future use	INTEGER	EXTRA2
reserved for future use	INTEGER	EXTRA3

II - Classification by usage

The record attributes discussed above can also be divided in 3 categories:

- **search attributes**
- **descriptive attributes**
- **internal attributes**

The search attributes are the ones that must be used at all times when using the RPN standard file subroutines. They can be used as selection criteria for querying sets of records, and their value must be defined when writing a record into a file. The descriptive attributes need only to be defined when writing a record into a file. Their value can be retrieved, but cannot be used to make queries. Finally, the values of the internal attributes are set by the standard file package; they can only be retrieved.

Here is the list of attributes, grouped according to this classification:

Search attributes

Data element	Suggested name
Variable name	NOMVAR
Type of field	TYPVAR
Label	ETIKET
Vertical level	IP1
Forecast hour	IP2
User defined index	IP3
Date of validity	DATEV=DATEO+ DEET * NPAS

Descriptive attributes

Data element	Suggested name
Length of time step	DEET
Time step number	NPAS
Date of original analysis	DATEO
Dimension of grid along the X-axis	NI
Dimension of grid along the Y-axis	NJ
Dimension of grid along the Z-axis	NK
Type of geographical projection	GRTYP
1st grid parameter	IG1
2nd grid parameter	IG2
3rd grid parameter	IG3
4th grid parameter	IG4
Numerical values data type	DATYP
Packing ratio	NPAK

Internal attributes

Data element	Suggested name
Erased field flag	DLTF
Key	KEY
Length of record in machine words	LNG
Starting address of record in machine words	SWA
Unused number of bits in the last word	UBC
reserved for future use	EXTRA1
reserved for future use	EXTRA2
reserved for future use	EXTRA3

Detailed description of RPN standard file attributes

Variable name (NOMVAR):

This is a 2-letter code (upper case only) representing a meteorological parameter. A partial list of existing codes will be found in appendix B, which users should follow. Here are some examples: PN (sea level pressure), GZ (height), TT (air temperature), HR (relative humidity).

Type of field (TYPVAR):

This is a 1-letter code (upper case only) representing the origin of the data. As for the 2-letter name described above, a partial list of existing codes will be found at appendix B. Some examples: A (analysis), C (climatology), P (forecast).

The official list of existing codes for NOMVAR and TYPVAR can be invoked on-line on the CID/CMC/RPN front-end computers using the "**r.dict**" command.

To get the dictionary definition of the code "GZ":

```
> r.dict -n gz
GZ    Geopotential Height                                  dam
```

To get the dictionary definition of the variable type "A":

```
> r.dict -t a
--A,        ANALYSIS
```

To get the dictionary definition of all variable codes starting with "A":

```
> r.dict -n a.
AA    Ammonium Aerosols (NH4)                              ppb
AL    Albedo                                            0 to 1
AM    Ammonia Gas (NH3)                                    ppb
AP    Planetary albedo                                  0 to 1
```

To get the dictionary definition of all existing variable codes:

```
> r.dict -n
(OUTPUT TOO LONG TO BE PRINTED HERE)
```

Label (ETIKET):

This is an 8-letter code (upper case only) allowing rapid identification of a field. The contents of this label is normally left to the discretion of the user. It can be used to identify a numerical model, or the code of an experiment. Some examples: the label for the operational regional finite element model is 'FE OPRUN' (**F**inite **E**lement **OP**erational **RUN**), the one for the spectral model is 'SEF79A21' (**S**pectral **E**lements **F**inis **79** waves **21** levels).

When coding the value of ETIKET in a FORTRAN program, always initialize the 8 characters of the label, such as

```
      ETIKET = 'AA      '
```

instead of

```
      ETIKET = 'AA'
```

because some implementations of FORTRAN may not pad the remaining characters with spaces but pad with null or random characters.

Vertical level (IP1):

This attribute represents a vertical level in pressure, sigma or height above mean sea level (amsl) coordinates.

In pressure coordinates, IP1 can take a value from 1 to 1200, indicating the pressure level in millibars. A value of 0 is used for a field defined at the surface, such as the sea level pressure, ground temperature or ice cover.

In sigma coordinates, IP1 can take a value from 2000 to 12000, from which we can extract the sigma level using the following formula:

sigma level = (IP1 - 2000) / 10000.

Therefore an IP1 of 12000 corresponds to a sigma level of 1.000 (terrain level), an IP1 of 6000 to a sigma level of 0.400.

In height (amsl) coordinates, IP1 can take a value from 12000 to 32000, from which we can extract the height using the following formula:

height level = (IP1 - 12000) * 5.

Therefore an IP1 of 12000 is located at the mean sea level, an IP1 of 12500 to a height (amsl) of 2500 meters.

Forecast hour (IP2):

This attribute normally represents the forecast hour (e.g. 12 hour forecast). Its value should normally be rounded to the nearest hour as given by the FORTRAN formula:

IP2 = ((NPAS * DEET+1800)/3600).

User defined identifier (IP3):

The contents of this attribute is left to the user. It should be set to 0 when not used.

Length of time step (DEET):

This is the length of a time step used during a model integration, in seconds.

Time step number (NPAS):

This is the time step number at which the field was written during an integration. The number of the initial time step is 0.

Date of original analysis (DATEO) and date of validity (DATEV):

Before the 1989 release of the RPN standard file library, the date of origin (DATEO) was originally encoded in FORTRAN programs in the following format: WMMDDYYHHR, where

- W day of the week (1=Sunday, 7=Saturday)
- MM month(01 to 12)
- DD day of the month (01 to 31)
- YY year (00-49 = from 2000 to 2049, 50-99 = from 1950 to 1999)
- HH GMT hour (00-23)
- R Operational Run (0 to 7)

That format is still used for printout in RPN utilities, such as “**voir**”. However, in the current release of the RPN standard file library, the ‘W’ part of the date-time stamp has been dropped, so that the actual format that should be used in FORTRAN programs is MMDDYYHHR (Example: 081192000 -> Augusth 11th, 1992, at 00Z, run 0).

The date of validity (DATEV) of a field is closely associated with the date of origin. It is normally computed using the subroutine “**incdat**”, which computes a date of validity from a date of original analysis and a time lapse defined by **deet*npas** (in hours). The following sample of code shows how to compute **datev** from **dateo**, **deet** and **npas**.

```
integer deltat, deet, npas, dateo, datev

deltat = (deet*npas+1800)/3600
```

```
call incdat(DATEV, dateo, deltat)
```

It is important to be aware of the difference between DATEO and DATEV. DATEO is used by the routines writing records into a file while DATEV is used by all routines querying records except one (FSTPRM).

Dimension of grid along the X, Y and Z axes (NI, NJ, NK):

This is the physical dimension of the grid along each spatial axis. On a geographical map, NI lies along the horizontal or X axis, NJ along the vertical or Y axis, and NK would point out of the map or along the Z axis.

Type of geographical projection (GRTYP) and grid parameters (IG1-IG2-IG3-IG4):

The usage of these parameters will be discussed extensively in appendix C, "**Conventions regarding the usage of grid descriptors in RPN standard files**".

Type of data (DATYP):

This is a numerical code indicating the data type of the numerical values stored in a record. This is the list of existing codes:

0:	raw binary (unexportable among platforms)
1:	floating point
2:	integer
3:	character
4:	signed integer
5:	IEEE style representation

Packing ratio (NPAK and NBITS):

In order to save disk space, the numerical values stored in RPN standard file records are not kept to their full precision (typically 32 bits on UNIX computers); they are usually compressed to occupy between 12 to 16 bits per floating point value. The compression factor can go as high as 1 bit per value.

At the time of creation of RPN standard files, users at CID/CMC/RPN counted memory and disk space in terms of words rather than bytes. At that time, the packing ratio was also expressed in items per word. A packing ratio of 4 (NPAK=4) meant that 4 floating point values could be stored with a precision of 16 bits into a 64-bit CRAY word. That precision becomes 15 bits on a CDC CYBER-720 60-bit word and 8 bits on a standard 32-bit word UNIX system. These differences in interpretation can be confusing, especially when the same code runs on different platforms. In order to get an absolute value for the arithmetic precision while keeping backward compatibility, the following standard has been adopted:

Let **NBITS** be the number of bits kept per floating point value.

NPAK = 0	->	No compaction, **NBITS = (number of bits/word)**
NPAK = 1	->	No compaction, **NBITS = (number of bits/word)**
NPAK > 0	->	**NBITS = (number of bits/word) /NPAK**
NPAK < 0	->	**NBITS = -NPAK**

On a 32-bit UNIX system, **NPAK = 4** means that 8 bits are kept for each floating point value; **NPAK=-16** means that 16 bits are kept for each value.

We recommend to use a value of **NPAK < 0**, so that the number of bits kept per item will always be absolute. As mentioned above, a value of -12 or -16 provides a good compromise between disk space and arithmetic precision. Positive values of NPAK are likely to be rejected by future versions of the software.

The RPN standard file library

The creation and manipulation of standard files can be done either by calling a set of FORTRAN integer functions (the RPN standard file library) or by using RPN utilities such as PGSM and EDITFST. The RPN standard file library contains a set of FORTRAN functions. 13 of these functions will be presented in this document, the others being reserved for a more specialized usage. All these functions return an INTEGER code. They can be divided in 5 categories.

- **file access and status**
  - FSTOUV (open a standard file)
  - FSTFRM (close a standard file)
  - FSTRWD (rewind a sequential standard file)
- **write records**
  - FSTECR (write a record)
- **erase records**
  - FSTEFF (erase a record in a random standard file)
- **read and query records**
  - FSTLIR (read a field using selection criteria)
  - FSTLIS (read the next field using selection criteria defined in FSTLIR)
  - FSTLUK (read a field using a key pointing to a record)
- **query records**
  - FSTINF (retrieve the key of the first record meeting selection criteria)
  - FSTINL (retrieve multiple keys of records meeting selection criteria)
  - FSTNBR (find the number of records in a standard file)
  - FSTPRM (retrieve all the attributes of a record)
  - FSTSUI (retrieve the key of the next record meeting selection criteria)
  - FSTVOI (print the catalog of a standard file)

5 auxiliary routines are often used in conjunction with the standard file library: the routines FNOM and FCLOS to associate and dissociate a FORTRAN unit number to a filename, the routines CXGAIG and CIGAXG to encode/decode the grid attributes IG1, IG2, IG3 and IG4, and the routine INCDAT to compute the date of validity from the date of origin and vice-versa.

Effect of standard file function calls on file pointer position

The following table lists the differences between standard file types regarding current file pointer position **after** the following function calls. (BOC=Beginning of catalog, BOF=Beginning of file, EOF=End of file, N/A=Not applicable)

Function name	**RND type file**	**SEQ or SEQ/FTN type file**
FSTOUV	BOC	BOF
FSTFRM	Undefined	Undefined
FSTRWD	N/A	BOF
FSTECR	Undefined	EOF
FSTEFF	Undefined	N/A
FSTLIR	Current record	Next record
FSTLIS	Current record	Next record
FSTLUK	Current record	Next record
FSTINF	Current record	Current record

FSTINL	Current record	EOF
FSTNBR	Current record	N/A
FSTPRM	Current record	Current record
FSTSUI	Current record	Current record
FSTVOI	BOC	EOF

The following table lists the differences between standard file types regarding current file pointer position **before** calling query functions. (BOC=Beginning of catalog, BOF=Beginning of file, EOF=End of file, N/A=Not applicable)

Function name	RND type file	SEQ or SEQ/FTN type file
FSTLIR	BOC	Current record
FSTINF	BOC	Current record
FSTINL	BOC	Current record
FSTLIS	Current record	Current record
FSTSUI	Current record	Current record

A basic example: creating a standard file from ASCII data

The examples discussed in this section should be accessible on-line, in the directory **$ARMNLIB/demo**. To start working on your own, you can make a copy of this directory in your own HOME directory. Here is a suggested list of commands:

```
       % cd
       % mkdir fstd
       % cp $ARMNLIB/demo/* fstd
```

The first example covers the conversion of data stored in ASCII format into the RPN standard file format. The file contains climatological monthly surface temperatures on a 120x60 grid, covering the globe and defined on a lat-lon projection. The data is stored in file "**ts.asc**" and will be converted into file "**ts.fst**".

The grid has the following structure: point (1,1) represents the southwest corner of the globe
(-88.5 lat, 0 lon), while point (ni,nj) represents the northeast corner (88.5 lat, 357 lon). There is one grid point every 3 degrees of latitude and longitude.

```
                                              (NI, NJ)
   *** |---------------------------------------|
    *  |                                       |
    *  |                                       |
    *  |                                       |
    NJ |                                       |
    *  |                                       |
    *  |                                       |
    *  |                                       |
   *** |---------------------------------------|
 (1, 1) *                                     *
       |**************   NI  ******************
        *                                     *
```

The input file has the following format: the first line contains the month of the year, followed by field values encoded in the following order: i, j, fld(i,j), fld(i+1,j), fld(i+2,j), fld(i+3,j), fld(i+4,j). Here are the first ten lines of the file.

```
            1
   1   1 -28.6152 -28.6816 -28.6015 -28.6230 -28.8769
   6   1 -28.8496 -28.7461 -28.6621 -28.5722 -28.7051
  11   1 -28.5605 -28.5742 -28.4902 -28.2500 -28.3613
  16   1 -28.1113 -28.2285 -28.2441 -28.1621 -28.2715
  21   1 -28.2070 -28.1133 -28.0410 -27.9941 -28.0879
  26   1 -28.1465 -28.2676 -28.3437 -28.3730 -28.4004
  31   1 -28.4394 -28.5703 -28.7109 -28.8008 -28.6797
  36   1 -28.5703 -28.5156 -28.4394 -28.3730 -28.4785
  41   1 -28.5664 -28.5410 -28.6367 -28.7871 -28.5468
```

The FORTRAN code is stored in the file "**ex1.f**". A copy of the whole program appears in Appendix A. The "**ex1**" program uses 5 routines from the standard file library. Here is an overview of what is done by the program.

- Declare variables
- Associate a file name with a FORTRAN unit number (**FNOM**)
- Open the file (**FSTOUV**)
- Initialize proper standard file attributes
- Read the data contained in the ASCII file
- Write records into the standard file (**FSTECR**)
- Close the file (**FSTFRM**)
- Dissociate the file name from the FORTRAN unit number (**FCLOS**).

Let's look more closely at the different parts of the program.

Declare variables

In the first part we use the attributes names that were suggested in the "**Detailed structure**" section.

```
      ---------------------------------------------------
      character*2 nomvar
      character*1 typvar, grtyp
      character*8 etiket

      integer key, dateo, deet, npas, ni, nj, nk, npak, datyp
      integer ip1, ip2, ip3
      integer ig1, ig2, ig3, ig4
      logical rewrit
```

We then declare the functions used by the program.

```
      ---------------------------------------------------
      external fstecr
      external fnom, fstouv, fclos, fstfrm

      integer fstecr
      integer fnom, fstouv, fclos, fstfrm
      ---------------------------------------------------
```

We continue with the other variables. "**ier**" contains the return status of the functions used in the examples. "**iun**" contains the logical FORTRAN unit. "**month**" contains the month index that will be used to initialize the date of origin. "**fld**" will contain the values of surface temperature for each month, and "**work**" contains a working storage area for the "**fstecr**" function. Although there is a precise formula to compute and reduce the size of the "**work**" array, we will keep things simple and initialize it with the same dimensions as "**fld**".

```
      ---------------------------------------------------
      integer ier
      integer i,j,ii,jj,iun
      integer month
      real fld(120, 60), work(120, 60)
      ---------------------------------------------------
```

Associate a file name with a FORTRAN unit number (FNOM)

```
      ---------------------------------------------------
      iun = 1
      ier = fnom(iun, 'ts.fst', 'STD+RND', 0)
      if (ier.lt.0) then
         print *, 'Fatal error while opening the file'
      endif
      ---------------------------------------------------
```

Open the file in random access mode (FSTOUV)

```
      ---------------------------------------------------
      iun = 1
      ier = fstouv(iun, 'STD+RND')
      ---------------------------------------------------
```

Initialize proper standard file attributes

```
      ---------------------------------------------------
      typvar = 'C'
      nomvar = 'TS'
      etiket = 'SFC TEMP '

      ip1 = 0
      ip2 = 0
      ip3 = 0

      ni = 120
      nj = 60
      nk = 1
      deet = 0
      npas = 0

      grtyp  = 'A'
      ig1 = 0
      ig2 = 0
      ig3 = 0
      ig4 = 0

      datyp = 1
      npak = -16
      ---------------------------------------------------
```

Read the data contained in the ASCII file.

We assume here that we are reading data from the console (through standard UNIX redirection) and that we know exactly the format of the input data.

```
      ---------------------------------------------------
        read(5, *) month
         do 200 j=1,120*60/5
            read(5,*) ii,jj,fld(ii,jj),fld(ii+1,jj),fld(ii+2,jj),
     *                fld(ii+3,jj),fld(ii+4,jj)
 200        continue
      ---------------------------------------------------
```

Write records in the standard file (FSTECR).

We set the date so that each climatological average has the 1st of each month as date of origin. We then call integer function FSTECR with the attributes that were set at the beginning of the program. The last argument of the routine is a flag indicating that we don't want to replace any existing record that would have the same search attributes (ip1, ip2, ip3, typvar, nomvar, etiket). Note that the date of origin (DATEO) is not included in the search attributes.

```
      ---------------------------------------------------
       dateo = month * 10000000 + 0199000
       ier = fstecr(fld, WORK, npak, iun, dateo, deet, npas, ni, nj,
     *                nk, ip1, ip2, ip3, typvar, nomvar, etiket, grtyp,
     *                ig1, ig2, ig3, ig4, datyp, .false.)
      ---------------------------------------------------
```

Close the file (FSTFRM)

```
      ---------------------------------------------------
      ier = fstfrm(1)
      ---------------------------------------------------
```

Dissociate the file name from the FORTRAN unit number (FCLOS).

```
      ---------------------------------------------------
      ier = fclos(1)
      ---------------------------------------------------
```

To compile the program, type

```
% f77 ex1.f -o ex1 $ARMNLIB/lib/rmnxlib.a
```

To execute the program, type

```
% ex1 < ts.asc
```

The program should then execute, producing the following output:

```
 UNIT =  1 RANDOM EST CREE
 UNIT =  1 EST OUVERT  RANDOM
 ECRIT( 1)       0-TS C     0     0    0   120    60     1 SFC TEMP 4010199000      0     0  A     0     0     0     0 R16    1531  3604
 ECRIT( 1)       1-TS C     0     0    0   120    60     1 SFC TEMP 7020199000      0     0  A     0     0     0     0 R16    5161  3604
 ECRIT( 1)       2-TS C     0     0    0   120    60     1 SFC TEMP 1030199000      0     0  A     0     0     0     0 R16    8791  3604
 ECRIT( 1)       3-TS C     0     0    0   120    60     1 SFC TEMP 4040199000      0     0  A     0     0     0     0 R16   12421  3604
 ECRIT( 1)       4-TS C     0     0    0   120    60     1 SFC TEMP 6050199000      0     0  A     0     0     0     0 R16   16051  3604
 ECRIT( 1)       5-TS C     0     0    0   120    60     1 SFC TEMP 2060199000      0     0  A     0     0     0     0 R16   19681  3604
 ECRIT( 1)       6-TS C     0     0    0   120    60     1 SFC TEMP 4070199000      0     0  A     0     0     0     0 R16   23311  3604
 ECRIT( 1)       7-TS C     0     0    0   120    60     1 SFC TEMP 7080199000      0     0  A     0     0     0     0 R16   26941  3604
 ECRIT( 1)       8-TS C     0     0    0   120    60     1 SFC TEMP 3090199000      0     0  A     0     0     0     0 R16   30571  3604
 ECRIT( 1)       9-TS C     0     0    0   120    60     1 SFC TEMP 5100199000      0     0  A     0     0     0     0 R16   34201  3604
 ECRIT( 1)      10-TS C     0     0    0   120    60     1 SFC TEMP 1110199000      0     0  A     0     0     0     0 R16   37831  3604
 ECRIT( 1)      11-TS C     0     0    0   120    60     1 SFC TEMP 3120199000      0     0  A     0     0     0     0 R16   41461  3604
 UNITE FORTRAN IUN=   1 EST FERME
```

Let's look at the first two lines of the output.

```
 UNIT =  1 RANDOM EST CREE
 UNIT =  1 EST OUVERT  RANDOM
```

These lines show that the file associated with logical unit 1 is created (**UNIT = 1 RANDOM EST CREE**), and then opened in random access mode (**UNIT = 1 EST OUVERT RANDOM**).

We then have a message for each record that has been created.

```
ECRIT( 1)      0-TS C     0     0    0   120    60    1 SFC TEMP 4010199000      0     0  A
0    0     0     0 R16    1531  3604
```

The contents of the line show that a record has been written on unit 1, followed by a list of attributes. The order of the attributes is key(**0**), var. name(**TS**), field type(**C**), ip1(**0**), ip2(**0**), ip3(**0**), ni(**120**), nj(**60**), nk(**1**), label(**SFC TEMP**), date of origin(**4010199000**), deet(**0**), npas(**0**), grtyp(**A**), ig1(**0**), ig2(**0**), ig3(**0**), ig4(**0**), data type and packing ratio(**R16**) and finally two other attributes reserved for internal use: swa(**1531**) and lng (**3604**).

The name of the standard file produced by the program is **ts.fst**. We can inspect the contents of this file by invoking the RPN utility **voir**.

```
% voir -iment ts.fst
1
  ************************************************************************************************
  *                                                                                              *
  *           VOIR                                                             3.2              *
  *                                                                                              *
  *                                                                                              *
  *        Thu Aug 20 11:11:43 1999                                                              *
  *                                                                                              *
  *        BEGIN  EXECUTION                                                                      *
  *                                                                                              *
  ************************************************************************************************
 UNIT = 10 EST OUVERT  RANDOM
1 FILE=ts.fst                                          TYPE=RANDOM       Thu Aug 20 1999  11:11:44     PAGE   1

      KEY#  ID    IP1    IP2  IP3    NI     NJ    NK  ETIQ.   DATE ORIG.   DEET  NPAS GR  IG1  IG2   IG3    IG4 DTY     SWA   LNG

        0-TS C     0      0    0   120    60     1 SFC TEMP 4010199000      0     0  A    0    0     0      0 R16    1531  3604
        1-TS C     0      0    0   120    60     1 SFC TEMP 7020199000      0     0  A    0    0     0      0 R16    5161  3604
        2-TS C     0      0    0   120    60     1 SFC TEMP 1030199000      0     0  A    0    0     0      0 R16    8791  3604
        3-TS C     0      0    0   120    60     1 SFC TEMP 4040199000      0     0  A    0    0     0      0 R16   12421  3604

        4-TS C     0      0    0   120    60     1 SFC TEMP 6050199000      0     0  A    0    0     0      0 R16   16051  3604
        5-TS C     0      0    0   120    60     1 SFC TEMP 2060199000      0     0  A    0    0     0      0 R16   19681  3604
        6-TS C     0      0    0   120    60     1 SFC TEMP 4070199000      0     0  A    0    0     0      0 R16   23311  3604
        7-TS C     0      0    0   120    60     1 SFC TEMP 7080199000      0     0  A    0    0     0      0 R16   26941  3604

        8-TS C     0      0    0   120    60     1 SFC TEMP 3090199000      0     0  A    0    0     0      0 R16   30571  3604
        9-TS C     0      0    0   120    60     1 SFC TEMP 5100199000      0     0  A    0    0     0      0 R16   34201  3604
       10-TS C     0      0    0   120    60     1 SFC TEMP 1110199000      0     0  A    0    0     0      0 R16   37831  3604
       11-TS C     0      0    0   120    60     1 SFC TEMP 3120199000      0     0  A    0    0     0      0 R16   41461  3604


 STATISTIQUES

 DIMENSION DU DIRECTEUR DISQUE       100
 NOMBRE D ENTREES UTILISEES           12
 LONGUEUR DU FICHIER               45090 MOTS
 NOMBRE D ECRITURES                   12
 NOMBRE DE RE-ECRITURES                0
 NOMBRE D EFFACAGES                    0
 NOMBRE D EXTENSIONS                   0
 NOMBRE DE CORRECTIONS                 0

 *******************************************
 UNITE FORTRAN IUN=  10 EST FERME

  ************************************************************************************************
  *                                                                                              *
  *           VOIR                                                             O.K.             *
  *                                                                                              *
  *        Thu Aug 20 11:11:44 1999                                                              *
  *                                                                                              *
  *        END EXECUTION                                                                         *
  *                                                                                              *
  *        CP SECS =      0.100                                                                  *
  *                                                                                              *
  ************************************************************************************************
```

A second example: querying the contents of a standard file

Most of the RPN standard file functions (9 out 13) are meant for querying and reading records. As mentioned in the "**Classification by category**" section, queries can be targeted to records using the following attributes:

Data element	Suggested name
Variable name	NOMVAR
Type of field	TYPVAR
Label	ETIKET
Vertical level	IP1
Forecast hour	IP2
User defined index	IP3
Date of validity	DATEV=DATEO+ DEET * NPAS

Queries can be made by using precise values for the attributes, or by wildcarding some of them. Wildcarding an attribute means to ignore it when making a search. A standard file query is usually of the type "*Give me the key of the first record where NOMVAR='GZ' and TYPVAR='P' and ETIKET='FE OPRUN' and IP1=1000 and IP2=3 and IP3=0 and DATEV=039217120*". A query where IP1, IP2 and IP3 would be wildcarded (and thus ignored during the search) would look like "*Give me the key of the first record where NOMVAR='GZ' and TYPVAR='P' and ETIKET='FE OPRUN'* .

Wildcarding the integer attributes IP1, IP2, IP3 and DATEV is done by assigning them a value of **-1**; wildcarding the character attributes NOMVAR, TYPVAR and ETIKET is done by using a single blank, **' '**.

The program "**ex2.f**", printed in appendix A, computes simple statistics from the "**ts.fst**" file created in example 1. The program reads records meeting certain search criteria (functions FSTLIR and FSTLIS), finds their average, minimum and maximum values (subroutine STATFLD), and then prints the value of all the standard file attributes. Here is an overview of what is done by the program.

- Declare variables
- Associate a file name with a FORTRAN unit number (**FNOM**)
- Open the file (**FSTOUV**)
- Initialize proper standard file attributes needed by the **FSTLIR** function
- Read the first record matching search criteria (**FSTLIR**)
- Find minimum, maximum and average values (**STATFLD - included in the main program**)
- Get and print the values of all standard file attributes (**FSTPRM**)
- Repeat until no more records are found (**FSTLIS**)
- Close the standard file (**FSTFRM**)
- Dissociate the file name from the FORTRAN unit number (**FCLOS**).

The part of the code of "**ex2.f"** that declares variables and opens the standard file is very similar to "**ex1.f**". We will discuss only the parts of the program that are different.

The first part is a call to the "**FSTNBR**" function, which return the number of records existing in a random standard file.

```
      ---------------------------------------------------
****
*     Get the number of records in the standard file
*     This function can only be used for random standard files
****
      nrecs = fstnbr(iun)
      print *, 'There are ', nrecs, ' records in that file'
      ---------------------------------------------------
```

The second part is a call to the "**FSTVOI**" function, which produces a listing (on standard output) of the attributes of all existing records in the standard file. This listing is identical to the one produced by the invocation of the "**VOIR**" utility.

```
      ---------------------------------------------------
****
*     Print the contents of the standard file directory
****
      ier = fstvoi(iun, 'STD+RND')
      if (ier.lt.0) then
         print *, '(FSTVOI) Cannot print the directory'
      endif
      ---------------------------------------------------
```

The next part initializes the standard file attributes to initiate a query for the first occurence of the variable '**TS**', where TYPVAR='**C**', ETIKET='**SFC TEMP** ', and where IP1, IP2 and IP3 are all zero. Note that DATEV (the date of validity) is the only wild card attribute in this call.

```
      ---------------------------------------------------
****
*     Initialize standard file variables for doing a query
****

      typvar = 'C'
      nomvar = 'TS'
      etiket = 'SFC TEMP '
      datev  = -1
      ip1 = 0
      ip2 = 0
      ip3 = 0
```

The FSTLIR function locates the first record meeting the search criteria, and then loads the field values into the FLD array.[2]

```
      ---------------------------------------------------
****
*     Reads the first field matching selection criteria
****
      key = fstlir(FLD, iun, NI, NJ, NK, datev, etiket,
     *             ip1, ip2, ip3, typvar, nomvar)
      ---------------------------------------------------
```

[2] This function can potentially be dangerous to use if the size of the record read is larger than the size allocated in memory. Note that in all the query functions, the values of NI, NJ and NK are updated to be the same as those of the selected standard file record.

The FSTLIR function returns a negative value if it cannot locate a record matching the search criteria.

```
       ---------------------------------------------------
50   if (key.lt.0) then
         print *, '(FSTLIR) Invalid key number:', key
      ---------------------------------------------------
```

The next part of the code contains a loop that reads all the other records in the standard file meeting the search criteria (using "**FSTLIS**"), until the end of file is reached (i.e. until a key less than zero is returned). For each record found, we compute its minimum, maximum and average values (subroutine **STATFLD**), and get the values of its attributes using the "**FSTPRM**" function.

Here is a schematic part of the loop

```
       --->   if (key.lt.0) then
       |          print error message
       |      else
       |          compute min, max, avg
       |          get and print the value of all attributes
       |      endif
       |      read next field matching search criteria
       |-----------
```

and the real code

```
       ---------------------------------------------------
50   if (key.lt.0) then
         print *, '(FSTLIR) Invalid key number:', key
      else
****
*        Computes minimum, maximum and average value of the field
****
         call statfld(minval, maxval, avgval, fld, ni, nj)
****
*        Get all standard file parameters and print them
****
         ier  = fstprm(key, dateo, deet, npas, ni, nj, nk,
     *                 nbits, datyp, ip1, ip2, ip3,
     *                 typvar, nomvar, etiket, grtyp,
     *                 ig1, ig2, ig3, ig4, swa, lng, dltf, ubc,
     *                 extra1, extra2, extra3)

         print *, '*****************************************'
         print *, ' minval = ', minval, 'maxval =', maxval,
     *            'avgval = ', avgval

         print 10, nomvar, typvar, etiket, dateo, deet, npas,
     *             ni, nj, nk,nbits, datyp, ip1, ip2, ip3,
     *             grtyp, ig1, ig2, ig3, ig4,
     *             swa, lng, dltf, ubc, extra1, extra2, extra3

****
*        Try to read the next field matching selection criteria set
*        by the first call to FSTLIR.
****
         key  = fstlis(fld, iun, NI, NJ, NK)
         goto 50
      endif
      ---------------------------------------------------
```

Here is the output produced by the program:

```
 *****************************************
 LU( 1)      0-TS C     0     0    0   120    60     1 SFC TEMP 6010199000      0     0  A    0    0     0     0
R16    1531  3604
 *****************************************
  minval =   -48.25580    maxval =   31.51569    avgval =    3.220830
  nomvar=        TS typvar=         C etiket=  SFC TEMP
  dateo=   10199000 deet=           0 npas=           0
  ni=           120 nj=            60 nk=             1
  nbits=         16 datyp=          1
  ip1=            0 ip2=            0 ip3=            0
  grtyp=          A ig1=            0 ig2=            0 ig3=            0 ig4=            0
  swa=         1531 lng=         3604 dltf=           0 ubc=            0
  extra1=         0 extra2=         0 extra3=         0
 LU( 1)      1-TS C     0     0    0   120    60     1 SFC TEMP 2020199000      0     0  A    0    0     0     0
R16    5161  3604
 *****************************************
  minval =   -49.15850    maxval =   30.90791    avgval =    2.926342
  nomvar=        TS typvar=         C etiket=  SFC TEMP
  dateo=   20199000 deet=           0 npas=           0
  ni=           120 nj=            60 nk=             1
  nbits=         16 datyp=          1
  ip1=            0 ip2=            0 ip3=            0
  grtyp=          A ig1=            0 ig2=            0 ig3=            0 ig4=            0
  swa=         5161 lng=         3604 dltf=           0 ubc=            0
  extra1=         0 extra2=         0 extra3=         0
 LU( 1)      2-TS C     0     0    0   120    60     1 SFC TEMP 2030199000      0     0  A    0    0     0     0
R16    8791  3604
 *****************************************
  minval =   -63.80290    maxval =   30.91976    avgval =    2.481087
  nomvar=        TS typvar=         C etiket=  SFC TEMP
  dateo=   30199000 deet=           0 npas=           0
  ni=           120 nj=            60 nk=             1
  nbits=         16 datyp=          1
  ip1=            0 ip2=            0 ip3=            0
  grtyp=          A ig1=            0 ig2=            0 ig3=            0 ig4=            0
  swa=         8791 lng=         3604 dltf=           0 ubc=            0
  extra1=         0 extra2=         0 extra3=         0

(...OUTPUT TOO LONG TO BE PRINTED HERE)
```

Other query methods

The remaining part of this document will show alternative methods of locating the records the program "**ex2.f**" searches for.

It may sometimes be useful to retrieve record information before reading data values. For example it may be valuable to verify the size of a record (in terms of NI, NJ, NK) before loading it into memory. This can be done by using the "**FSTINF**" function. The next occurence of a record meeting the search criteria defined by "**FSTINF**" can be found be using the "**FSTSUI**" function.

Once a record has been located by "**FSTINF**", it can be loaded in memory by a call to "**FSTLUK**", which uses the key returned by "**FSTINF**". In fact, the code

```
              key1 = fstlir(FLD, iun, NI, NJ, NK, datev, etiket,
                            ip1, ip2, ip3, typvar, nomvar)
              key2 = fstlis(FLD, iun, NI, NJ, NK)
```

produces the same result as

```
             key1 = fstinf(iun, NI, NJ, NK, datev, etiket,
                                   ip1, ip2, ip3, typvar, nomvar)
             ier  = fstluk(FLD, key1, NI, NJ, NK)
             key2 = fstsui(iun, NI, NJ, NK)
             ier  = fstluk(FLD, key2, NI, NJ, NK)
```

Sample code using "**FSTINF**", "**FSTLUK**" and "**FSTSUI**" can be found in the program "**ex3.f**".

The last function to be presented in this document is "**FSTINL**". The "**FSTINL**" calling sequence is similar to "**FSTINF**", except that it returns a list of record keys satisfying search criteria. So, instead of executing the following loop **NKEYS** times,

```
             key = fstinf(iun, NI, NJ, NK, datev, etiket,
                                   ip1, ip2, ip3, typvar, nomvar)
      -->    if (key > 0) then
      |         do something
      |         key = fstsui(iun, NI, NJ, NK)
      |      endif
      ------------
```

it is possible to use

```
             integer maxkeys
             parameter (maxkeys = 100)
             integer keys(maxkeys), nkeys

             (...)

             ier = fstinl(iun, NI, NJ, NK, datev, etiket,
                                   ip1, ip2, ip3, typvar, nomvar,
                            KEYS, NKEYS, nmax)

             do 100 i=1, nkeys
                ier = fstluk(FLD, keys(i), NI, NJ, NK)
                do something
      100    continue
```

Sample code using "**FSTINL**" can be found in the file "**ex4.f**".

Appendix A

```
      program ex1
      implicit none

****
*     This program converts climatological monthly surface
*     temperatures stored in ASCII format
*     into the RPN standard file format.
****

****
*     Declare variables used by the RPN standard file library
****
      character*2 nomvar
      character*1 typvar, grtyp
      character*8 etiket

      integer key, dateo, deet, npas, ni, nj, nk, npak, datyp
      integer ip1, ip2, ip3
      integer ig1, ig2, ig3, ig4

****
*     Declare the name and type of the RPN standard file functions
****
      external fstecr
      external fnom, fstouv, fclos, fstfrm

      integer fstecr
      integer fnom, fstouv, fclos, fstfrm

****
*     Declare other variables used by the program
****
      integer ier, nrecs
      integer i,j,ii,jj,iun
      integer month
      real fld(120, 60), work(120, 60)

****
*     Association of the RPN standard file produced by the
*     program with the FORTRAN logical unit 1.
****
      iun = 1
      ier = fnom(iun, 'TS.FST', 'STD+RND', 0)
      if (ier.lt.0) then
         print *, 'Fatal error while opening the file (FNOM)'
         stop
      endif

****
*     Opening of the standard file
****
      iun = 1
      ier = fstouv(iun, 'RND')
      if (ier.lt.0) then
         print *, 'Cannot open unit:', iun,
     *            ' in random access mode (FSTOUV)'
         stop
      endif


****
*     Initialization of the standard file attributes that remain
*     constant for all fields
****
      typvar = 'C'
      nomvar = 'TS'
      etiket = 'SFC TEMP '

      ip1 = 0
      ip2 = 0
      ip3 = 0

      ni = 120
      nj = 60
      nk = 1
      deet = 0
      npas = 0

      grtyp  = 'A'
      ig1 = 0
      ig2 = 0
      ig3 = 0
      ig4 = 0

      datyp = 1
      npak = -16

****
*     Start loop over the 12 months of the year
****
      do 100 i=1,12
****
*        read month and field contents
****
         read(5, *) month
         do 200 j=1,120*60/5
            read(5,*) ii,jj,fld(ii,jj),fld(ii+1,jj),fld(ii+2,jj),
     *                 fld(ii+3,jj),fld(ii+4,jj)
 200        continue

****
*        Set a date equal the 1st of each month in 1999
****
         dateo = month * 10000000 + 0199000

****
*        Write a standard file record
****
         ier = fstecr(fld, WORK, npak, iun, dateo, deet, npas, ni, nj,
     *                 nk, ip1, ip2, ip3, typvar, nomvar, etiket, grtyp,
     *                 ig1, ig2, ig3, ig4, datyp, .false.)

 100  continue

****
*     Close the standard file
****
      ier = fstfrm(1)

****
*     Unlink the unit 1 from the file "ts.fst"
****
      ier = fclos(1)

      stop
      end
```

program ex2

```
      implicit none

****
*     This program uses the RPN standard file functions
*     FSTNBR, FSTVOI, FSTLIR and FSTPRM to make queries about the
*     contents of an RPN standard file.
*
*     Author: Yves Chartier
*     Last revision: September 1992.
****

****
*     Declare variables used by the RPN standard file library
****
      character*2 nomvar
      character*1 typvar, grtyp
      character*8 etiket

      integer key, dateo, datev, deet, npas, ni, nj, nk
      integer npak, datyp, nbits
      integer ip1, ip2, ip3
      integer ig1, ig2, ig3, ig4
      integer swa, lng, dltf, ubc, extra1, extra2, extra3

****
*     Declare the name and type of the RPN standard file functions
****
      integer fstecr, fstlir, fstlis, fstprm
      integer fnom, fstouv, fclos, fstfrm, fstnbr, fstvoi, fsteof

****
*     Declare other variables used by the program
****
      integer ier, nrecs
      integer i,j,ii,jj,iun, iunout
      integer month
      real fld(120, 60)
      real minval, maxval, avgval

****
*     Association of the RPN standard file produced by the
*     program with the FORTRAN logical unit 1.
****
      iun = 1

      ier = fnom(iun, 'ts.fst', 'STD+RND', 0)
      if (ier.lt.0) then
         print *, 'Fatal error while opening the file (FNOM)'
      endif

****
*     Opening of the standard file
****
      ier = fstouv(iun, 'RND')
      if (ier.lt.0) then
         print *, 'Cannot open unit:', iun,
     *            ' in random access mode (FSTOUV)'
         stop
      endif

****
*     Get the number of records in the standard file
****
      nrecs = fstnbr(iun)
      print *, 'There are ', nrecs, ' records in that file'

****
*     Print the contents of the standard file directory
****

      ier = fstvoi(iun, 'STD+RND')
      if (ier.lt.0) then
         print *, '(FSTVOI) Cannot print the directory'
      endif

****
*     Initialize standard file variables for doing a query
****
```

```
      typvar = 'C'
      nomvar = 'TS'
      etiket = 'SFC TEMP '
      datev  = -1
      ip1 = 0
      ip2 = 0
      ip3 = 0

****
*     Reads the first field meeting selection criteria
****

      key = fstlir(fld, iun, NI, NJ, NK, datev, etiket,
     *             ip1, ip2, ip3, typvar, nomvar)

 50   if (key.lt.0) then
         print *, '(FSTLIR) Invalid key number:', key
      else
****
*        Computes minimum, maximum and average value of the field
****
         call statfld(minval, maxval, avgval, fld, ni, nj)
****
*        Get all standard file parameters and print them
****
         ier  = fstprm(key, dateo, deet, npas, ni, nj, nk,
     *                 nbits, datyp, ip1, ip2, ip3,
     *                 typvar, nomvar, etiket, grtyp,
     *                 ig1, ig2, ig3, ig4, swa, lng, dltf, ubc,
     *                 extra1, extra2, extra3)

         print *, '****************************************'
         print *, ' minval = ', minval, 'maxval =', maxval,
     *            'avgval = ', avgval

         print 10, nomvar, typvar, etiket, dateo, deet, npas,
     *             ni, nj, nk,nbits, datyp, ip1, ip2, ip3,
     *             grtyp, ig1, ig2, ig3, ig4,
     *             swa, lng, dltf, ubc, extra1, extra2, extra3

****
*        Try to read the next field meeting selection criteria set
*        by the first call to FSTLIR.
****
         key  = fstlis(fld, iun, NI, NJ, NK)
         goto 50
      endif

****
*     Close the standard file
****
      ier = fstfrm(1)

****
*     Unlink the unit 1 from the file "ts.fst"
****
      ier = fclos(1)

 10   format(' ', ' nomvar=', a10, ' typvar=', a10, ' etiket=', a10, /,
     *       ' ', ' dateo= ', i10, ' deet=  ', i10, ' npas=  ', i10, /,
     *       ' ', ' ni=    ', i10, ' nj=    ', i10, ' nk=    ', i10, /,
     *       ' ', ' nbits= ', i10, ' datyp= ', i10, /,
     *       ' ', ' ip1=   ', i10, ' ip2=   ', i10, ' ip3=   ', i10, /,
     *       ' ', ' grtyp= ', a10, ' ig1=   ', i10, ' ig2=   ', i10,
     *            ' ig3=   ', i10, ' ig4=   ', i10, /,
     *       ' ', ' swa=   ', i10, ' lng=   ', i10, ' dltf=  ', i10,
     *            ' ubc=   ', i10, /,
     *       ' ', ' extra1=', i10, ' extra2=', i10, ' extra3=', i10)

      stop
      end
c     ***************************************************************
c     **                                                           **
c     ***************************************************************

      subroutine statfld(minval, maxval, avgval, fld, ni, nj)
      implicit none

      real minval, maxval, avgval
      real fld(ni,nj)
      integer i,j, ni, nj

      minval = fld(1,1)
      maxval = fld(1,1)
      avgval = 0.0

      do 100 j=1,nj
         do 100 i=1,ni
            avgval = avgval + fld(i,j)

            if (fld(i,j).lt.minval) then
               minval = fld(i,j)
            endif

            if (fld(i,j).gt.maxval) then
               maxval = fld(i,j)
            endif

 100  continue

      avgval = avgval / (ni * nj)

      return
      end

c     *****************************************************************
c     **                                                             **
c     *****************************************************************
```

```
      program ex3
      implicit none

****
*     This program uses the RPN standard file functions
*     FSTINF, FSTSUI, FSTLUK and FSTPRM to make queries about the
*     contents of an RPN standard file.
*
*     Author: Yves Chartier
*     Last revision: September 1992.
****

****
*     Declare variables used by the RPN standard file library
****
      character*2 nomvar
      character*1 typvar, grtyp
      character*8 etiket

      integer key, dateo, datev, deet, npas, ni, nj, nk
      integer npak, datyp, nbits
      integer ip1, ip2, ip3
      integer ig1, ig2, ig3, ig4
      integer swa, lng, dltf, ubc, extra1, extra2, extra3

****
*     Declare the name and type of the RPN standard file functions
****
      integer fstecr, fstinf, fstsui, fstluk, fstprm
      integer fnom, fstouv, fclos, fstfrm

****
*     Declare other variables used by the program
****
      integer ier, nrecs
      integer i,j,ii,jj,iun, iunout
      integer month
      real fld(120, 60)
      real minval, maxval, avgval

****
*     Association of the RPN standard file produced by the
*     program with the FORTRAN logical unit 1.
****
      iun = 1

      ier = fnom(iun, 'ts.fst', 'STD+RND', 0)
      if (ier.lt.0) then
         print *, 'Fatal error while opening the file (FNOM)'
      endif

****
*     Opening of the standard file
****
      ier = fstouv(iun, 'RND')
      if (ier.lt.0) then
         print *, 'Cannot open unit:', iun,
     *            ' in random access mode (FSTOUV)'
         stop
      endif

****
*     Initialize standard file variables for doing a query
****

      typvar = 'C'
      nomvar = 'TS'
      etiket = 'SFC TEMP '
      datev  = -1
      ip1 = 0
      ip2 = 0
      ip3 = 0

****
*     Reads the first field meeting selection criteria
****

      key = fstinf(iun, NI, NJ, NK, datev, etiket,
     *     ip1, ip2, ip3, typvar, nomvar)
```

```
      if (ier.lt.0) then
         print *, '(FSTINF) No records found'
         go to 200
      endif

****
*     Try to read the next field meeting selection criteria set
*     by the first call to FSTINF.
****

 50   ier = fstluk(fld, key, NI, NJ, NK)
****
*     Computes minimum, maximum and average value of the field
****
      call statfld(minval, maxval, avgval, fld, ni, nj)
****
*     Get all standard file parameters and print them
****
      ier  = fstprm(key, dateo, deet, npas, ni, nj, nk,
     *     nbits, datyp, ip1, ip2, ip3,
     *     typvar, nomvar, etiket, grtyp,
     *     ig1, ig2, ig3, ig4, swa, lng, dltf, ubc,
     *     extra1, extra2, extra3)

      print *, '*****************************************'
      print *, ' minval = ', minval, 'maxval =', maxval,
     *     'avgval = ', avgval

      print 10, nomvar, typvar, etiket, dateo, deet, npas,
     *     ni, nj, nk,nbits, datyp, ip1, ip2, ip3,
     *     grtyp, ig1, ig2, ig3, ig4,
     *     swa, lng, dltf, ubc, extra1, extra2, extra3

      key = fstsui(iun, NI, NJ, NK)

      if (key.lt.0) then
         print *, '(FSTSUI) Invalid key number:', key
      else
         goto 50
      endif

****
*     Close the standard file
****
 200  ier = fstfrm(1)

****
*     Unlink the unit 1 from the file "ts.fst"
****
      ier = fclos(1)

 10   format(' ', ' nomvar=', a10, ' typvar=', a10, ' etiket=', a10, /,
     *       ' ', ' dateo= ', i10, ' deet=  ', i10, ' npas=  ', i10, /,
     *       ' ', ' ni=    ', i10, ' nj=    ', i10, ' nk=    ', i10, /,
     *       ' ', ' nbits= ', i10, ' datyp= ', i10, /,
     *       ' ', ' ip1=   ', i10, ' ip2=   ', i10, ' ip3=   ', i10, /,
     *       ' ', ' grtyp= ', a10, ' ig1=   ', i10, ' ig2=   ', i10,
     *            ' ig3=   ', i10, ' ig4=   ', i10, /,
     *       ' ', ' swa=   ', i10, ' lng=   ', i10, ' dltf=  ', i10,
     *            ' ubc=   ', i10, /,
     *       ' ', ' extra1=', i10, ' extra2=', i10, ' extra3=', i10)

      stop
      end
c     ****************************************************************
c     **                                                            **
c     ****************************************************************

      subroutine statfld(minval, maxval, avgval, fld, ni, nj)
      implicit none

      real minval, maxval, avgval
      real fld(ni,nj)
      integer i,j, ni, nj

      minval = fld(1,1)
      maxval = fld(1,1)
      avgval = 0.0
      do 100 j=1,nj
         do 100 i=1,ni
            avgval = avgval + fld(i,j)

            if (fld(i,j).lt.minval) then
               minval = fld(i,j)
            endif

            if (fld(i,j).gt.maxval) then
               maxval = fld(i,j)
            endif

 100  continue

      avgval = avgval / (ni * nj)

      return
      end

c     ****************************************************************
c     **                                                            **
c     ****************************************************************
```

```
      program ex4
      implicit none

****
*     This program uses the RPN standard file functions
*     FSTINL, FSTLUK and FSTPRM to make queries about the
*     contents of an RPN standard file.
*
*     Author: Yves Chartier
*     Last revision: September 1992.
****

****
*     Declare variables used by the RPN standard file library
****

      integer maxkeys
      parameter (maxkeys = 100)

      character*2 nomvar
      character*1 typvar, grtyp
      character*8 etiket

      integer keys(maxkeys), nkeys
      integer dateo, datev, deet, npas, ni, nj, nk
      integer npak, datyp, nbits
      integer ip1, ip2, ip3
      integer ig1, ig2, ig3, ig4
      integer swa, lng, dltf, ubc, extra1, extra2, extra3

****
*     Declare the name and type of the RPN standard file functions
****
      integer fstecr, fstinl, fstsui, fstluk, fstprm
      integer fnom, fstouv, fclos, fstfrm

****
*     Declare other variables used by the program
****
      integer ier, nrecs
      integer i,j,ii,jj,iun, iunout
      integer month
      real fld(120, 60)
      real minval, maxval, avgval

****
*     Association of the RPN standard file produced by the
*     program with the FORTRAN logical unit 1.
****
      iun = 1

      ier = fnom(iun, 'ts.fst', 'STD+RND', 0)
      if (ier.lt.0) then
         print *, 'Fatal error while opening the file (FNOM)'
      endif

****
*     Opening of the standard file
****
      ier = fstouv(iun, 'RND')
      if (ier.lt.0) then
         print *, 'Cannot open unit:', iun,
     *            ' in random access mode (FSTOUV)'
         stop
      endif

****
*     Initialize standard file variables for doing a query
****

      typvar = 'C'
      nomvar = 'TS'
      etiket = 'SFC TEMP '
      datev  = -1
      ip1 = 0
      ip2 = 0
      ip3 = 0

****
*     Reads the first field meeting selection criteria
****

      ier = fstinl(iun, NI, NJ, NK, datev, etiket, ip1, ip2, ip3,
     *             typvar, nomvar, keys, nkeys, maxkeys)
      if (ier.lt.0) then
         print *, '(FSTINL) No records found'
         go to 200
      endif

****
*     Try to read the next field meeting selection criteria set
*     by the first call to FSTINF.
****

      do 50 i=1,nkeys
         ier = fstluk(fld, keys(i), NI, NJ, NK)
****
*     Computes minimum, maximum and average value of the field
****
         call statfld(minval, maxval, avgval, fld, ni, nj)
****
*     Get all standard file parameters and print them
****
         ier  = fstprm(keys(i), dateo, deet, npas, ni, nj, nk,
     *        nbits, datyp, ip1, ip2, ip3,
     *        typvar, nomvar, etiket, grtyp,
     *        ig1, ig2, ig3, ig4, swa, lng, dltf, ubc,
     *        extra1, extra2, extra3)

         print *, '*****************************************'
         print *, ' minval = ', minval, 'maxval =', maxval,
     *        'avgval = ', avgval

         print 10, nomvar, typvar, etiket, dateo, deet, npas,
     *        ni, nj, nk,nbits, datyp, ip1, ip2, ip3,
     *        grtyp, ig1, ig2, ig3, ig4,
     *        swa, lng, dltf, ubc, extra1, extra2, extra3

 50   continue
****
*     Close the standard file
****
 200  ier = fstfrm(1)

****
*     Unlink the unit 1 from the file "ts.fst"
****
      ier = fclos(1)

 10   format(' ', ' nomvar=', a10, ' typvar=', a10, ' etiket=', a10, /,
     *       ' ', ' dateo= ', i10, ' deet=  ', i10, ' npas=  ', i10, /,
     *       ' ', ' ni=    ', i10, ' nj=    ', i10, ' nk=    ', i10, /,
     *       ' ', ' nbits= ', i10, ' datyp= ', i10, /,
     *       ' ', ' ip1=   ', i10, ' ip2=   ', i10, ' ip3=   ', i10, /,
     *       ' ', ' grtyp= ', a10, ' ig1=   ', i10, ' ig2=   ', i10,
     *            ' ig3=   ', i10, ' ig4=   ', i10, /,
     *       ' ', ' swa=   ', i10, ' lng=   ', i10, ' dltf=  ', i10,
     *            ' ubc=   ', i10, /,
     *       ' ', ' extra1=', i10, ' extra2=', i10, ' extra3=', i10)

      stop
      end
c     ****************************************************************
c     **                                                            **
c     ****************************************************************
```

```
      subroutine statfld(minval, maxval, avgval, fld, ni, nj)
      implicit none

      real minval, maxval, avgval
      real fld(ni,nj)
      integer i,j, ni, nj

      minval = fld(1,1)
      maxval = fld(1,1)
      avgval = 0.0

      do 100 j=1,nj
         do 100 i=1,ni
            avgval = avgval + fld(i,j)

            if (fld(i,j).lt.minval) then
               minval = fld(i,j)
            endif

            if (fld(i,j).gt.maxval) then
               maxval = fld(i,j)
            endif

 100  continue

      avgval = avgval / (ni * nj)

      return
      end

c     ****************************************************************
c     **                                                            **
```

Appendix B

CODE	DESCRIPTION	UNITS
AL	ALBEDO	0 TO 1
AP	PLANETARY ALBEDO	0 TO 1
DD	DIVERGENCE	S-1
DP	ISOBARIC DIVERGENCE	S-1
EN	SNOW DEPTH (ECMWF)	M
ES	DEW POINT DEPRESSION	C
FC	HEAT FLUX FROM SURFACE TO THE TOP OF THE ATMOSPHERE	W/M2
FE	EVAPORATION FACTOR	0 TO 1
FI	INCOMING INFRA-RED SURFACE FLUX	W/M-2
FL	SOIL HEAT FLUX	W/M2
FQ	MOMENTUM FLUX AT SURFACE	PA
FS	INCOMING SURFACE SOLAR HEAT FLUX	W/M-2
FV	WATER VAPOR FLUX FROM SURFACE TO THE ATM.	KG/M-2/S
GL	ICE COVER	0 TO 1
GZ	GEOPOTENTIAL HEIGHT	DAM
HR	RELATIVE HUMIDITY	0 TO 1
HS	SOIL HUMIDITY	0 TO 1
HU	SPECIFIC HUMIDITY	KG/KG
MG	LAND/SEA MASK	0 TO 1
MT	TOPOGRAPHY (GEOPOTENTIEL MODELE)	M2.S-2
NB	LOW CLOUDS	0 TO 1
NE	SNOW COVER	0 TO 1
NU	CLOUD FRACTION	0 TO 1
NH	HIGH CLOUDS	0 TO 1
NM	MIDDLE CLOUDS	0 TO 1
PN	SEA LEVEL PRESSURE	MB
PR	ACCUMULATION OF PRECIPITATION	M
PS	SOIL CONDUCTIVITY	W.M-2.K-1
PO	SURFACE PRESSURE	MB
QQ	ABSOLUTE VORTICITY	S-1
QP	ISOBARIC VORTICITY	S-1
QR	RELATIVE VORTICITY	S-1
RR	RAINFALL RATE	M/S
SN	SNOW (QPF)	
TD	DEW POINT TEMPERATURE (DEW POINT)	C
TM	SEA TEMPERATURE	K OU C
TP	DEEP SOIL TEMPERATURE	C
TS	SURFACE TEMPERATURE	DEG K
TT	AIR TEMPERATURE	C
UU	U-COMPONENT OF THE WIND (EAST-WEST)	KNOTS
UV	WIND MODULUS	KNOTS
VT	VIRTUAL TEMPERATURE	C
VV	V-COMPONENT OF THE WIND (NORTH-SOUTH)	KNOTS
WP	ISOBARIC VERTICAL MOTION	PA/S
WS	SIGMA VERTICAL MOTION	S-1
WW	VERTICAL MOTION	MB/H
ZO	ROUGHNESS LENGTH	M

WARNING: SUBJECT TO CHANGE WITHOUT NOTICE

EXCERPT FROM EXISTING CODES FOR THE TYPVAR ATTRIBUTE IN RPN STANDARD FILES

```
CODE    DESCRIPTION
-----------------------------------------------------------------
--A,    ANALYSIS
--C,    Climatology
--D     Weather station raw data
--E     Monthly errors
--K     Various physical constants
--M     Verification matrix
--O     Observations
--P     Forecast
--Q     Quantity of precipitation forecast (QPF)
--S     Various scores
--T     Time Series
--X,    Varia
```

Appendix C

Conventions regarding the usage of grid descriptors in RPN standard files

In RPN standard files, the geographical location of a grid is defined by the parameters "**GRTYP**", "**IG1**", "**IG2**", "**IG3**" and "**IG4**"."**GRTYP**" is defined as CHARACTER*1, and **IG1** through **IG4** are INTEGERs.

The actual convention supports the following grids:

- Gaussian
- Polar stereographic (North and South)
- Cylindrical Equidistant (alias lat-lon)

One can also define, in a polar stereographic or lat-lon projection, and within certain limits, a cartesian grid with an irregular mesh (like the one used in the Finite Element model).

Here is the description of the grid types currently supported in RPN standard files. The following convention applies:

- NI: horizontal dimension of the grid
- NJ: vertical dimension of the grid
- in FORTRAN subroutine calls, input arguments are written in lower case, output arguments in capitals.

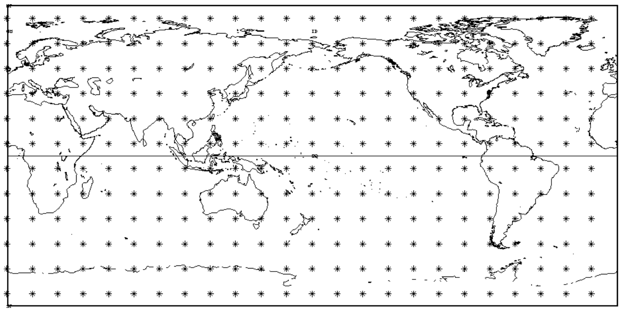
'A' grid:
This is a lat-lon grid covering either an hemisphere or the whole globe. There is no grid point at the pole and at the equator, and the first latitude has an offset of 0.5 grid point. The first longitude is 0° (the Greenwich meridian), and is not repeated at the end of the grid. The latitudinal grid length is 180/NJ for a global grid, 90/NJ otherwise. The longitudinal grid length is 360/NI. For such a grid,
IG1 contains the domain of the grid:

0: Global
1: Northern Hemisphere
2: Southern Hemisphere

IG2 contains the orientation of the grid:

0: South -> North (pt (1,1) is at the bottom of the grid)
1: North -> South (pt (1,1) is at the top of the grid)

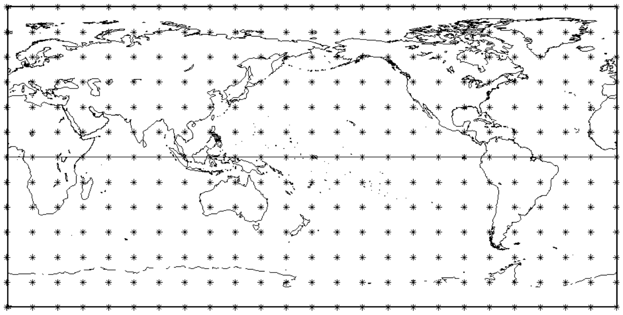
IG3 should be 0.
IG4 should be 0.



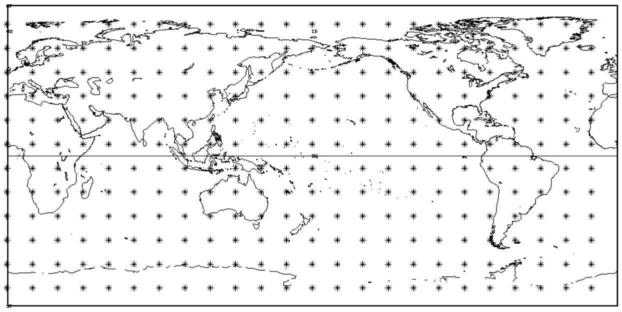
'B' grid:
The 'B' grid is a lat-lon grid covering either an hemisphere or the whole globe. There is a grid point at the pole and at the equator (if the grid is hemispheric or global with NJ odd). The first longitude is 0° (the Greenwich meridian), and is repeated at the end of the grid. The latitudinal grid length is 180/(NJ-1) for a global grid, 90/(NJ-1) otherwise. The longitudinal grid length is 360/(NI-1). For such a grid,
IG1 contains the domain of the grid:

0: Global
1: Northern Hemisphere
2: Southern Hemisphere

IG2 contains the orientation of the grid:
    0: South -> North (pt (1,1) is at the bottom of the grid)
    1: North -> South (pt (1,1) is at the top of the grid)
IG3 should be 0.
IG4 should be 0.



'G' grid:
The 'G' grid is a gaussian grid covering either an hemisphere or the whole globe. This grid is used in the spectral model; it is very much alike the 'A' grid, except that the latitudes are not equidistant. There is no grid point at the pole and at the equator. The first longitude is 0° (the Greenwich meridian), and is not repeated at the end of the grid. The longitudinal grid length is 360/NI. For such a grid,
IG1 contains the domain of the grid:
    0: Global
    1: Northern Hemisphere
    2: Southern Hemisphere
IG2 contains the orientation of the grid:
    0: South -> North (pt (1,1) is at the bottom of the grid)
    1: North -> South (pt (1,1) is at the top of the grid)
IG3 should be 0.
IG4 should be 0.



'L' grid:
The 'L' grid is a cylindrical equidistant grid (alias lat-lon). This grid is defined by the following parameters:
XLAT0: latitude of the southwest corner of the grid.
XLON0: longitude of the southwest corner of the grid.
DLAT: latitudinal grid length in degrees.
DLON: longitudinal grid length in degrees.

These parameters cannot be encoded directly into IG1, IG2, IG3 and IG4. To set the values of IG1, IG2, IG3 and IG4 from XLAT0, XLON0, DLAT and DLON, the conversion routine CXGAIG must be called. The calling sequence is:

```
CALL CXGAIG(grtyp, IG1, IG2, IG3, IG4, xlat0, xlon0, dlat, dlon)
```

Conversly, to get the values of XLAT0, XLON0, DLAT and DLON from IG1, IG2, IG3 and IG4, the conversion routine CIGAXG must be called. The calling sequence is:

```
CALL CIGAXG(grtyp, XLAT0, XLON0, DLAT, DLON, ig1, ig2, ig3, ig4)
```

The conversion from real to integer values may cause some loss of precision.

Precision of descriptors:
XLAT0, XLON0: 0.01°
DLAT, DLON: 0.001° if DLAT, DLON < 1°,
0.01 ° if DLAT, DLON >= 1 and <= 20°
1.0 ° if DLAT, DLON > 20 and <= 55°

'N' and 'S' grid:
These grids are polar stereographic; the 'N' grid is defined in the northern hemisphere, the 'S' grid in the southern hemisphere.
These grids are defined by the parameters PI, PJ, D60 and DGRW.
PI: Horizontal position of the pole, in grid points, from bottom left corner (1,1).
PJ: Vertical position of the pole, in grid points, from bottom left corner (1,1).
D60: grid length, in meters, at 60° of latitude.
DGRW: angle (between 0 and 360, +ve counterclockwise) between the Greenwich meridian and the horizontal axis of the grid.

As for the 'L' grid, one must use the routines CXGAIG and CIGAXG to do the conversion between PI, PJ, D60, DGRW and IG1, IG2, IG3 and IG4.

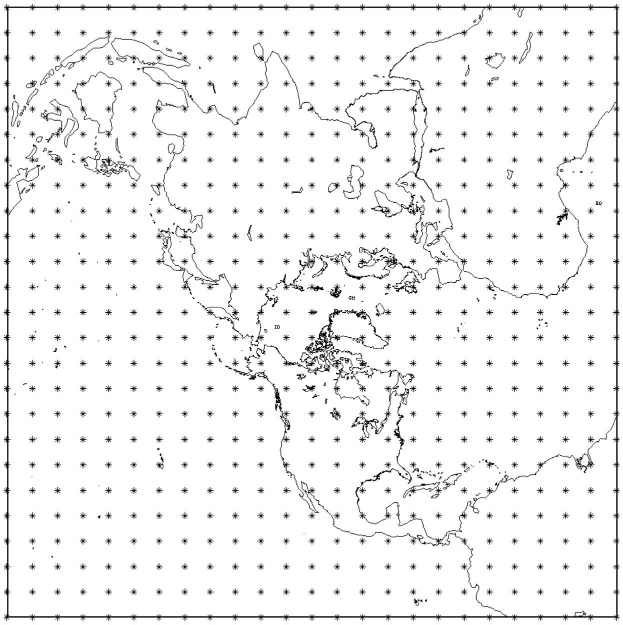
To set the values of IG1, IG2, IG3 and IG4 from PI, PJ, D60 and DGRW, the routine CXGAIG must be called. The calling sequence is:

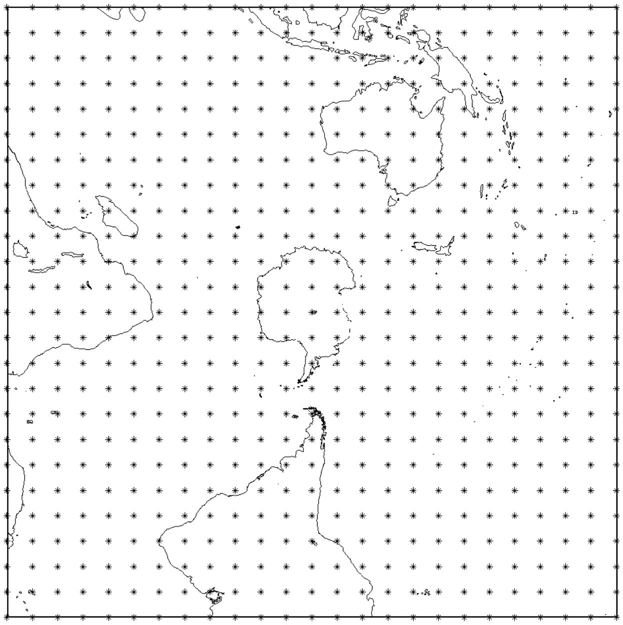
```
CALL CXGAIG(grtyp, IG1, IG2, IG3, IG4, pi, pj, d60, dgrw)
```

Conversly, to get the values of PI, PJ, D60 and DGRW, from IG1, IG2, IG3 and IG4, the routine CIGAXG must be called. The calling sequence is:

```
CALL CIGAXG(grtyp, PI, PJ, D60, DGRW, ig1, ig2, ig3, ig4)
```

Precision of the descriptors:
D60: 100 meters
DGRW: 0.01°
PI, PJ: 0.01 of a grid point if PI and PJ are +ve and < 200.
Otherwise, the lat-lon coordinates of the southwestern corner of the grid are kept with an error of about 0.01°.





'X' grid:
In the 'X' grid, the contents of the grid are not related to any geographical location on the earth. The parameters IG1 through IG4 should be set to 0. A typical application would be a theoretical experiment, like the evolution of a bubble in a cylinder.

'Y' grid:
The 'Y' grid represents a data set without a regular structure. A typical application is a record containing values for a stream of lat-lon points. For such a data set, one must use 2 special "positional" standard file records to define the location of each point: one record holding the horizontal positions of the points, and the other the vertical positions.

The parameter "NOMVAR" of the positional records has to be set to ">>" (an horizontal arrow) and to "^^"(a vertical arrow) for the records containing respectively the vertical and horizontal position of each point.

The "^^" and ">>" records can only be defined on 'L', 'N' and 'S' grids. The position of the points should be stored in grid point units. A common practice used to store absolute lat-lon coordinates is to define a lat-lon grid where the soutwest corner is positioned at (0° lat, 0° lon) and where the grid length is 1° in both directions. On such a grid, one can read and write lat-lon values without having to do conversions from grid space to geographical space.

The connection between one record and the associated positional records is done by "linking" the IG1, IG2 and IG3 descriptors of the data record to the IP1, IP2 and IP3 descriptors of the positional records. Consider the following example. We have here 2 positional records and 3 data records.

```
NOMVAR NI  NJ  IP1   IP2   IP3      GRTYP  IG1   IG2   IG3    IG4
------ --  --  ----  ----  ----     -----  ----  ----  ----   ----
^^     50  10  1001  1002  1003     L      100   100   9000   0
>>     50  10  1001  1002  1003     L      100   100   9000   0
GZ     50  10  1000  12    0        Y      1001  1002  1003   0
TT     50  10  1000  12    0        Y      1001  1002  1003   0
ES     50  10  1000  12    0        Y      1001  1002  1003   0
------ --  --  ----  ----  ----     -----  ----  ----  ----   ----
```

The following figure shows the link between data records and positional records.

```
NOMVAR NI  NJ  IP1   IP2   IP3      GRTYP  IG1   IG2   IG3    IG4
------ --  --  ----  ----  ----     -----  ----  ----  ----   ----
               --------------------
^^     50  10  | 1001  1002  1003 |   L    100   100   9000   0
>>     50  10  | 1001  1002  1003 |   L    100   100   9000   0
               --------------------
                        |
                        ----------------------------
                                                   |
                                         |------------------|
TT     50  10    1000  12    0        Y  | 1001  1002  1003 | 0
                                         |------------------|
------ --  --  ----  ----  ----     -----  ----  ----  ----   ----
```

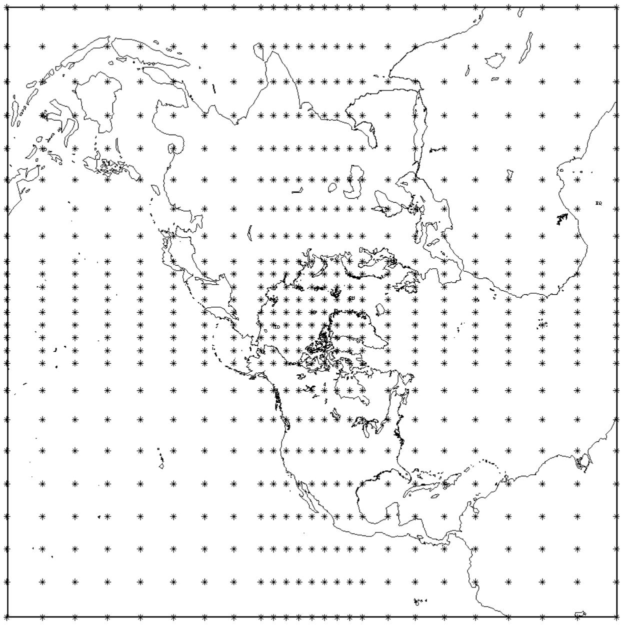
Here is a recipe to read positional records after having read a data record.

1- Do an FSTPRM of the record that has been read.
2- Get the IG1, IG2, IG3 values of that record.
3- Set IP1POS=IG1, IP2POS=IG2, IP3POS=IG3, and locate the corresponding "^^" and ">>" records with the help of an FSTINF call.

```
   ikeyver = FSTINF(iun, NIVER, NJVER, NKVER,
                    -1, -1, ip1pos, ip2pos, ip3pos, ' ','^^')
   ikeyhor = FSTINF(iun, NIHOR, NJHOR, NKHOR,
                    -1, -1, ip1pos, ip2pos, ip3pos, ' ','>>')
```

4- Read the positional records.

'Z' grid:
This grid is a cartesian grid with a non-constant mesh. As for the 'Y' grid, the deformation of the mesh is described with the help of the positional records "^^" and ">>". However, the positional records are 1-dimensional in each direction for this type of grid. The record containing the deformation of the grid should contain NI points on the X-axis, and NJ points on the Y-axis. As for the 'Y' grid, the GRTYP parameter of the positional records has to be 'L', 'N' or 'S', and values have to be stored in grid point units.



```
FSTARG1(3)                  STANDARD FILE SOFTWARE 1989                 FSTARG1(3)


NAME
    FSTARG1  -  Definition of arguments for standard file functions

DESCRIPTION
    FSTARG1 - lists, by alphabetical order, the definition of all function arguments used for
reading, writing, searching and erasing records in a standard file.

Alphabetical list of arguments

    NAME                DEFINITION

    buffer()            array containing the field to read or write

    cetiket             character value of 'etiket' key

    cgrtyp              character value of 'grtyp' key

    cnom                character value of 'nomvar' key

    ctype               character value of 'typvar' key

    dateo               date of origin of the field (MMDDYYHHR -
                        CMC DATE TIME STAMP without the first digit).

    datev               date of validity of the field (MMDDYYHHR -
                        CMC DATE TIME STAMP without the first digit)

    datyp               data type: 0 = binary, 1 = real, 2 = unsigned integer,
                                   3 = character,        4 = signed integer

    deet                time step length, in seconds (0 to 32767).

    dltf                erased field flag (1 = erase, 0 = non erased)

    etiket              eight character stamp

    extra1              reserved for future use

    extra2              reserved for future use

    extra3              reserved for future use

    grtyp               grid type

    hetiket             hollerith value of 'etiket' key

    hgrtyp              hollerith value of 'grtyp' key

    hnom                hollerith value of 'nomvar' key

    holacar             .true. :  conversion from hollerith to character
                        .false.:  conversion from character to hollerith

    htype               hollerith value of 'typvar' key

    ig1                 grid descriptor 1 (0 to 2047)

    ig2                 grid descriptor 2 (0 to 2047)

    ig3                 grid descriptor 3 (0 to 65535)

    ig4                 grid descriptor 4 (0 to 65535)

    ier                 return value of some functions.
```

```
                    ier >= 0  no error, everything ok
                    ier <  0  error code
                    NOTE:    consult 'fsterr' to identify the type of error

infon               length of the 'liste' array

ip1                 descriptor 1 (0 to 32767)

ip2                 descriptor 2 (0 to 32767)

ip3                 descriptor 3 (0 to 4095)

iun                 reference number of the file

key                 key used to locate a record in a standard file.
                    This key is obtained by:
                    key = FSTINF(...)
                    key = FSTLIR(...)
                    key = FSTSEL(...)
                    ier = FSTINL(...,LISTE,INFON,...)

                    NOTE:  The number that appears in the listing from
                           'FSTVOI' is not a key and MUST NOT BE USED.

liste               list of keys returned by FSTINL

lng                 length of a record (in machine words)

nbits               number of bits used by each data item.

ni                  1st dimension of a field (max 32767)

nj                  2nd dimension of a field (max 32767)

nk                  3rd dimension of a field (max 4095)

nmax                maximal dimension of 'liste' in a FSTINL call

nomvar              name of field (2 characters)

npak                compaction ratio
                    if npak = 0 or 1: nbits = bitmot (no compaction)
                    if npak > 1: nbits = bitmot/npak
                    if  npak  <  0:  number of bits kept per item
                                     (nbits = -npak)

npas                time step number

rewrit              .true.  :  replaces a record that has the same search
                               attributes (except for date) before write
                    .false.  : appends a record to the file without erasing
                               anything

swa                 address of the beginning of a record

typvar              field type (1 character)

ubc                 number of unused bits in the last word.

work                work array of dimension
                    (120+ni*nj*max(1,nk)*nbits+bitmot-1)/bitmot
                    where bitmot = number of bits per machine word
                          nbits = number of bits kept per item
                          if npak = 0 ou 1, dimension = ni*nj*max(1,nk)
```

```
                         Nature of arguments


    FORTRAN

    INTEGER DATEO, DATEV, DATYP, DEET, DLTF, EXTRA1, EXTRA2, EXTRA3, IG1,
            IG2, IG3, IG4, INFON, IP1, IP2, IP3, IUN, KEY, LNG, NBITS,
            NI,  NJ, NK, NMAX, NPAK, NPAS, SWA, UBC, LISTE(), HETIKET(2),
            HNOM, HTYPE, HGRTYP

    CHARACTER *8 ETIKET, CETIKET
    CHARACTER *2 NOMVAR, CNOM
    CHARACTER *1 GRTYP, CGRTYP, CTYPE, TYPVAR
    LOGICAL HOLACAR, REWRIT


    C

    int dateo,  datev,  datyp, deet, dltf, extra1, extra2,  extra3,  ig1,
        ig2,  ig3, ig4, infon, ip1, ip2, ip3, iun, key, lng,  nbits,  ni,
        nj,  nk,  nmax, npak, npas, swa, ubc, liste[], hetiket[2],  hnom,
        htype, hgrtyp

    char etiket[9], cetiket[9], nomvar[3], cnom[3], grtyp[2],  cgrtyp[2],
         ctype[2], typvar[2]

    int holacar, rewrit

NOTE
    See also fsterr(3), fstd89(3)
```

FSTARG2(3) STANDARD FILE SOFTWARE 1989 FSTARG2(3)

NAME

FSTARG2 - Definition of arguments for standard file functions

DESCRIPTION

FSTARG2 - lists, by alphabetical order, the definition of all arguments used in standard file functions used for opening, closing and marking of standard files.

Alphabetical list of arguments

NAME	DEFINITION
ier	return value ier >= 0 : everything ok ier < 0 : error
iun	reference number of the file
niveau	level of a logical end of sequential file mark (0 to 14)
nombre	number of active records in a standard file
options	value of options for fstouv, fstvoi possible values are: 'RND' : direct access file 'SEQ' : sequential file 'SEQ/FTN' : FORTRAN sequential file

Nature of arguments

FORTRAN

```
INTEGER IUN, NIVEAU, NOMBRE, IER
CHARACTER*(8) OPTIONS
```

C

```
int iun, niveau, nombre, ier
char options[9]
```

NOTE

See also fsterr(3), fstd89(3)

FSTARG3(3) STANDARD FILE SOFTWARE 1989 FSTARG3(3)

NAME

FSTARG3 - Definition of arguments for standard file functions

DESCRIPTION

FSTARG3 - presents, by alphabetical order, the list and definition of all function arguments controlling the options of standard file functions.

Alphabetical list of arguments

```
NAME                DEFINITION

entier             value to give to an integer type option

getset             when set to .true., initializes partial search masks or
                   the value of an option. When set to .false., returns
                   the current partial search mask or the value of an
                   option.

ier                return value
                   ier >= 0  : everything ok
                   ier <  0  : error

iun                reference number of the file

key                key pointing to a record

metik              partial mask of 8 characters corresponding to the
                   'etiket' search key

mip1               partial mask of 15 bits corresponding to the 'ip1'
                   search key

mip2               partial mask of 15 bits corresponding to the 'ip2'
                   search key

mip3               partial mask of 12 bits corresponding to the 'ip3'
                   search key

nrec               number of records by which one wants to go forward or
                   backward in a sequential file

option             name of the option to which one wants to set a value

string             value to give to an option of character type

reel               value to give to an option of real type

yesno              value to give to an option of logical type
```

Nature of arguments

FORTRAN

```
INTEGER ENTIER, NREC, IUN, IER, MIP1, MIP2, MIP3, KEY
CHARACTER *6 OPTION, STRING
CHARACTER *8 METIK
REAL REEL
LOGICAL YESNO, GETSET
```

```
    C

    int entier, nrec, iun, ier, mip1, mip2, mip3, key
    char option[7], metik[9], string[7]
    int yesno, getset
    float reel

NOTE
    See also fsterr(3), fstd89(3)
```

FSTECR(3) STANDARD FILE SOFTWARE 1989 FSTECR(3)

NAME

FSTECR - Writes a record in a standard file.

USAGE

```
    ier = FSTECR(buffer, WORK, npak, iun, dateo, deet, npas, ni, nj,
                 nk, ip1, ip2, ip3, typvar, nomvar, etiket, grtyp,
                 ig1, ig2, ig3, ig4, datyp, rewrit)

    ier = c_fstecr(buffer, work, napk,iun, dateo, deet, naps, ni, nj,
                   nk, ip1, ip2, ip3, typvar, nomvar, etiket, grtyp,
                   ig1, ig2, ig3, ig4, datyp, rewrit)
```

For the description of arguments, consult the man page about 'fstarg1'

An argument in CAPITALS is an OUTPUT argument, an argument in lower case is an input argument. In C, one has to make sure that an address is provided for the output arguments.

DESCRIPTION

FSTECR - writes on a standard file a field of type: real, integer, character or binary. If 'rewrit' has the value .true., the first record in the file having the same values of IP1,IP2, IP3, NOMVAR, TYPVAR and ETIKET as the field being written will be erased.

The field type is given by DATYP:

```
                    0 = binary
                    1 = real
                    2 = unsigned integer
                    3 = character
                    4 = signed integer
```

AUTHOR

Michel Valin - RPN - April 1989

NOTE

See also fstarg1(3), fstd89(3), fsterr(3)

```
FSTEFF(3)                    STANDARD FILE SOFTWARE 1989                    FSTEFF(3)


NAME
    FSTEFF  -  Erases a record in a direct access standard file.

USAGE
    ier = FSTEFF(key)

    ier = c_fsteff(key)


    For the description of arguments, consult the man page about 'fstarg2'



DESCRIPTION
    FSTEFF  - erases a record in a random access standard file. 'key' is set by a previous call to
FSTINF, FSTINL, FSTLIR, FSTSUI, FSTLIS.

    FSTEFF cannot be used in a sequential standard file.

AUTHOR
    Michel Valin - RPN - May 1989

NOTE
    See also fstinf(3),  fstinl(3),  fstlir(3),  fstsui(3),  fstlis(3), fstarg1(3), fstd89(3),
fsterr(3)
```

```
FSTEOF(3)                   STANDARD FILE SOFTWARE 1989                   FSTEOF(3)



NAME
    FSTEOF - returns the level of a logical end of file mark

USAGE

    ieof = FSTEOF(iun)

    ieof = c_fsteof(iun)


    For the description of arguments, consult the man page about 'fstarg2'


DESCRIPTION
    FSTEOF  - returns the level of a logical end of file mark reached in a sequential standard
file. The mark is written by the 'fstweo' function and has a value ranging from 1 to 14.

AUTHOR
    Michel Valin - RPN - July 1989

NOTE
    See also fstweo(3), fstarg2(3), fsterr(3), fstd89(3)
```

```
FSTERR(3)                    STANDARD FILE SOFTWARE 1989                    FSTERR(3)



NAME
    FSTERR - List of error codes returned by standard file functions

DESCRIPTION
    FSTERR  -  presents the list of error codes returned by the 1989 release of standard file
software accompanied by a brief description of these codes

                              Error codes


    CODE               DESCRIPTION

    -11      (ERRFII)   invalid file index
    -12      (ERRFNE)   file non existent
    -13      (ERRPTB)   page number too big
    -14      (ERRPTF)   page table full
    -15      (ERRFNO)   file not open
    -16      (ERRFNR)   file not an RPN random standard file
    -17      (ERRFSQ)   sequential file
    -18      (ERRFOP)   file already open
    -19      (ERRFTO)   too many files open
    -20      (ERRFDI)   odd number of entries in the directory
    -21      (ERRFNE)   damaged file
    -22      (ERRIVM)   invalid mode - option not SEQ or RND
    -23      (ERRTOV)   page table capacity exceeded
    -24      (ERRIVP)   invalid parameter (fstinf-fstecr-fstmsq)
    -25      (ERRIO )   input/output error
    -26      (ERREOF)   end of file
    -27      (ERRRNE)   record non existent
    -28      (ERRNVR)   no valid record available
    -29      (ERRKNV)   invalid 'key' (user trafic)
    -30      (ERRMDT)   bad 'datyp' in common fstd88 (fstlir)
    -31      (ERRNUI)   invalid unit number 'iun'  (fstfrm)
    -32      (ERRIPQ)   invalid parameter (qqqfnom-fstvoi)
    -33      (ERRNPW)   no write permission
    -34      (ERRDPL)   directory full (random standard file) (fstecr)
    -35      (ERRBNI)   base (filei) not initialized  (qdropn)
    -36      (ERRONV)   invalid option MSGLVL   TOLRNC
    -37      (ERRINV)   invalid option level in (fstopc-fstopl)
    -38      (ERRUNW)   bad unit number (<1 ou >99)
    -39      (ERRNRV)   no valid record - sequential file
    -40      (ERRDLN)   negative list dimension (fstinl)
    -41      (ERRFNS)   file not sequential (fstweo)
    -42      (ERRMRL)   bad record read (fstskp)

NOTE
    See also fstd89(3)
```

```
FSTFRM(3)                    STANDARD FILE SOFTWARE 1989                 FSTFRM(3)



NAME
    FSTFRM - closes a standard file.

USAGE

    ier = FSTFRM(iun)

    ier = c_fstfrm(iun)


    For the description of arguments, consult the man page about 'fstarg2'

DESCRIPTION
    FSTFRM - closes a standard file

AUTHOR
    P. Sarrazin - RPN - March 1989


NOTE
    See also fstd89(3), fsterr(3), fstarg2(3)
```

```
FSTINF(3)                    STANDARD FILE SOFTWARE 1989                    FSTINF(3)
```

NAME

FSTINF - Finds a record in a standard file.

USAGE

```
    key = FSTINF(iun, NI, NJ, NK, datev, etiket, ip1, ip2, ip3, typvar,
                 nomvar)

    key = c_fstinf(iun, &ni, &nj, &nk, datev, etiket, ip1, ip2, ip3,
                   typvar, nomvar)
```

For the description of arguments, consult the man page about 'fstarg1'

An argument in CAPITALS is an OUTPUT argument, an argument in lower case is an input argument. In C, one has to make sure that an address is provided for the output arguments.

DESCRIPTION

FSTINF - looks for the first record meeting search criteria specified by the arguments. An attribute to ignore is indicated by a value of -1 for the integer type arguments 'datev', 'ip1', 'ip2' and 'ip3'. For the character type arguments 'etiket', 'nomvar' and 'typvar', a single blank (' ') should be used.

AUTHOR

Michel Valin - RPN - February 1989

NOTE

See also fstsui(3), fstinl(3), fstsel(3), fstarg1(3), fstd89(3), fsterr(3)

```
FSTINL(3)                    STANDARD FILE SOFTWARE 1989                   FSTINL(3)
```

NAME

FSTINL - Finds all records satisfying selection criteria.

USAGE

```
    ier = FSTINL(iun, NI, NJ, NK, datev, etiket, ip1, ip2, ip3, typvar,
                 nomvar, LISTE, INFON, nmax)

    ier = c_fstinl(iun, &ni, &nj, &nk, datev, etiket, ip1, ip2, ip3,
                   typvar, nomvar, liste, &infon, nmax)
```

For the description of arguments, consult the man page about 'fstarg1'

An argument in CAPITALS is an OUTPUT argument, an argument in lower case is an input argument. In C, one has to make sure that an address is provided for the output arguments.

DESCRIPTION

FSTINL - looks for all the records satisfying the selection criteria specified. The function returns the record keys in the 'liste' table. An attribute to ignore is indicated by a value of -1 for the integer type arguments datev, ip1, ip2 and ip3. For the character type arguments 'etiket', 'nomvar' and 'typvar', a single blank (' ') should be used. If more than 'nmax' records meet selection criteria, only the 'nmax' first are returned and 'infon' takes the value 'nmax'.

In a sequential file, search starts from the current position. In a random file, search starts from the 1st record.

AUTHOR

Michel Valin - RPN - June 1989

NOTE

See also fstinf(3), fstsui(3), fstsel(3), fstarg1(3), fstd89(3), fsterr(3)

FSTLIR(3) STANDARD FILE SOFTWARE 1989 FSTLIR(3)

NAME

FSTLIR - Looks and reads a record in a standard file.

USAGE

```
    key = FSTLIR(BUFFER, iun, NI, NJ, NK, datev, etiket, ip1, ip2, ip3,
                 typvar, nomvar)

    key = c_fstlir(buffer, iun, &ni, &nj, &nk, datev, etiket, ip1, ip2,
                   ip3, typvar, nomvar)
```

For the description of arguments, consult the man page about 'fstarg1'

An argument in CAPITALS is an OUTPUT argument, an argument in lower case is an input argument. In C, one has to make sure that an address is provided for the output arguments.

DESCRIPTION

FSTLIR - Locates and reads the first record satifying the search criteria specified by the arguments 'date', 'etiket', 'ip1', 'ip2', 'ip3', 'typvar' and 'nomvar'. An attribute to ignore is indicated by a value of -1 for the integer type arguments 'datev', 'ip1', 'ip2' and 'ip3'. For the character type arguments 'etiket', 'nomvar' and 'typvar', a single blank (' ') should be used.

AUTHOR

Michel Valin - RPN - February 1989

NOTE

See also fstlis(3), fstluk(3), fstsel(3), fstarg1(3), fstd89(3), fsterr(3)

FSTLIS(3)   STANDARD FILE SOFTWARE 1989   FSTLIS(3)

NAME

FSTLIS - Reads the next record meeting selection criteria.

USAGE

```
    key = FSTLIS(BUFFER, iun, NI, NJ, NK)

    key = c_fstlis(buffer, iun, &ni, &nj, &nk)
```

For the description of arguments, consult the man page about 'fstarg1'

An argument in CAPITALS is an OUTPUT argument, an argument in lower case is an input argument. In C, one has to make sure that an address is provided for the output arguments.

DESCRIPTION

FSTLIS - locates and reads the next record meeting the selection criteria set by a previous call to FSTLIR, FSTINF and FSTSEL.

AUTHOR

Michel Valin - RPN - March 1989

NOTE

See also fstlir(3), fstluk(3), fstsel(3), fstarg1(3), fstd89(3), fsterr(3)

```
FSTLUK(3)                        STANDARD FILE SOFTWARE 1989                     FSTLUK(3)



NAME
    FSTLUK - Reads a record in a standard file.

USAGE

    key = FSTLUK(BUFFER, key, NI, NJ, NK)

    key = c_fstluk(buffer, key, &ni, &nj, &nk)


    For the description of arguments, consult the man page about 'fstarg1'

    An argument in CAPITALS is an OUTPUT argument, an argument in lower case is an input argument.
In C, one has to make sure that an address is provided for the output arguments.



DESCRIPTION
    FSTLUK - reads a record from which the key ('key') has been obtained through a previous call
to FSTINF, FSTSUI, FSTINL, FSTLIR or FSTLIS.

AUTHOR
    Michel Valin - RPN - March 1989

NOTE
    See also fstinf(3),  fstsui(3),  fstinl(3),  fstlir(3),  fstlis(3),
fstarg1(3), fstd89(3), fsterr(3)
```

```
FSTNBR(3)                      STANDARD FILE SOFTWARE 1989                    FSTNBR(3)



NAME
    FSTNBR - Returns the number of active records in a standard file.

USAGE

    nombre = FSTNBR(iun)

    nombre = c_fstnbr(iun)


    For the description of arguments, consult the man page about 'fstarg2'


DESCRIPTION
    FSTNBR - returns the number of active records contained in a random standard file. It is an
error to use 'fstnbr' for a sequential file.

AUTHOR
    Michel Valin - RPN - January 1989

NOTE
    See also fstarg2(3), fstd89(3), fsterr(3)
```

```
FSTOUV(3)                    STANDARD FILE SOFTWARE 1989                    FSTOUV(3)



NAME
    FSTOUV - opens a standard file.

USAGE

    nombre = FSTOUV(iun, options)

    nombre = c_fstouv(iun, options)


    For the description of arguments, consult the man page about 'fstarg2'



DESCRIPTION
    FSTOUV  - opens a standard file and returns the number of active records in the directory.

AUTHOR
    Michel Valin - RPN - January 1989

NOTE
    See also fstarg2(3), fstd89(3), fsterr(3)
```

```
FSTPRM(3)                      STANDARD FILE SOFTWARE 1989                     FSTPRM(3)
```

NAME

FSTPRM - Returns all the information associated to a standard file record.

USAGE

```
    ier = FSTPRM(key, DATEO, DEET, NPAS, NI, NJ, NK, NBITS, DATYP, IP1,
                 IP2, IP3, TYPVAR, NOMVAR, ETIKET, GRTYP, IG1, IG2, IG3,
                 IG4, SWA, LNG, DLTF, UBC, EXTRA1, EXTRA2, EXTRA3)

    ier = c_fstprm(key, &dateo, &deet, &npas, &ni, &nj, &nk, &nbits,
                   &datyp, &ip1, &ip2, &ip3, typvar, nomvar, etiket,
                   grtyp, &ig1, &ig2, &ig3, &ig4, &swa, &lng, &dltf,
                   &ubc, &extra1, &extra2, &extra3)
```

For the description of arguments, consult the man page about 'fstarg1'

An argument in CAPITALS is an OUTPUT argument, an argument in lower case is an input argument. In C, one has to make sure that an address is provided for the output arguments.

DESCRIPTION

FSTPRM   -   returns all the information associated to the record which is pointed to by 'key'

AUTHOR

Michel Valin - RPN - July 1989

NOTE

See also fstarg1(3), fstd89(3), fsterr(3)

```
FSTRWD(3)                        STANDARD FILE SOFTWARE 1989                     FSTRWD(3)
```

NAME

FSTRWD - Rewinds a sequential standard file.

USAGE

```
    ier = FSTRWD(iun)

    ier = c_fstrwd(iun)
```

For the description of arguments, consult the man page about 'fstarg2'.

DESCRIPTION

FSTRWD - rewinds a standard file of type 'SEQ' or 'SEQ/FTN'. This function has no effect on a random (RND) standard file.

AUTHOR

Michel Valin - RPN - January 1989

NOTE

See also fstarg3(3), fsterr(3), fstd89(3), fstpos(3), fstskp(3)

FSTSUI(3)   STANDARD FILE SOFTWARE 1989   FSTSUI(3)

NAME

FSTSUI - Locates in a standard file, the next record matching selection criteria.

USAGE

```
    key = FSTSUI(iun, NI, NJ, NK)

    key = c_fstsui(iun, &ni, &nj, &nk)
```

For the description of arguments, consult the man page about 'fstarg1'

An argument in CAPITALS is an OUTPUT argument, an argument in lower case is an input argument. In C, one has to make sure that an address is provided for the output arguments.

DESCRIPTION

FSTSUI - finds the next record satisfying search criteria used in a previous call to FSTINF, FSTLIR or FSTSEL.

AUTHOR

Michel Valin - RPN - February 1989

NOTE

See also fstinf(3), fstinl(3), fstlir(3), fstsel(3), fstarg1(3),fstd89(3), fsterr(3)

```
FSTVOI(3)                    STANDARD FILE SOFTWARE 1989            FSTVOI(3)



NAME
    FSTVOI - prints the record description tags of a standard file.

USAGE

    ier = FSTVOI(iun, options)

    ier = c_fstvoi(iun, options)


    For the description of arguments, consult the man page about 'fstarg2'

DESCRIPTION
    FSTVOI - prints the record description tags of a standard file. In a sequential file, 'fstvoi'
prints the record description tags of the file from the current position up to the next end of
file marker.

AUTHOR
    Michel Valin - RPN - February 1989

NOTE
    See also fstarg2(3), fsterr(3), fstd89(3)
```