CERN Program Library Long Writeups I302

FFREAD

Format Free Input Processing

Version 3.13

Application Software and Databases Group

Computing and Networks Division

CERN Geneva, Switzerland

CERN Program Library Office
CERN-CN Division
CH-1211 Geneva 23
Switzerland
Tel. +41 22 767 4951
Fax. +41 22 767 7155
Bitnet: CERNLIB@CERNVM
DECnet: VXCERN::CERNLIB (node 22.190)
Internet: CERNLIB@CERNVM.CERN.CH

Contact Person: CERN Program Library (CERNLIB@CERN.CH)
Technical Realization: Michel Goossens /CN (Michel.GOOSSENS@CERN.CH)

Edition – February 1994

Preliminary remarks

Throughout this manual, commands to be **entered** are `underlined`

This document has been produced using LaTeX [1] with the `cernman` style option, developed at CERN. A compressed PostScript file `ffread.ps.gz`, containing a complete printable version of this manual, can be obtained from any CERN machine by anonymous ftp as follows (commands to be typed by the user are underlined)[1]:

```
ftp asis01.cern.ch
Trying 128.141.201.136...
Connected to asis01.cern.ch.
220 asis01 FTP server (Version 6.10 ...) ready.
Name (asis01:username): anonymous
Password: your_mailaddress
230 Guest login ok, access restrictions apply.
ftp¿ cd cernlib/doc/ps.dir
ftp¿ binary
ftp¿ get ffread.ps.gz    ! one page per physical page
ftp¿ quit
```

[1] If your site does not carry the gnu `gzip` utility you can get the uncompressed file by dropping the `.gz` suffix from the `get` command, and also skipping the `binary` specification below.

Table of Contents

Chapter 1: Introduction

FFREAD is a set of FORTRAN-77 subroutines that allow input of variables to a program in a format-free way, much like the NAMELIST statement of old. Unlike the latter, it is portable across a range of computers and has some additional features, like the capability of reading input from files and, on demand, call- ing a user action routine.

1.1 Principles of Operation

To use FFREAD, the user first identifies a variable or array he wishes to set up by a key. These variables and arrays should be in COMMON; the use of local variables can lead to problems with optimizing compilers. After calling an initialization subroutine, he then defines each key he wants to use to the system, giving its location, type (real, integer, or logical), and length (for arrays). When he has defined all keys, he can at any time during his program call the main subroutine of FFREAD, which will read in the data cards and modify the variables in memory accordingly.

Chapter 2: Syntax of the data cards

Data cards are one of four types:

1. a totally blank card is just skipped
2. a comment card has as first non-blank character the single letter 'C'; the further contents of the card are ignored
3. the first non-blank item is a user-defined key or one of the system keys (see below)
4. the remaining cards are considered to be continuation cards, i.e. their information applies to the last key identified. If the last key is not defined (because this is the first non-blank, non-comment card read or because the last key was a system key), an error message is issued.

In all cases but for the exceptions connected with hollerith items, upper and lower case characters are considered identical. Once a key has been identified, the data card is scanned for items of infor- mation or, in the case of a system key, some special action is taken.

An item of information consists of three parts, two of which are optional, and it must be wholly contained on one card:

1. an indication of where the information is to be stored in an array (optional); this is indicated by any positive integer followed by the letter '='. If this part is missing, it defaults to 1 if this is the first item for a key or to the updated value from the previous item, i.e. *position of previous item + repeat count of previous item*.
2. a repeat count (optional); this is indicated by any positive integer followed by the letter '*'. If this is missing, it defaults to 1.
3. a value (required); this takes one of four forms:
   (a) a standard FORTRAN signed integer
   (b) a standard FORTRAN signed floating point number in either fixed-point or 'E' notation
   (c) a hollerith text, which is delimited by a special character (default is the apostrophe '). Such a value may be continued on the following card(s) and is copied literally, i.e. without case conversion.
   (d) a logical constant, where the following values are defined:
      - 'T', 'ON' , or 'TRUE' for the FORTRAN logical value .TRUE.
      - 'F', 'OFF', 'FALS', or 'FALSE' for the FORTRAN logical value .FALSE.

If a key is identified and the rest of the card is blank, this defaults to '1', '1.', or '.TRUE.', depending on the type of key. If no recognized type was given for the key, an integer 1 is stored.

If more than one part is present, they must be in the above order and adjacent, i.e. with no blanks between them.

The value is then stored, starting at the indicated position in the array and for the indicated number of times given by the repeat count.

If anything on a data card doesn't fit the above description, it is considered to be an in-line comment and is skipped.
For all numerical values, the space used per value is one FORTRAN word. For a hollerith value, the space consumed depends on the number of characters in the string and the number of characters which can be stored in one word on the particular computer. However, one word is the minimum and any remaining space in the last (or only) word is blank-filled. As the space required by hollerith values cannot be predicted before the end of the string is reached (which might be on a continuation card), a repeat count is not allowed for such values.
The value actually stored depends on the data type for the key given by the user when he defined it:

- If the key is of type 'REAL' and the item found is an integer, the integer is converted to the corresponding floating point value before storing.
- If the key is of type 'INTEGER' and the item found is a floating-point number, the value stored is the result of the FORTRAN function IFIX.

In either case, if a logical constant is given, the corresponding logical value is stored (i.e. in this case no type checking is performed).

- If the key is of type 'LOGICAL', any value but a logical constant results in an error.

In all cases, bounds checking is performed, i.e. the user cannot store values into locations that are not part of an array (as declared at the time of key definition).
If any error occurs during scanning of an item, no action is performed for this card (even if correct items were found before) and the scan is terminated. Continuation cards are however interpreted for this key; the current position may not be what the user expected.

2.1 Examples of items

The following are valid items; their interpretation is given in parantheses.

```
    1  (store one copy of an integer or floating value '1' at the next location)
  1*1  (equivalent to the above)
1=1*1  (equivalent to the above except force storage at first element of array)
'abcd' (store the hollerith string 'abcd')
  1.2 (1.2) -.1 (-0.1) 1E0 1e0 (1.) 13e-4 (1.3E-5) -1.05E-7 (-1.05E-7)
```

The following are all strange, but valid representations of the number 0:

```
- . e E
-. .e .E -e -E e- E-
-.e -.E .e- .E-
-.e- -.E-
.E3 -.E-10
```

The following are not valid; the reason is given in parantheses.

```
-1*1 (repeat count not positive)
-17=5 (location indicator not positive)
3*'abcd' (repeat count not allowed on hollerith item)
```

Note: If you write '- 1.3 E -3', you won't get the number -1.3E3, but the four numbers 0., 1.3, 0., and -3.

Chapter 3: How keys are identified

3.1 Key length

By default, keys have a significant length of four characters. This can be changed (see the description of FFSET below) at initialization. If the first non-blank text on a data card compares sucessfully to a user or system key when truncated to the significant length, the key is recognized and the rest of the text is considered to be comment.

3.2 System keys

Apart from the single letter 'C' which marks a comment card and could be considered to be a system key, there are eight system keys. They are listed below with their arguments (if any) and their actions.

`LIST`	(no argument) Turn on listing of data cards as recognized by FFREAD; also output error messages.
`NOLIst`	(no argument) Turn off listing; don't print error messages.
`READ`	(one integer argument) Read input from FORTRAN logical unit as given by the integer. The current input unit is pushed on a stack. Reading stops at end-of-file or when the data cards STOP or END are encountered.
`WRITe`	(one integer argument) Write listing and error messages to FORTRAN logical unit as given by the integer (if listing is enabled).
`STOP`	(no argument) Unconditionally stop reading data cards and return to the user.
`END`	(no argument) If reading a file via READ, return to the previous unit (pop the stack). Otherwise, it has the same effect as STOP.
`HOLL`	(one character from the set ('=$()"/) Change the character delimiting a hollerith value.
`KEYS`	(no argument) List all system and user keys when the STOP condition is reached. This is done independent of the LIST flag setting.

For these keys, any abbreviation between their shortest form (indicated by upper case in the list above) and the number of significant characters as defined above is recognized.

3.3 User keys

User keys are truncated to the number of significant characters; abbreviations are not allowed. Before comparison with the list of defined keys, a potential key is converted to uppercase. If it is recognized, it is printed on the lis- ting in upper case.
Note: Although you can define a key to be the same as a system key, this key will not be recognized, as the list of system keys is checked first.

3.4 '*' keys

If a valid key (user or system) is preceded with the character '*', this indicates the user wants an action routine to be called for this key. This is done after all values for this key (including those on any continuation cards) have been stored and it is supplied (as a hollerith value) the key which triggered the call. The action routine has the name FFUSER; a dummy routine is supplied with the library.

Chapter 4: Description of FFREAD subroutines

4.1 Initialization

```
CALL FFINIT (NW)
```

This subroutine initializes the FFREAD package.

If any keys had been previously specified, they are erased with all associated information. The values for the input and output logical units are reset to their default values, which direct them to the terminal or command file/batch log. The significant key size is reset to its default value of 4.

The integer argument NW specifies the number of words the user has allocated to the COMMON /CFREAD/, in which FFREAD keeps all its information. If NW = 0, the default size is large enough to hold 50 keys of the default length (on all machines). If a larger number of keys is required, the user must do the following:

- Define CFREAD to be of appropriate size by including a statement like

  ```
  COMMON /CFREAD/ SPACE(1000)
  ```

  in his programme.

- Call FFINIT with the proper size, i.e. for the above example:

  ```
  CALL FFINIT (1000)
  ```

  The calculation of the required size for a given number of keys is as follows:

- divide the required key size by the number of characters per word (as returned by FFGET with option 'NCHW') and round, if not integer, to the next higher integer
- add to this a fixed overhead of three words per key
- multiply by the number of keys needed
- add a fixed overhead of 41 words for FFREAD's internal management.

4.2 Set optional values

```
CALL FFSET (CHOPT, IVALUE)
```

This will set some optional values which are identified by the character argu- ment CHOPT to the integer value given by IVALUE, if that is valid. Options are:

`LINP` Set input logical unit to given value. IVALUE must be a valid logical unit number.

`LOUT` Set output logical unit to given value. IVALUE must be a valid logical unit number.

`SIZE` Set the significant size of a key (in characters). IVALUE must be in the range from 4 to 32.

If called with option 'SIZE', `FFSET` must be called after `FFINIT` and before the first call to `FFKEY` is made; otherwise, an error message is printed. The input and output logical units may be changed at any time.

4.3 Define a key

```
CALL FFKEY (KEY, ADRESS, LENGTH, CHTYPE)
```

This subroutine actually defines a key to FFREAD. Its arguments have the following meaning:

`KEY` character string containing the key to use (character)

`ADRESS` variable/array to store values to (anything but character)

`LENGTH` number of values associated with this key (integer)

`CHTYPE` character string specifying the type of variable/array For the values 'INTE', 'REAL', or 'LOGI' (only four characters are significant), type checking as described above will be performed. If this argument has any other value, no type checking will be done.

KEY is truncated to the significant key length and converted to upper case.
If one of the following conditions is met,the key is not defined and an error message is issued:

- no space is available to define a new key
- the key is identical to a previously defined one
- LENGTH is not positive.

4.4 Reading in the data cards

```
CALL FFGO
```

This subroutine actually reads in the data cards and modifies the memory locations associated with the keys given.

4.5 The user action routine

```
CALL FFUSER (KEY)
```

This routine is called whenever a key on input is preceded by '*', after all memory locations set up by the associated data card(s) have been written.
Its integer argument KEY is dimensioned to as many words as necessary to hold the key which caused the call to be stored in hollerith format to its significant length (blank padded, if necessary).

4.6 Get some internal and machine parameters

```
CALL FFGET (CHOPT, IVALUE)
```

This subroutine returns some internal parameters of FFREAD as selected by the character variable CHOPT. Valid values and their meaning are:

`LINP` return LUN for input

`LOUT` return LUN for output

`NBIT` number of bits per word

`NCHW` number of characters per word

`NBCH` number of bits per character

`NCHK` number of characters per key

NBIT, **NCHW** and **NBCH** are machine constants.
The requested parameter is retuned in the integer variable IVALUE.

4.7 An example of use

Here is an example of the use of FFREAD in the real world. First, a slightly reduced version of the GEANT routine GFFGO, which sets up the standard GEANT keys and then reads in the data cards. Note the usage of key types and array lengths and, as a convention, the use of the keyword 'MIXED'.

Example of using FFREAD

```
      SUBROUTINE GFFGO
C
      COMMON/GCCUTS/CUTGAM,CUTELE,CUTNEU,CUTHAD,CUTMUO,BCUTE,BCUTM
     +            ,DCUTE ,DCUTM ,PPCUTM,TOFMAX,GCUTS(5)
C
      COMMON/GCFLAG/IDEBUG,IDEMIN,IDEMAX,ITEST,IDRUN,IDEVT,IEORUN
     +        ,IEOTRI,IEVENT,ISWIT(10),IFINIT(20),NEVENT,NRNDM(2)
C
      COMMON/GCKINE/IKINE,PKINE(10),ITRA,ISTAK,IVERT,IPART,ITRTYP
     +      ,NAPART(5),AMASS,CHARGE,TLIFE,VERT(3),PVERT(4),IPAOLD
C
      COMMON/GCLIST/NHSTA,NGET ,NSAVE,NSETS,NPRIN,NGEOM,NVIEW,NPLOT
     +       ,NSTAT,LHSTA(20),LGET (20),LSAVE(20),LSETS(20),LPRIN(20)
C
C           Define standard keys
C
      CALL FFKEY ('CUTS',CUTGAM,16,'REAL')
      CALL FFKEY ('DEBU',IDEMIN, 3,'INTEGER')
      CALL FFKEY ('GET ',LGET  ,20,'MIXED')
      CALL FFKEY ('HSTA',LHSTA ,20,'MIXED')
      CALL FFKEY ('KINE',IKINE ,11,'MIXED')
      CALL FFKEY ('PRIN',LPRIN ,20,'MIXED')
      CALL FFKEY ('RNDM',NRNDM , 2,'INTEGER')
      CALL FFKEY ('RUN ',IDRUN , 2,'INTEGER')
      CALL FFKEY ('RUNG',IDRUN , 2,'INTEGER')
      CALL FFKEY ('SAVE',LSAVE ,20,'MIXED')
      CALL FFKEY ('SETS',LSETS ,20,'MIXED')
      CALL FFKEY ('SWIT',ISWIT ,10,'INTEGER')
      CALL FFKEY ('TRIG',NEVENT, 1,'INTEGER')
C
C           Now read data cards
C
      CALL FFGO
C
  999 END
```

And here is an example input, taken from the GEANT test job GEXAM1:

Input file to previous example

```
LIST
TRIGGERS   10
DEBUG    1  1   1
SWIT     1
CUTS  0.0001    0.001
KINE 2=10. 1.
SETS   'CDET' 'HBAR'  'MUEC'
PRINT   'MATE' 'VOLU'  'TMED'
END
```

4.8 For compatibility with FFREAD 1.0 - do not use

`CALL` **FFREAD** `(NKEY, KEY, LOCVAR, LENVAR)`

This is the old-fashioned form of using FFREAD and is only supplied for backward compatibility.

If NKEY is set to 0, it is used to modify the values for the default input/output LUNs.

If NKEY is negative, it will return them.

If NKEY is positive, it specifies the number of keys in KEY. The keys will be set up (destroying any previously defined keys) and FFGO called to do the work.

The arguments and their meaning:

`NKEY` number of keys. Different actions depending on sign of NKEYS are described above.

`KEY` array containing user's keys, one per machine word, with four significant characters.

`LOCVAR` locations of variables/arrays to change as returned by LOCF dimensioned to NKEY).

`LENVAR` length of array at location given in LOCVAR (dimensioned to NKEY).

4.9 Internal and external routines used by FFREAD

Internally, FFREAD uses the subroutines FFCARD, FFFIND and FFUPCA.

The following external routines are used by FFREAD. They all come out of the CERN library KERNLIB:

`V304` IUCOMP

`N100` LOCF

`V301` UCOPY2

`M409` UCTOH/UHTOC

4.10 Installation of FFREAD from the PAM file

FFREAD contains two patches: FFCDES defines the sequences for COMMON blocks and PARAMETERs, FFREAD contains the actual code. A PATCHY flag is used to define each machine. Currently, the following are available:

`APOLLO` for APOLLO

`BESM6`	for ?
`CDC`	for CDC
`CRAY`	for CRAY
`IBM`	for IBM
`NORD`	for NORD50 and NORD500
`PDP10`	for PDP10
`UNIVAC`	for UNIVAC
`VAX`	for VAX

NORMAL can be used if your machine is not in the list and:

- is a 32 bit, eigth bits per character machine
- uses ASCII code for its characters.
- only uses standard FORTRAN logical unit numbers 0 to 99.

The necessary cradle then looks like this:

Cradle to extract FFREAD code

```
+USE, FFCDES, FFREAD, ¡machine flag¿, T=EXE.
+PAM,T=CARDS.
+QUIT.
```

The PAM also contains a short test program with associated input. This programme and its data are extracted as follows (assuming you have given a file name to stream 23):

Extracting the test program

```
+ASM, 23.
+USE, FFTEST, T=EXE.
+PAM,T=CARDS.
+QUIT.
```

After having compiled and linked the programme (it contains a short user action routine), assign the data file produced by the above PATCHY run to a suitable unit number. Then run the program and give the data cards:

Data cards for test program

```
LIST
KEYS
READ ¡logical unit number of data file¿
```

and observe the result. For the IBM and the CDC, example jobs to perform the above are included in the patch TEST, the required version is selected as before.

Bibliography

[1] L. Lamport. *LaTeX A Document Preparation System*. Addison-Wesley, 1986.

Index