D!   (d addr --)   "d store"

Store the 64 bit double number D at memory address ADDR. ADDR must be even. Use ODDD! for odd addresses.

```
: DVARIABLE ( name -- , define double variable )
   VARIABLE CELL ALLOT ;
DVARIABLE DVAR1
$ 12345678.32415762 DVAR1 D!
DVAR1 D@ D.
```

`Related Words: D@ D+ D- 2@ 2!`

D+   (d1 d2 -- d1+d2)   "d plus"

Add two 64 bit double numbers.

D-   (d1 d2 -- d1-d2)   "d minus"

Subtract top double number from second double number.

D.   (d --)   "d dot"

Output the ASCII string for a double number. Utilizes JForth COMMAS mode (commas inserted every 3 digits in decimal; 4 digits in other bases). See COMMAS.

```
4567884. D. ( prints 4,567,884 )
```

`Related Words: COMMAS NOCOMMAS . D.R  .R U. U.R`

D.R   (d field-size --)   "d dot r"

Print a double number D right justified in a field of FIELD-SIZE characters.

```
12345678.90  12 D.R
```

`Related Words: D. . COMMAS NO-COMMAS .R U.R U.`

D2*   (d -- d*2)   "d two times"

Double number fast multiply by 2.

`Related Words: 2* *`

D2/   (d -- d/2)   "d two slash"

Double number fast divide by 2.

```
10.  D2/  D. ( will print out 5 )
```

`Related Words: D2*`

D<   (d1 d2 -- flag)   "d less than"

Compare two double numbers. Return true if D1 less than D2.

D=   (d1 d2 -- flag)   "d equals"

Compare two double numbers. Return true if D1 equals D2.

D@   (addr -- d)   "d fetch"

Fetch the double number D from even address ADDR.

`Related Words: ODDD@ @ C@ W@ 2@`

DABS (d -- |d|) "d absolute"

Calculate absolute value of D.

Related Words: ABS

DCALL (various-params <library> <routine> -- D0 D1)

DCALL is functionally identical with CALL, except it returns both D0 and D1 (-- D0 D1) as a double number.

Double number results are returned by the MATHIEEEDOUBBAS library. See the chapter on Calling Amiga Libraries

Related Words: CALL

DDROP (d --) "d drop"

Discards the double number D that was on top of the stack.

Related Words: 2DROP DROP RDROP

DDUP (d -- d d) "d dup"

Duplicates the top stack item d .

Related Words: DUP 2DUP

DECIMAL (--)

Set numeric base to decimal by setting BASE to 10.

```
    HEX 100 DECIMAL . ( prints 256 )
```

Related Words: HEX BINARY BASE

DEF (<name> --)

DEF will disassemble a given Forth word in Motorola format. See the chapter '68000 Assembly'.

Related Words: WORDS-LIKE DISM ADISM DUMP '

DEFER (<name> --)

Create a vectored word.

DEFER creates a function that has a user variable area storage cell containing a pointer to the word it will execute when it is called. It is a vectored execution word. Use IS to change what a deferred word will execute. Use WHAT's to get the CFA of what a deferred word will execute.

```
    DEFER EXAMPLE
    EXAMPLE  ( will print nothing, calls QUIT )
    ' ORDER IS EXAMPLE  ( will set EXAMPLE  to execute ORDER )
    EXAMPLE  ( will execute ORDER )
    WHAT'S EXAMPLE  ( will leave the CFA of ORDER )
    >NAME ID.       ( prints ORDER )
```

See the section on DEFER under Forth Tools.

Related Words: WHAT'S IS GLOBAL-DEFER

DEPTH (-- #of-cells-on-stack)

Counts the number of cells on the stack . Places the count on top of the stack .

```
    : CHECK-OPERATION  ( -- , monitor stack change )
```

```
          DEPTH >R  DO-OPERATION
          DEPTH R> - ABS  ( calculate change )
          ." Stack depth changed by " . CR ;
```

Related Words: US-DEPTH RDEPTH .S 0SP SP@

DETACHMODULE   (-- , <modulename>)

See the chapter on MODULES.

DICTIONARYSIZE   (-- var-addr)

See the chapter on CLONE.

DIGIT   (char base -- digit true | char false)

DIGIT attempts to convert the ASCII character CHAR to a number according to the given numeric BASE.

If successful, it returns the converted digit and a TRUE, otherwise it leaves the unaffected CHAR and a FALSE.

```
     DECIMAL  ASCII 9  10  DIGIT SWAP . .  ( print 9 -1 )
     ASCII A  10  DIGIT SWAP . . ( failed, print 65 0 )
```

Related Words: (NUMBER)

DISKFONT?

```
     DISKFONT_LIB
     DISKFONT_NAME
```

Used to manage the Amiga DISKFONT library. See :LIBRARY .

DLITERAL   "d literal"

Compile time: (d --)

Run time: (-- d)

Compile a literal double number. DLITERAL is an immediate word that takes a double number from the stack, and compiles it into the dictionary following DLIT . When this code is executed, the number will be pushed to the stack. DLITERAL is similar to LITERAL which compiles a single precision number.

```
     : D64K+   ( d -- d+64K , add 64K to double number on stack )
     \ calculate double 64K at compile time
         [ DECIMAL 64 1024 * S->D ]
         DLITERAL     \ compile DLIT and the double number
         D+ ;         \ and, at run time, add it to whatever
```

DNEGATE   (d -- -d)   "d negate"

Negate the value of a double number.

The bitwise logical operation is -1 XOR 1+

```
     3.4 DNEGATE D.  ( prints -34 )
```

Related Words: NEGATE

DO   (limit index --)

DO marks the beginning of a DO....LOOP instruction.

If, at run time the limit is not larger than the index, DO will NOT execute the body of the loop;

instead it will drop the terms, and jump past the corresponding LOOP.

Standards: 83. '79 and fig are slightly different, executing the body of the DO-LOOP at least once, regardless of limit-index relationship.

```
     : PRINT-0-9  ( -- )  10 0  DO  I .  LOOP  ;
```

`Related Words: LOOP -LOOP  +LOOP  LOOP-DROP`

DO-DOES-SIZE   (-- N)   "do does size"

DO-DOES-SIZE is a constant equal to the number of bytes between the CFA of a CREATE word and the data area. It is used to define >BODY and BODY> .

`Related Words: >BODY  DOES>  >PARENT  VLINK>  CREATE  CONSTANT`

DOES>   (-- addr , at run time)

DOES> is used with CREATE to create new defining words in Forth. The code following CREATE determines how the new word is created. The code following DOES> determines what the new word does.

We could define a simple version of constant as:

```
     : CONSTANT CREATE , ( save number in dictionary )
        DOES> @ ( fetch saved value ) ;
     73 CONSTANT CON1  ( execute CREATE portion )
     CON1 .  ( execute DOES> portion then print 73 )
```

Suppose we want to CREATE a word that contains an offset supplied on the stack at compile time. This word should add the offset to a number on the stack at run time. This is a simplified version of the structure members used for accessing Amiga 'C' structures. Here is how that code would be produced.

```
     : ADDER  ( offset -- , define new kind of Forth word )
        CREATE   ( offset -- , how to make )
           ,  ( save offset in dictionary )
        DOES>    ( n addr -- value+offset , what to do )
           @ + ( fetch saved offset and add it to N )
     ;
     20 ADDER ADD20  ( n -- n+20 , make one )
     12 ADDER DOZENMORE   ( n -- n+12 )
     100 ADD20 .  ( now do it, print 120 )
     100 DOZENMORE . ( print 112 )
```

`Related Words: CREATE  IMMEDIATE`

DOLINES (<filename> --)

Process each line of a file using a deferred word. See the chapter on file I/O.

DOS   (<command-line> --)   "dos"

This word will parse the rest of the current line of the input stream, submitting it to AmigaDOS as a CLI command line.

Virtually any AmigaDOS command may be invoked, with the following limitations:

1. AmigaDOS output will appear in the JForth window, but cannot be parsed or interrupted. Generally, AmigaDOS commands that produce long streams of text are better executed from a CLI. The DOS interface is primarily provided for task and file control.
2. Commands executed via DOS are executed in their own environment which terminates when the

command ends. Therefore, any AmigaDOS command that changes only the local environment (like CD) will have no effect. JForth provides its own CD for changing directories. See CD.

```
DOS COPY MYFILE RAM:
DOS DATE
DOS CD   ( will print the current directory )
```

3. DOS will look for the command in the C: directory. It will not use the search path. If you want it to look in a directory in RAM: instead of on the Workbench disk, you must assign C: to your directory in RAM.

4. DOS cannot be used in a colon definition because it is not IMMEDIATE. Use $DOS instead.

`Related Words: $DOS DOSCOMMAND CD`

DOS0   (-- addr)   "dos zero"

DOS0 returns the address of the string buffer used for converting forth-type strings to AmigaDOS compatible null-terminated strings.

The DOS0 buffer resides in the user-area, and is 256 characters long. A count byte is maintained 1 byte below DOS0.

`Related Words: >DOS  +DOS`

DOS?   (--)   "dos question"

`DOS_NAME`	`dos name`
`DOS_LIB`	`dos lib`

Used to manage the DOS library. See :LIBRARY. DOS is opened and closed automatically by JForth.

DP   (-- addr)   "d p"

A user-variable used to hold the next available location in the dictionary for this user.

DP is referenced by several high-level words, and is therefore not generally needed by the programmer. The contents of DP are put on the stack by HERE , and incremented by ALLOT.

Whenever the system is INTERPRETing, the contents of DP MUST BE WORD-ALIGNED! See ALIGN.

Standards: fig '79 '83

`Related Words: ALLOT HERE ALIGN`

DPLIMIT   (-- addr)   "d p limit"

User variable that holds the maximum value allowed for dictionary to grow to. Checked by ?STACK . The data stack and the dictionary occupy the same memory area. The stack starts at the top and grows down. The dictionary starts at the bottom and grows up.

Standards: JForth

`Related Words: ?STACK  DP`

DPL   (-- addr)   "d p l"

User variable that shows how many digits are to the right of a decimal point when converting a character string to a number . When no decimal point is seen , NUMBER leaves -1 in DPL .

Standards: fig '79 '83

```
23.45 DPL ?  ( prints 2 )
```

`Related Words: USER INTERPRET NUMBER (NUMBER)`

`DROP   ( n -- )`

DROP discards the top value N on the stack.

Standards: fig '79 '83

```
    DUP DROP  ( cancel each other out )
```

`Related Words: DDROP 2DROP RDROP XDROP`

`DST   ( addr <structure> -- )   "dump struct"`

Dump a structure at address ADDR using the named structure as a template. DST will display the contents of each member, whether it is signed or unsigned, its width in bytes, and its name.

```
    GETMODULE INCLUDES
    BITMAP MYBMAP
    3 MYBMAP ..! BM_DEPTH
    MYBMAP DST BITMAP  ( display contents )
```

`Related Words: DUMP  ..@  FILE?`

`DSWAP   ( d1 d2 -- d2 d1 )    "d swap"`

```
    or ( a b c d -- c d a b )
```

Reverses the top two pairs of numbers on the stack. This can be used to swap two double precision numbers.

Standards: fig '79 '83, sometimes called 2SWAP. Both are allowed in JForth.

`Related Words: SWAP 2SWAP DDUP >R`

`DU2*   ( ud1 -- ud1*2 )   "d u 2 times"`

Left shift the unsigned double number UD1. MSB is shifted out and discarded. 0 is shifted in to LSB. This can be used as a fast multiply by 2 or as a shift.

Standards: JForth unique

`Related Words: * U* 2* U2* M*`

`DU2/   ( ud1 -- ud1/2 )   "d u 2 slash"`

Right shift the unsigned double number UD1. 0 is shifted in to MSB . LSB is shifted out and discarded. This can be used as a fast divide by 2.

Standards: JForth unique.

`Related Words: / */ u/ U2/`

`DU<    ( d1 d2 -- flag )   "d u less than"`

Flag = TRUE if D1 is less than D2, FALSE otherwise. Same as U< only for double numbers.

Standards: '83

`Related Words: <  >  U<`

`DUMP   ( addr u -- )`

Display memory from ADDR for U bytes on the standard EMIT device.

Each line displays 16 bytes in hexadecimal and ASCII forms; any control/non-printing characters in the ASCII columns will be displayed as a period . The CONSOLE window should be set to the full screen width.

Every 23 lines, a header is printed showing the low nibble value of the address for that column above both formats.

DUMP allows the programmer to suspend the output; see ?PAUSE.

`Related Words: EMIT ?PAUSE`

`DUP   ( n -- n n )   "doop"`

Duplicates the top item on the stack.

Standards: fig '79 '83

`Related Words: 2DUP DDUP -DUP DUP>R ?DUP`

`DUP>R   ( n -- n )  ( --R-- n )   "doop to r"`

Push a copy of N onto the return stack. This is simply a faster way to do DUP >R

`Related Words: DUP  ?DUP >R`

`ECHO ( -- addr )`

If this variable is TRUE, then INCLUDE will echo each line it compiles to the screen.

`ELSE   ( -- )`

ELSE is used with IF and THEN to specify the code to execute if the conditional was false.

```
: PRINT-ACCOUNT-STATUS   ( -- , good or bad news )
   ." Your account is in the "
   ACCOUNT-BALANCE @  0>
   IF   ." BLACK."
   ELSE ." RED"
   THEN ;
```

The above example uses the convention of putting related IF ELSE and THEN words at the same level of indentation.

Standards: FIG '79 '83. However, the specifics of the JForth implementation is fig. The fig models provided the most error detection and also give the most information.

`Related Words: IF THEN BEGIN WHILE UNTIL BRANCH`

`EMIT   ( char -- )`

EMIT is a DEFERed word, see (EMIT).

The basic function of this word is to make the ASCII character appear on an output device, usually the system CONSOLE.

```
ASCII X EMIT   ( Prints X )
```

`Related Words: TYPE DUMP CR SPACES R D`

`EMIT-TO-COLUMN   ( char column# -- )`

Emit CHAR to the standard EMIT device, until column# is reached by the cursor. This word is useful for producing columnized, tabled lists.

```
: 1LINER ( page# $item -- , connect item and page )
    CR $TYPE  ( draw item )
    ASCII . 40 EMIT-TO-COLUMN  ( draw bar )
```

```
        4 .R  ( page number )
    ;
    17 " Chapter 2" 1LINER
    234 " Chapter 5" 1LINER
```

Related Words: EMIT OUT >NEWLINE CR?

ENABLE_CANCEL (-- addr)

See the chapter on CLONE.

END (--)

Synonym for UNTIL, see UNTIL.

Standards: fig '79 '83 .

Related Words: UNTIL

END-CODE (--)

See the chapter on 68000 ASSEMBLY.

ENDCASE (n --)

ENDCASE is the word used to end a CASE statement. See CASE. ENDCASE is an immediate word that resolves all of the CASE forward branches.

Related Words: CASE OF ENDOF =

ENDOF (--)

ENDOF is a word used in a CASE statement. See CASE.

EOL (-- eol) "e o l" or "end of line"

EOL is a CONSTANT, and returns the value recognized by the Amiga CONSOLE device as a End-Of-Line. You will also find this value used in files at the end of lines.

UNIX and Amiga DOS use $0A, or "linefeed" as the EOL character. The Macintosh, from Apple Computer, uses a $0D or "return" as an EOL.

Related Words: CR FEMIT READLINE >NEWLINE

EOF (-- EOF) "e o f" or "end of file"

EOF is a constant returned by FKEY when an end of file has been reached.

Related Words: FKEY

ERASE (addr count --)

Set COUNT bytes beginning with ADDR to zero.

Standard note: JForth

```
    PAD 256 ERASE  ( clear 256 bytes at PAD )
```

Related Words: FILL CMOVE MOVE DO

ERROR (error# --)

When executed, ERROR will:

1. Go to a new-line, if not already there.

2. Print the text currently at HERE (that which was last INTERPRETed) followed by a '?'.
3. If error# is non-zero, print "Message # " and the error number.
4. Clear the data stack and execute QUIT.

ERROR is the standard error-exit routine for the compiler/interpreter, which passes 0 for the error number.

The use of numbers for error codes is discouraged in JForth; the suggested method is to provide a full-text error message via ?ABORT" , .ERR , or your own error handler, executing QUIT if necessary.

Standards: fig. '79 and '83 use ABORT and ABORT"

```
: IN-DICTIONARY?  ( -- cfa , QUITS with ? if not there )
    BL WORD FIND NOT
    IF 0 ERROR
    THEN ;
```

Related Words: ABORT ABORT" ?ERROR .ERR HERE QUIT

ERRORCLEANUP (--)

See the chapter on CLONE.

EVEN-UP   (n -- n+1 | n)

Increment N if odd. N remains the same if even. If N is odd, add 1 to make it even.

Standards: JForth unique.

Related Words: ALIGN AND

EXEC?   (--)

```
EXEC_LIB
EXEC_NAME
```

Used to manage the Amiga exec library. See :Library. The Exec library is opened automatically by JForth.

EXECUTE   (cfa --)

EXECUTE causes immediate execution of the word whose cfa (code-field-address) is on the stack. Execution then continues at the next instruction following the call to EXECUTE . EXECUTE uses a 68000 Jump Subroutine instruction.

Standards: '79 '83.

```
: HI ." HELLO" ;
' HI .S EXECUTE

THE-ACTION @  ACTIONS-ARRAY @  EXECUTE   ( jump table )
```

Please note!!! If you use a "jump table" in a cloned program, you must initialize the program at run time! See Clone.

Related Words: @EXECUTE '

EXISTS? (<name> -- exists-flag) "exists question"

EXISTS? returns a TRUE if the following word is in the dictionary.

```
EXISTS? GR.INIT .IF ." Graphics loaded!" .THEN
```

Related Words: FIND .NEED ' .IF

EXIT (--)

Exits from a colon definition by executing an RTS, Return from Subroutine, instruction. Don't use inside DO loops .

Standards: '79 '83 . Called ;S in fig.

Related Words: UNTIL WHILE RETURN LEAVE

EXPECT (addr maxchars --)

Used for inputting a string. Read up to MAXCHARS characters from current KEY device to address starting at ADDR. EXPECT will echo characters as they are entered. EXPECT will stop getting characters when the maximum is reached or when a Carriage Return key is hit. EXPECT handles Backspace characters and Control-X. The number of characters read will be available in the user-variable SPAN following the call.

The Command Line History system works by installing a special version of EXPECT. Use HISTORY.OFF to get vanilla traditional version.

When using EXPECT in cloned programs, set the variable RAWEXPECTECHO if you are using RAW: windows and need EXPECT to echo characters as they are typed.

Standards: fig '79 '83

```
    : GET-NAME  ( -- , put name at PAD )
        CR ." What is your name? "
        PAD 80 EXPECT
        CR ." Hello " PAD SPAN @ TYPE
    ;
    GET-NAME
```

Related Words: SPAN QUERY KEY