F, (file-pointer var-addr n --) "f comma"

Send the cell N to the specified file at its current position using buffered I/O mode, the buffer address being held in the VARIABLE or USER-variable VAR-ADDR .

The 1024-byte buffer is created by a previous call to OPENFV which also initializes the VAR-ADDR . Buffers allocated for purposes of writing should be closed with CLOSEFVWRITE .

See the chapter on JForth File I/O for more information on virtual file usage.

```
( store DATA in next buffer cell )
MYFILE @ MYBUFFER  DATA F,
```

Related Words: OPENFV CLOSEFVWRITE FFLUSH?

F@, (file-pointer -- cell) "f fetch comma"

Fetch the next sequential 32-bit-value from a file; the cell will be read from the current location for the file. F@, calls FREAD; the programmer should check FERROR after to insure the value returned is valid.

See chapter on File I/O.

Related Words: @ FREAD FERROR F@,?

F@,? (file-pointer -- cell) "f fetch comma question"

This word is functionally identical to F@, except it will print an error message and execute QUIT if an error occurs during the FREAD (calls FREAD?).

JForth provides for optional automatic cleanup of opened files if an error forces a QUIT. See MARKFCLOSE and UNMARKFCLOSE.

Related Words: F@,

FALSE (-- 0)

FALSE is a constant which returns the value of a logical false. Usually zero in most systems.

Related Words: TRUE 0= IF

FAST (--)

After FAST has been executed, the EMIT I/O will be in line-buffered mode, a highly-efficient alternative to the single-character I/O mode. This is the default mode of CONSOLE I/O as JForth is distributed.

In FAST mode, any characters EMITted are held in a buffer until either:

1) The buffer is full.

2) CR , CR? , or FLUSHEMIT is executed.

3) Input is requested by KEY , EXPECT .

FAST acts to install its own vectors in the EMIT vector.

Any characters EMITted in FAST mode will update OUT (via +OUT) to reflect the current cursor column number.

Related Words: SLOW <FASTEMIT> CONSOLE EMIT KEY FLUSHEMIT

FBLK (-- addr) "f b l k"

This is a USER-variable, used to hold the file-pointer for the ASCII file currently being INCLUDEd.

FBLK should not normally be altered by the programmer.

Related Words: INCLUDE

FCLOSE (file-pointer --) "f close"

FCLOSE will close the specified file. Files should be closed when you are through using them. When the user executes BYE, any open files will be closed.

See the chapter on File I/O.

```
MYFILE @ FCLOSE
```

Related Words: FOPEN $FOPEN

FCLOSEVAR (var-addr --)

If you use a variable to hold the address of a file pointer, this will close the file then clear the variable. It first checks to make sure the variable does not contain zero.

```
VARIABLE MY-FILEID
NEW FOPEN RAM:TESTFILE
MY-FILEID !  \ save pointer in variable
MY-FILEID FCLOSEVAR  \ now close it if open
MY-FILEID FCLOSEVAR  \ yes, this is safe
```

Related words: FCLOSE FREEVAR

FENCE (-- addr)

FENCE is a user-variable which may be set to an address below which FORGET will issue a warning.

FENCE may be set by the programmer, but is automatically set by the JForth System Manager to the current HERE in the following situations:

1) By FREEZE (executed by SAVE-FORTH).

2) If FORGET will lower the dictionary below the currently frozen FENCE.

```
( cautions if FORGETting below AARDVARK )
' AARDVARK FENCE !
```

Related Words: FORGET FREEZE SAVE-FORTH

FERROR (-- var-addr) "f error"

FERROR is a user variable which provides the programmer an alternate means (other than return codes) for checking for file operation errors.

Each of the 3 primary file system primitives, FREAD, FWRITE and FSEEK clear FERROR at their start, and adjust it upon completion; if used, it should be checked as soon as the call returns.

```
MYFILE @  PAD  256 FREAD
FERROR @ abort" Error in file read"
```

Related Words: FREAD FWRITE FSEEK

FFLUSH? (file-pointer var-addr --)

Flush the contents of the virtual buffer to disk. The variable should contain the address of a buffer which must have been opened by OPENFV .

See chapter on File I/O

Related Words: OPENFV CLOSEFVWRITE F,

FIG-NOT (n -- 1 | 0) "fig not"

If n=0 then leave 1 on stack otherwise leave a zero.

Standard: FIG

```
Related Words: NOT 0= IF
```

FILE? (<name> --) "file question"

FILE? is used to identify the file that a word was compiled from, and optionally allows the programmer to search that file for instances of the same text.

In order for FILE? to work for the words from a given file, that file had to have been compiled with FILEHEADERS enabled. This is the default state for JForth as distributed. When FILEHEADERS contains a non-zero value, special dictionary entries are created marking the beginning and end of the file. (4 colons are appended to the name, i.e. ::::MYFILE, to mark the beginning; the end is noted by 3 semi-colons...;;;).

Once the file has been identified and opened, its text is searched for instances of the word's name. If found, that line will be EMITted along with each following non-blank line. The search for another instance will resume when an empty line is encountered.

As the source is not supplied for words below TASK, this utility cannot be used for kernal primitive words, approx. the first 30K of the image. These filenames begin with J and end with either .I or .ASM . JCOMPILER.I and JKERNAL.ASM are examples of these files.

```
FILE? :STRUCT
```

```
Related Words: FILEHEADERS  EACH.FILE?
```

FILEHEADERS (-- addr)

This user-variable is examined by INCLUDE to see if the programmer desires file headers, those required for the operation of FILE? . You can save some space in the dictionary by turning FILEHEADERS OFF but then FILE? won't work.

FILEWORD (<filename> -- addr)

FILEWORD operates similarly to WORD, in that it parses the input stream, moving suitable text to HERE, however, it uses parsing rules that allow a filename with spaces to be passed to AmigaDOS.

If the first word that it sees starts with a double-quote "" then it will parse up to the next double-quote, skipping over any spaces.

```
: FOPEN  FILEWORD $FOPEN ;
FOPEN "df1:icons are silly"
```

The resultant string may be subsequently null-terminated and moved to the DOS0 buffer with: (-- addr) COUNT >DOS

```
Related Words: COUNT >DOS WORD HERE FOPEN $FOPEN
```

FILL (addr u byte --)

Fill U bytes of memory starting at ADDR with BYTE .

```
PAD 256 BL FILL   ( put 256 blanks at PAD )
```

```
Related Words: C! ERASE ! MEM! CMOVE
```

FIND ($addr -- $addr 0 | cfa -1 | cfa 1)

FIND searches the CONTEXT vocabulary for the string at $addr, and returns whether it was found, and if so, whether it is an IMMEDIATE word.

0 means the word was **not** found. The original name address will be returned.

If the word was found. A 1 or a -1 and the word's CFA is returned.

1 means the words is IMMEDIATE. INTERPRET uses this when compiling to decide whether to immediately execute the word it finds.

-1 means the word is **not** IMMEDIATE.

If the search is not successful, AND the user-variable SEARCH-CURRENT is non-zero, the search will also be performed in the CURRENT vocabulary.

```
: SHOWDEF  ( <name> -- , disassemble word if found )
    BL WORD  FIND 0=
    IF $TYPE ."  could not be found!" CR QUIT
    THEN DISM ;
SHOWDEF DUP
```

Related Words: WORDS-LIKE FILE? FORGET

FIND-DATA (addr count n --)

Beginning with ADDR , search for COUNT bytes, looking for the 32-bit-value N. Note that ADDR MUST be WORD-aligned, and that only such addresses are checked. The addresses containing matching cells are printed on the standard EMIT device.

Related Words: SCAN

FIND-WDATA (addr count w --)

This is functionally identical to FIND-DATA, except that it searches for WORD-sized data. See FIND-DATA.

FLD (-- addr)

User variable for controlling field length in numeric output.

FLUSHEMIT (--)

FLUSHEMIT is used in FAST mode to force the EMIT buffer to be EMITted even though it is not full, and an end-of-line character has not yet been received.

Related Words: FAST SLOW EMIT

FOPEN (<filename> -- file-pointer | 0) "f open"

Open a file by name. Return zero if file could not be opened. The programmer should ALWAYS check the returned value from FOPEN.

Please see the chapter on File I/O for details.

```
FOPEN RAM:MYFILE  ( open existing file )
DUP 0= ABORT" File could not be opened!"

NEW FOPEN RAM:HOTNEWS  ( open new file )
```

Related Words: FILEWORD DOS0 NEW HERE FREAD FWRITE 0FOPEN $FOPEN FCLOSE

FORGET (--)

FORGET deletes definitions from the dictionary . The specified definition and all definitions following up to the end of the dictionary are forgotten.

If you would like the contents of a file to automatically be forgotten when you REinclude the file, use ANEW which calls FORGET.

If you would like a word to be called if something is forgotten use IF.FORGOTTEN. This can be important if you are forgetting code that has allocated memory or opened files, etc. and want to clean up before losing your pointers.

If you need to redefine what FORGET does, define a word called [FORGET] that calls [FORGET] . This will get called by FORGET. See [FORGET].

```
: FOO ." Hello" cr ;
: GOO 23 + . ;
FORGET FOO
FOO ( won't find FOO or GOO )
```

Related Words: ANEW IF.FORGOTTEN [FORGET] FIND WORDS

FREAD (file-pointer addr max-to-read -- #bytes | -1)

READ will read from the specified file at its current location, placing the contents at RELADDR. FREAD will read until MAX-TO-READ bytes have been read, or end-of-file, whichever occurs first.

FREAD always returns a value. If the call was successful, it returns the number of bytes #BYTES that were read in If an error occurred, it returns -1.

An error is also indicated by a non-zero value in the user-variable FERROR, immediately following the call.

See chapter on File I/O.

```
MYFILE @ PAD 256 FREAD
PAD SWAP DUMP  ( dump stuff read )
```

Related Words: FOPEN 0FOPEN FSEEK FREAD?

FREAD? (file-pointer addr max-to-read -- #bytes)

FREAD? is functionally identical with FREAD with one exception; if an error occurs during the read operation, FREAD? will print "Error on file read!" and execute QUIT.

JForth provides for optional automatic cleanup of opened files if an error forces a QUIT. See MARKFCLOSE and UNMARKFCLOSE.

For further information, see FREAD.

```
MYFILE @  PAD  256  FREAD? .
```

Related Words: FREAD QUIT MARKFCLOSE UNMARKFCLOSE

FREEBLOCK (memblock --)

FREEBLOCK is used to return memory to the Amiga memory manager that had been previously allocated with ALLOCBLOCK. It performs housekeeping functions along with the JForth Memory Manager and should therefore be done ONCE AND ONLY ONCE for each previous ALLOCBLOCK call.

At BYE, The JForth Memory Manager will return all memory-blocks allocated with ALLOCBLOCK that had NOT been returned via FREEBLOCK. JForth also provides for automatic cleanup of allocated memory areas if an error forces a QUIT. See MARKFREEBLOCK and UNMARKFREEBLOCK.

See section on Memory Management.

Related Words: ALLOCBLOCK MARKFREEBLOCK UNMARKFREEBLOCK FREEVAR

FREEBLOCKS (memory-block --)

FREEBLOCKS accepts the address of a memory block previously allocated via ALLOCBLOCK, which contains a list of other memory blocks.

FREEBLOCKS will proceed through such a list, closing each memory block. The memory block used to store the list will not be freed.

```
Related Words: FREEBLOCK  @&FREEBLOCKS
```

FREEBYTE (memory-block -- contents-of-counter)

A 32-bit storage location is available for each memory block allocated; the contents of which are returned by FREEBYTE. This is used to keep track of data when the memory block is used as a stack or a virtual file buffer.

The FREEBYTE-counter is initialized to zero when the memory-block is allocated, and update automatically by PUSH POP +STACK -STACK FVREAD and FVWRITE.

See section on Memory Management for details.

```
     MYSTACK @  FREEBYTE  CELL/  ( #cells on MYSTACK )
Related Words: FREEBYTEA  PUSH  POP  +STACK  -STACK
```

FREEBYTEA (memory-block -- address-of-counter)

FREEBYTEA returns the address of the FREEBYTE-counter for a memory-block. This is particularly useful if the programmer is using the FREEBYTE-counter for custom purposes and wishes to update its value.

```
     0  MYMEMBLK @  FREEBYTEA  !
     ( now FREEBYTE will return 0 )
```

FREECELL (memory-block -- freecell)

This is functionally identical to FREEBYTE , except that it returns the offset of the next free CELL, based on the FREEBYTE-counter.

FREEVAR (var-addr --)

If you use a variable to hold the address of allocated memory, this will free the memory then clear the variable. It first checks to make sure the variable does not contain zero.

```
     VARIABLE BIG-BUFFER
     MEMF_CLEAR  2000 ALLOCBLOCK
     BIG-BUFFER !  \ save pointer in variable
     BIG-BUFFER FREEVAR  \ now free it if allocated
     BIG-BUFFER FREEVAR  \ yes, this is safe
Related words:  FREEBLOCK FCLOSEVAR
```

FREEZE (--)

FREEZE is used to snapshot important JForth system variables which, at COLD-start time, define:

1) The most recently defined word in the dictionary, and the resulting VOCABULARY structure.

2) The vectors for the following DEFERed system words:

```
     EMIT        KEY         ?TERMINAL   CR          QUIT
     INTERPRET   FIND        :CREATE     NUMBER      ID.
     SOURCE      BLOCK       'WORD       LONGCFA,    CFA,
     FLUSHEMIT   COLDEXEC    ABORT
```

3) All USER-variables defined below TASK.

Even though the dictionary may be extended, and the above vectors altered by the programmer, their contents at the last FREEZE will be restored by COLD.

FREEZE is called automatically by SAVE-FORTH or if FORGET lowers the dictionary below that saved by the last 'FREEZE'.

Related Words: SAVE-FORTH FORGET COLD FENCE

FSEEK (file offset mode -- prev-position | -1)

FSEEK is used to move the current location in a file at which subsequent calls to FREAD and FWRITE will operate.

The mode parameter, defined with the same names as referenced in the Amiga technical literature, is one of three types:

```
mode                interpret positional value as
OFFSET_BEGINNING    offset from beginning of file
OFFSET_END          offset from end of file
OFFSET_CURRENT      offset from current position
```

See chapter on File I/O for details.

```
( move 5 bytes forward )
MYFILE @  5 OFFSET_CURRENT   FSEEK .
```

Related Words: FREAD FWRITE

FSEEK? (file offset mode -- prev-position)

FSEEK? is functionally identical with FSEEK with one exception; if an error occurs during the seek operation, FSEEK? will print "Error on file seek!" and execute QUIT.

JForth provides for optional automatic cleanup of opened files if an error forces a QUIT. See MARKFCLOSE and UNMARKFCLOSE.

For further information, see FSEEK.

FWRITE (file addr #bytes -- #bytes-written | -1)

FWRITE will write #bytes from reladdr to the file specified by file-pointer at its current location.

FWRITE always returns a value. If the call was successful, it returns the number of bytes that were written; if an error occurred, it returns -1. An error is also indicated by a non-zero value in the user-variable FERROR, immediately following the call.

See chapter on File I/O.

Related Words: FOPEN 0FOPEN FSEEK FWRITE?

GETMODULE (-- , <modulename>)

See the chapter on MODULES.

GLOBAL-DEFER (<name> --)

Just like DEFER except DEFER stores its vector in the user area while GLOBAL-DEFER stores its vector in the dictionary. This distinction is important in multi-tasking Forths.

See section on DEFER.

Related Words: WHAT'S IS DEFER

GRAPHICS? (--)

```
GRAPHICS_LIB
GRAPHICS_NAME
```

Used to manage the Amiga Graphics Library . See LIBRARY .