CHForth

version 1.2.5

CHForth

Authors: Coos Haak and Willem Ouwerkerk

Contents

Chapter 1

Introduction to CHForth

CHForth is a ANSI Standard implemtation for Intel 80x86 processors running MS-DOS or DR-DOS. It runs in real or virtual 16 bits 8086 mode, the default operating mode of DOS.

1.1 Background

This year the ANS Standard document was published and as some members of the Dutch Forth Users Group were writing a new version of Forth, this Standard was adopted. As a writer of Forth compilers since about 1984, I adapted my version to the Standard and CHForth 1.2.5 is the first official release and is presented at the HCC dagen held on November 18 and 19, 1994 in Utrecht.

1.2 Contents of CHForth

CHForth contains a full developmemt environment, it contains a full 8086 assembler with 386 extensions, a source code decompiler and assembly language disassembler and error logging. The use of a multi segment model (the 8086 uses a segmented memory) provides more room in the dictionary than models that use one segment of 64 Kbytes.

1.3 Organisation of this manual

Most chapters in this manual start with a description of the items in the chapter, a description of methods, examples and a glossary of the words that are of interest to the chapter. As this manual is not ready, some chapters are still under construction.

Chapter 2

Installation

To install, you need a fairly compatible PC or AT with at least 8086 or 8088 processor, 256 Kb memory above DOS and 1 Mb free on your harddisk (It is possible to install the program on a 1.2 Mb or 1.44 Mb floppy and with more experience it might be possible on a 720 Kb or even a 360 Kb floppy system).

2.1 Installation on your system

First make a subdirectory with

```
    MD CHF.
```

It is not necessary to have this subdirectory in the root of your C: drive, it can be anywhere in your computer system, even on a ramdisk if you don't trust me.

Then type

```
    CD CHF
```

to go to the directory.

When the distribution floppy is in your A: drive type:

```
    A:PKUNZIP -d A:CHF125
```

to unpack the files. The A:'s can be B:'s in your system. The name CHF125 can be different, it is the name of the .ZIP file on the floppy.

To use the program you can follow two methods, the first is copy the CHFORTH.EXE file and CHFORTH.CFG from the CHF\BIN directory to a directory in your path, like C:\DOS or C:\BIN. The second is to extend the PATH= command in your AUTOEXEC.BAT file with

...\CHF\BIN.

With a DOS editor like EDIT or EDLIN you may have to change the following two lines in CHFORTH.CFG:

```
    S" c:\chf\lib" LIBPATH PLACE
    S" c:\chf\doc" HELPPATH PLACE
```

into for example:

```
    S" d:\programs\develop\forth\ansi\chf\lib" LIBPATH PLACE
    S" e:\helpfiles\programming\forth\ansi"    HELPPATH PLACE
```

You may also have to change the line

```
    S" c:\chf\lib" LIBPATH PLACE
```

in the file CHF\TURNKEY\CHFORTH.CFG

2.2 Directories

CHForth uses some directories, these are made automatically during the installation.

In CHF\BIN are CHFORTH.EXE, the 8086 version, CHF386.EXE, the 386 version and the configuration file CHFORTH.CFG.
In CHF\DOC are the .HLP and .TXT files.
In CHF\LIB are library files
In CHF\MISC are some miscellaneous programs.
In CHF\SPEED are some benchmarks and .LOG files that show the benchmarks on a 40 MHz 486DLC machine.
In CHF\TURNKEY are some application programs in source form.

2.3 DOS interface

In the CHFORTH.CFG are provided some interfaces with DOS.

```
S" "            DOS: OS         -- Go to the operating system for a while
S" dir"         DOS: DIR        -- This looks familiar
\ S" copy"      DOS: COPY       -- Idem
\ S" ren"       DOS: REN        -- Ditto
S" list"        DOS: L          -- View a file
S" sz"          DOS: SZ         -- Tom Zimmer's editor
S" ne"          DOS: NE         -- Peter Norton's editor
S" nc"          DOS: SHELL      -- Alias OS exists already
S" chforth"     DOS: CHFORTH    -- Load another copy, probably useless
```

```
S" ts"          DOS: TS         -- If you have this program
S" ts *.frt"    DOS: ST         -- Search text in *.FRT files
S" grep"        DOS: GREP       -- Idem
S" ls"          DOS: LS         -- My version of DIR, source in \TURNKEY
```

At the left is the name of the program or command as it is known to DOS, after the word DOS: the name of the program as it is known to CHForth. If you do not have the program NE.COM you can delete that line. Then the file LIB\EDITOR.FRT will use SZ.COM as the default editor. If you do not have TS.EXE delete those lines. LS.EXE can be made by CHForth itself, see the chapter about turnkey programs.

2.4 CHF386.EXE

If you do have a 386SX, 386DX, 486SX, 486DX or even a Pentium, you can rename the file CHF386.EXE to CHFORTH.EXE and use this program. The difference is in some arithmetic routines that now use 32 bit aritmetic for speed and the shifting is more efficient and you can use some 386 instructions in the assembler. The program still runs in real or virtual 8086 mode, for 32 bit Forth implementations, see the literature.

2.5 Starting CHForth

The file CHFORTH.EXE has to be in the current directory or a directory mentioned in the DOS environment variable PATH or you can prefix the name of the program with the path. See your DOS manual if this is not clear. When you type CHFORTH at the prompt, it tries to read the CHFORTH.CFG that is in the current directory and else the one that is in the directory where CHFORTH.EXE is found. In this way you can have different configuration files that can be tailored to the need of the moment, for example you can have different libraries and helpfiles in other paths than the standard ones. When the configuration file is read, the word .FREE (which is an option on the last line of the file) is executed to give you some information about the size of the program and how many bytes there are free in each of the three segments. On the command line you can give parameters that have to be normal Forth, like:

```
        CHFORTH in life
```

to load the program in the file LIFE.FRT. When the loading is done, a diagnostics line is given, first three numbers giving the bytes compiled in the three segments, the sum of it, the number of bytes compiled per minute, the number of lines, the number of lines per minute and then the count of seconds elapsed. These are all since the loading of the configuration file.

2.6 Leaving CHForth

The standard way to leave CHForth is type BYE. This words takes care to reset used interrupt vectors to their initial values (see also chapter 17). The same is accomplished by typing ALT+Q, when the module -accept is present.

You can also return with <number> HALT to return a 8 bit code to DOS that can be tested with ERRORLEVEL, when you run CHForth in a batch file. This also can be used in make files. When you press ALT+X, CHForth terminates with a returncode of 1.

Chapter 3

Loading programs

The normal way in Forth for compiling programs is loading them in source form from disk. In CHForth this can be done by loading blocks and by loading textfiles.

3.1 Loading blocks

This paragraph is for those that still use block files like in the sixties and seventies.

First you have to include extensions for handling block files, this can be done by typing at the DOS prompt the following line:

```
    C:\CHF\>chforth empty in blockext close save blocks bye
```

Now a program BLOCKS.EXE will be generated, of course the name is arbitrary. When the file BLOKKEN.BLK exist in the current directory, this program will automatically open that file at startup. It is a blocks file organised in 100 blocks. The first one can be loaded with

```
    1 LOAD .
```

When the file BLOKKEN.BLK is not present, a message is given but this is not considered an error.

Some examples are in this file. You can browse through the file by typing BROWSE that has its own help. A simple editor in FIGFORTH style is loaded with the file blockext.frt.

Opening other files with extension .BLK is possible. Open an other one with:

```
    S" MYBLOCKS" USE-BLOCKS
```

When the first screens of the file BLOKKEN.BLK have been loaded with:

```
    1 LOAD
```

you can use now:

```
    OPEN MYBLOCKS
```

The use of CLOSE is optional when you open another file or when you leave the program because it is present in OPEN and BYE .

It is also possible to create a new block file by:

```
    100 MAKE-BLOCKS-FILE MYBLOCKS
```

to create a file MYBLOCKS.BLK containing 100 blocks.

You can change the default extension for example:

```
    S" .SCR" BEXT$ PLACE
```

As I now seldom use blocks, further help is not available, try to figure out the workings by reading the FigForth manual or the source.

3.2 Loading text files

In Europe, already in the seventies, Forth used standard operating system files, that could be edited, copied and printed with programs already available in the system software.

This is the preferred way in CHForth. To load the file MYFILE.FRT you can use a few methods:

```
    1) INCLUDE MYFILE.FRT
    2) IN MYFILE
    3) S" MYFILE.FRT" INCLUDED
```

The second is preferred and shorter. The default extension .FRT is in the counted string at FEXT$, changing this is unwise.

Files from CHF\LIB can be loaded by:

```
    IN LIB\MYLIB
```

But the normal way is by

```
    NEEDS -MYLIB
```

as some programs depent on this procedure. The word NEEDS will skip loading if the MARKER -mylib already is loaded. You may leave out the minus sign, this is only to remember that the word MARKER is the first defining word in this file and you can forget the compiled words with -mylib .

3.3 More about loading

The files on the distribution disk use always .FRT as default extension. So the extension is never mentioned to load files with IN or NEEDS .
When an error occurs during loading and you have an editor installed in the CHFORTH.CFG file for NE or SZ, typing WHAT gets you in the editor on the offending line. Also some information about the error is written to a file ERROR.LOG in the current directory so you could perhaps determine what caused the error.

3.4 Load words glossary

FEXT$ "f-ext-string" EXTRA

(-- c-addr)
c-addr is the address of a counted string containing the default extension of Forth text files.

IN EXTRA

("name" --)
Skip leading space delimiters. Parse name delimited by a space and load the file with that name. If the length of name is zero, load the file that was previously load with IN .

INCLUDE EXTRA

("name" --)
Skip leading delimiters. Parse name delimited by a space and load the file with that name. The appropriate extension must be included in name.

INCLUDE-FILE FORTH

(fileid --)
Remove fileid from the stack. Save the current input source specification, including the current value of SOURCE-ID . Store fileid in SOURCE-ID . Make the file specified by fileid the input source. Store zero in BLK . Other stack effects are due to the words INCLUDEd.

Repeat until end of file: read a line from the file, fill the input buffer from the contents of that line, set >IN to zero, and interpret.

Interpretation begins at the file position where the next file read would occur.

When the end of the file is reached, close the file and restore the input source specification to its saved value.

An ambiguous condition exists if fileid is invalid, if an I/O exception occurs reading fileid, or an I/O exception occurs while closing fileid. When an ambiguous condition exists, the status (open or closed) of any files that were being interpreted is implementation defined.

INCLUDED FORTH

(c-addr u --)
Remove c-addr u from the stack. Save the current input source specification, including the current value of SOURCE-ID . Open the file specified by c-addr u, store the resulting fileid in SOURCE-ID and make it the input source. Store zero in BLK . Other stack effects are due to the words INCLUDEd.

Repeat until end of file: read a line from the file, fill the input buffer from the contents of that line, set >IN to zero, and interpret.

Interpretation begins at the file position where the next file read would occur.

When the end of the file is reached, close the file and restore the input source specification to its saved value.

An ambiguous condition exists if the named file can not be opened, if an I/O exception occurs reading the file, or an I/O exception occurs closing the file. When an ambiguous condition exists, the status (open or closed) of any files that were being interpreted is implementation defined.

LOAD FORTH

(i*x u -- j*x)
Save the current input source specification. Store u in BLK , thus making block u the input source and setting the input buffer to encompass its contents, set >IN to zero, and interpret. When the parse area is exhausted, restore the prior input source specification. Other stack effects are due to the words LOADed.

Exceptions -33, -34 or -35 will occur if u is zero, or is not valid block number.

NEEDS EXTRA

(name --)
Find name and when found continue. When not found, load the file with the same name (excluding a trailing minus sign) from the directory in LIBPATH .

THRU FORTH

(i*x u1 u2 -- j*x)
LOAD the mass storage blocks numbered u1 through u2 in sequence. Other stack effects are due to the words LOADed.

Chapter 4

How to get help

CHForth offers a number of ways to help the user during running of the program. The tools available are helpfiles, a decompiler, a viewer, a disassembler, a file browser and a referencer.

4.1 The helpfiles

In the directory CHF\DOC are some files with the extension .HLP. These are normal textfiles. Do not change their structure, they are generated with a glossary generator and use the word \G that you will find in the source files (with extension .FRT). They are used by the HELP command. If you can not remember the use of for example the word DUP just type 'HELP DUP'.

```
FORTH> help dup
File: KERNEL.HLP
DUP              "dupe"                                   FORTH
    ( x -- x x )
    Duplicate x.
```

On top is the name of the helpfile and on the left on the next line is the word looked after. It is sometimes followed by the pronounciation in double quotes. When the helpfile is KERNEL.HLP the word on the far right is FORTH or EXTRA, the wordlists in which the definition is compiled. In other helpfiles it is the name of the file in the CHF\LIB directory where the word is defined.

When a full screen is displayed, HELP waits for a key press. Pressing Esc stops, others will continue displaying the remaining text.

Needed file: LIB\HELP.FRT

4.2 The file browser

You can look up words in files in the current directory by typing SF followed by a string of characters, including spaces. For example to find any occurence of the string 'SWAP DUP':

```
FORTH> sf swap dup
  1 BENCH.FRT      23                    DUP SWAP DUP ROT DROP 1 AND
  2 COREWARS.FRT   375    SWAP DUP @ 5 * SWAP CELL+         \ #bytes and st
```

On the left is the number of lines found, followed by the filename, followed by the linenumber in the file and the line itself is printed till the end of the screen line. Leading and trailing spaces are significant.

The word SL does the same, but uses the files in CHF\LIB

The word LOOK must be followed with a filename, including the extension and an optional path, and the string to look for. It looks in a single file.

```
FORTH> look corewars.frt swap dup
  1    SWAP DUP @ 5 * SWAP CELL+            \ #bytes and start addr ok
```

When a full screen is displayed, these words wait for a key press. Pressing Esc stops, others will continue displaying the remaining text.

Needed file: LIB\SEARCHER.FRT

4.3 The referencer

To look up words in compiled code, you can use the referencer, e.g.:

```
FORTH> ref cells
        DECOMPILER
TAB@
        EXTRA
RESTORE-SCREEN  RECOVER-SCREEN  SAVE-SCREEN  SAVE  FINDMESSAGE  >LOCAL  H,  L,
DOFORGET  DIAGNOSE
        FORTH
(VIEW)  2VARIABLE  VARIABLE
13 references of: $1008 CELLS found. ok
```

Needed file: LIB\REF.FRT

4.4 Help words glossary

(REF) "paren-ref" REF

(addr --)

Find compiled references in colon definitions of addr in all word lists. Display the words where the references occur and the count of the words where the references are found.

ANY SEARCHER

("ccc" --)

Skip leading space delimiters. Parse ccc delimited by a space. Search the files with extension given by HEXT$ in the directory given by HELPPATH . Display the description of the names that contain ccc. If a full screen is displayed, wait for the user to press a key. Stop if the key is the escape key.

HELP HELP

("name" --)

Skip leading space delimiters. Parse name delimited by a space. Look up name in the files with extension given in HEXT$ in the directory given by HELPPATH and display the description of name. As a binary search on the sorted file is performed, only one description per file is displayed. When a full screen is displayed, wait for the user to press any key, escape stops. Otherwise convert name to a number (the prefixes % $ # & etc. are permitted) and display its type and decimal value and the character if it can be displayed or display the exception message if it is defined for the number.

LOOK SEARCHER
("name" "ccc" ---)
Skip leading space delimiters. Parse name delimited by a space. Skip leading SEPARATOR delimiters. Parse ccc delimited by SEPARATOR . Search file name with optional extension given by FEXT$. Find ccc in the file. Display the number of the lines found, the line number and the line containing ccc depending on the width of the screen. If a full screen is displayed, wait for the user to press a key. Stop if the key is the escape key.

REF REF
("name" --)
Skip leading space delimiters. Parse name delimited by a space. Find compiled references in colon definitions of name in all word lists. Display the words where the references occur and the count of the words where the references are found.

SF "search-forth" SEARCHER
("ccc" --)
Skip leading SEPARATOR delimiters. Parse ccc delimited by SEPARATOR . Search the files with extension given by FEXT$ in the current directory. Find ccc in the files. Display the number of lines found, the name of the file, the line number and the line depending on the width of the screen. If a full screen is displayed, wait for the user to press a key. Stop if the key is the escape key.

SL "search-libraries" SEARCHER
("ccc" --)
Skip leading SEPARATOR delimiters. Parse ccc delimited by SEPARATOR . Search the files with extension given by FEXT$ in the directory given by LIBPATH . Find ccc in the files. Display the number of lines found, the name of the file, the line number and the line depending on the width of the screen. If a full screen is displayed, wait for the user to press a key. Stop if the key is the escape key.

The decompiler, disassembler and viewer are described elsewhere.

Chapter 5

Local variables

Instead of creating definitions with complex stack uses, the programmer can use variables. The problem with variables is that they are not local to a definition and other words can use them and may produce unwanted side-effects.

ANS Forth offers a way to use variables local to a definition that are not known outside that definition. In this way the user can give them names that do not conflict with global or other local variables. A further improvement is that the use of a local variable's name will give the value directly without @, like a VALUE . To change the value, use TO , +TO or CLEAR.

5.1 Use of locals

The calculation of the discriminant in square roots is without the use of local values:

```
: DISCRIMINANT      ( a b c -- d )      \ d=b*b-4*a*c
        SWAP                            \ a c b
        DUP *                           \ a c b*b
        -ROT                            \ b*b a c
        *                               \ b*b a*c
        4 *                             \ b*b 4*a*c
        -                               \ d
    ;
```

The standard ANS Forth way to use locals is as follows:

```
: DISCRIMINANT      ( a b c -- d )      \ d=b*b-4*a*c
        LOCALS| c b a |                 \ stack empty
        b b *                           \ b*b
        4 a * c *                       \ b*b 4*a*c
        -                               \ d
    ;
```

Remember that at the start of the definition, the value on the top of the stack will be placed in the first local value. The names after the words LOCALS| are therefore in reverse order to the stack diagram.

In CHForth the restriction that no operation is allowed between declaring locals is not applicable (but the program will be non-standard):

```
: DISCRIMINANT      ( a b c -- d )      \ d=b*b-4*a*c
        LOCAL c                         \ a b
        DUP *                           \ do some operation
        LOCAL b*b                       \ a
        c * -4 *                        \ -4*a*c
        b*b +                           \ d
    ;
```

5.2 Internals of local variables

When defining a local variable the pointers HERE and HHERE are temporary changed to a special area that is also used by FLYER to compile code and headers that will not interfere with the normal compiling.

For every local variable the word PUSH-LOCAL is compiled that transfers the value on the top of the stack to the local stack and pushes the address of special routine on the return stack. At the end of the definition this routine is executed and it will discard the storage area on the local stack and then return to the calling definition with EXIT as normal.

When decompiling you will see that the first named local will be called LOCAL 0 and the second LOCAL 1 and so on.

The word (LOCAL) can be used do make defining words for locals. Try decompiling the definition of LOCAL or LOCALS| to see examples for this.

5.3 More local types

Double locals and other types are defined in the file LIB\DLOCALS.FRT and can be included by NEEDS -dlocals .

5.4 Local words glossary

(LOCAL) "paren-local-paren" FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (c-addr u --)
When executed during compilation, (LOCAL) passes a message to the Forth system that has one of two meanings. If u is non-zero, the message identifies a new local whose word name is given by the string of characters identified by c-addr u. If u is zero, the message is 'last local' and c-addr has no significance. The result of executing (LOCAL) during compilation of a definition is to create a set of named local identifiers, each of which is a word name, that have execution semantics within the scope of that definition's source only.

local Execution: (-- x)
Push the local's value, x, onto the stack. An ambiguous condition exists when (LOCAL) is executed while in interpret state.

Note: This word is not intended for direct use in a definition to declare that definition's locals. It is instead used by system or user compiling words. These compiling words in turn define their own syntax, and may be used directly in definitions to declare locals.

+TO "plus-to" EXTRA

Interpretation: (n|u "name" --)

Skip leading space delimiters. Parse name delimited by a space. Add n|u to name. Exception -32 occurs if name was not defined by VALUE or VARIABLE .

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VALUE , VARIABLE or (LOCAL).

Run-time: (x --)
Add n|u to name.

CLEAR EXTRA

Interpretation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Store zero in name. Exception -32 occurs if name was not defined by VALUE or VARIABLE .

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VALUE , VARIABLE or (LOCAL).

Run-time: (--)
Store zero in name.

END-LOCAL EXTRA

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (--)
Terminate creation of local values.

LOCAL EXTRA

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space.

Create a definition for name with the execution and run-time semantics defined below.

Execution: (x --)
Store x in name.

name Execution: (-- x)
Place x on the stack. The value can be manipulated by TO +TO and CLEAR .

LOCAL-WORDLIST ONLY
(-- wid)
Return the wid of the LOCAL-WORDLIST .

LOCALS| "locals-bar" FORTH
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("namen" .. "name2" "name1" "|" --)
Define up to 8 local variables with "name1" to "namen". The list of locals to be defined is terminated with "|". The actual number in CHForth may be greater, depending on the length of the input line. Append the run-time semantics for name given below.

name Run-time: (-- x)
Place x on the stack. The value can be manipulated by TO +TO and CLEAR .

TO FORTH
Interpretation: (x "name" --)
Skip leading space delimiters. Parse name delimited by a space. Store x in name. Exception -32 occurs if name was not defined by VALUE or VARIABLE .

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VALUE , VARIABLE or (LOCAL).

Run-time: (x --)
Store x in name.

Chapter 6

Forget and forget fields

As Forth can be used as a development environment, sometimes the user wants to get rid of old definitions and start new ones. The words provided for this are the old word FORGET and the new word MARKER .

6.1 FORGET

When FORGET followed by a name is typed in all definitions made later than name and name itself are forgotten, the dictionary pointer (HERE) is reset as all Forth versions do, but there is a field in the header of each word that may contain the execution token of a special routine. Every time FORGET is typed, it scans the headers, starting with the newest word by using HIGHEST and looks for the contents of this field and executes the token and continues with the next word. This routine is entered with the data field of the word found by HIGHEST on the stack and can therefore perform some restoration action with that address. For example when a word from type INTVEC is forgotten, it will restore the former contents of the vector. A colon definition has a forget routine that places the list dictionary pointer back to what it was when the word was created. Remember that any word that was in the dictionary when EXTEND was executed can not be forgotten, and any word that was in the executable file when CHForth was started, because SAVE contains EXTEND .

6.2 MARKER

The new method is MARKER . This is a defining word, it creates a definition for the following name. When name is typed, everything including name and later definitions are removed, this process is the same if you typed: FORGET name. As a convention to remember its action, a '-' (minus sign) is sometimes appended in front of name. When name is executed, apart from the normal FORGET action, the search order is restored to the point where MARKER was executed, so you do not have to remember it yourself.

6.3 Examples

```
: ONE ;  : TWO ;  : THREE ;  : FOUR ;   \ some new definitions
FORGET THREE                            \ forget the last two
: FIVE ; : SIX ;                        \ and start other words

FORTH DEFINITIONS                       \ Set a starting order
MARKER -vergessen                       \ Set a marker
VIERTE DEFINITIONS                      \ Set new search order
: Eins ONE ; : Zwei TWO ; : Sechs SIX ;
FUENFTE DEFINITIONS                     \ Change order
-vergessen                              \ The last four are gone
ORDER                                   \ Will print FORTH as 1st

DOER: DOMESSAGE                         \ address of data field
        CR COUNT TYPE                   \ Print the message
    ;

: MESSAGE  CREATE [CHAR] " WORD C@      \ Define message definer
           CHAR+ ALLOT                  \ Compile the string
        DOMESSAGE
    ;

:NONAME  CR ." Forgetting message "     \ Print a forget message
        COUNT TYPE                      \ The same address as
    ;                                   \ after DOES>
    IS-FORGET DOMESSAGE                 \ Put in a MESSAGE type

MESSAGE MESS-1 This is message one"
```

MESSAGE MESS-2 This is message two"

6.4 Forget words glossary

```
FENCE                                                       EXTRA
    ( -- a-addr )
    a-addr is the adress of a cell containing the dictionary pointer
    since the last SAVE or EXTEND . Forgetting of words created when
    the dictionary pointer was less than this value is not possible.

FORGET                                                      FORTH
    ( "name" -- )
    Skip leading space delimiters. Parse name delimited by a space.
    Find name in the compilation word list, then delete name from the
    dictionary along with all words added to the dictionary after
    name. Exception -13 occurs if name cannot be found. Exception -15
    occurs if FORGET removes a word required for correct execution.

    Note: this word is obsolescent and is included as a concession to
    existing implementations.

    Note: In CHForth words can be protected against FORGET with
    EXTEND and SAVE .

HEAD>FORGET        "head-to-forget"                         EXTRA
    ( dea -- h-addr )
    h-addr is the forget field address of the dictionary entry dea.

HIGHEST                                                     EXTRA
    ( -- wid dea )
    Return the dictionary entry address of the newest definition with
    dictionary entry address dea and the word list identification wid
    in which it is compiled. Used in FORGET .

IS-FORGET                                                   EXTRA
    ( xt "name" -- )
    Skip leading space delimiters. Parse name delimited by a space.
    Append the semantics of execution token xt to the forget method
    of name.

MARKER                                                      FORTH
```

```
( "name" -- )
Skip leading space delimiters. Parse name delimited by a space.
Create a dictionary for name with the execution semantics defined
below.

name Executing: ( -- )
Restore all dictionary allocation and search pointers to the
state they had just prior to the definition of name. Remove the
definition of name and all subsequent definitions. Restoration of
any structures still existing that could refer to deleted
definitions or deallocated data space is not necessarily
provided. No other contextual information such as numeric base is
affected.
```

Chapter 7

Numbers and strings

Standard Forth has two types of numbers, single precision: in CHForth signed numbers from -32768 to 32767 or unsigned from 0 to 65535, and double precision numbers (entered by one or more decimal points in the number) from -2147483648 to 2147483647 or from 0 to 4294967295. The only allowed prefix is the minus sign.

7.1 Numbers

Numbers in other bases than decimal ten are in Standard Forth only possible if you change BASE before the number and restore it afterwards. In CHForth (and other Forths as well) this is solved by prefixing the number by special characters as follows:

- ’#’ for decimal numbers, digits 0..9,
- ’$’ for hexadecimal numbers, digits 0..9 and A..F,
- ’%’ for binary numbers, digits 0 and 1.

The minus sign if present must be after this prefix.

7.2 Characters

Characters are in Standard Forth entered by placing the word CHAR (when interpreting) or [CHAR] (when compiling) and a space before the character. In CHForth this is extended by prefixes. Placing a ’&’ character without a space before a single character places this number on the stack while interpreting or compiles it as a literal. It is also possible to place a ’’’ character just before and just after the desired character.

Control characters are entered by placing the word CTRL (when interpreting) or [CTRL] (when compiling) and a space before an uppercase character which is converted to is value between 0 and 31. An other way is placing a ’^’ character without a space before a single character.

7.3 Strings

Strings in Forth come in two varieties, the first and oldest species is the counted string. On the stack is an address. On that address is a byte containing the size of the string right after that byte. The length of the string is between 0 and 255 both inclusive. It is used by words as WORD and FIND and converted to the new type by COUNT .

The newer version has an address and a length on the stack, this length can be from zero to 65535, practically infinite. These are handled by other words that expect a string on the stack. But it is not generally possible to convert such a string to a member of the old type as this type of strings is often not preceded by a count byte and sometimes the length is larger than 255.

CHForth provides some operators to store and concatenate strings. PACK and PLACE put a c-addr u string on an address as a counted string and PACK leaves this address on the stack and PLACE does not. APPEND places a c-addr u string at the end of a counted string and corrects the size of the compound string. APPEND-CHAR appends a character to a counted string and increments the count of the string with one.

7.4 Numbers and strings word glossary

", "quote-comma" EXTRA

("ccc<">" --)

Parse ccc delimited by ’"’ (double quote) and compile it as a counted string in the dictionary. Execution of HERE just before the execution of ", will give the address of the string.

# "number-sign" FORTH

(ud1 -- ud2)

Divide ud1 by the number in BASE giving the quotient ud2 and the remainder n. (n is the least-significant digit of ud1). Convert n to external form and add the resulting character to the beginning of the pictured numeric output string. An ambiguous condition exists if # executes outside of a <# #> delimited number conversion.

See also: #> #S <#

#> "number-sign-greater" FORTH

(xd -- c-addr u)

Drop xd. Make the pictured numeric output string available as a character string. c-addr and u specify the resulting character string. A Standard Program may replace characters within the string.

See also: # #S <#

#S "number-sign-s" FORTH

(ud1 -- ud2)

Convert one digit of ud1 according to the rule for # . Continue conversion until the quotient is zero. An ambiguous condition exists if #S executes outside of a <# #> delimited number conversion.

See also: # #> <#

(.) "paren-dot" EXTRA

(n -- c-addr u)

Convert n to a numeric output string with a leading minus sign if n is negative.

(D.) "paren-d-dot" EXTRA

(d -- c-addr u)

Convert d to a numeric output string with a leading minus sign if d is negative.

(NUMBER?) "paren-number-question" EXTRA

(c-addr u -- 0 | n 1 | d 2)

Convert a string to a number. If it fails, return a false flag. Otherwise return a single number with a flag of 1 and a double number with a flag of 2. The number is negative if prefixed by '-'. CHForth allows decimal numbers to be prefixed by '#' , hexadecimal numbers by '$' and binary numbers by '%' . These may be followed by ’-’ to signify negative numbers. Single characters are converted to single precision number when prefixed by ’&’ or when they are enclosed by ’’’. Uppercase letters can be converted to the corresponding control characters when prefixed by ’^’.

-TRAILING "dash-trailing" FORTH
(c-addr u1 -- c-addr u2)
If u1 is greater than zero, u2 is equal to u1 less the number of spaces at the end of the character string specified by c-addr u1. If u1 is zero or the entire string consists of spaces, u2 is zero.

. "dot" FORTH
(n --)
Display n in free field format.

.DEC "dot-decimal" EXTRA
(n --)
Display n as a signed decimal number.
See also: .HEX

.HEX "dot-hex" EXTRA
(u --)
Display u as a four digit hexadecimal number with a leading ’$’ character and a trailing space.
See also: .DEC H.

.R "dot-r" FORTH
(n1 n2 --)
Display n1 right aligned in a field n2 characters wide. If the number of characters required to display n2 is greater than n2, all digits are displayed with no leading spaces in a field as wide as necessary.

.S "dot-s" FORTH
(--)
Copy and display the values currently on the data stack. Starting on a new line, a ’(’ (left parenthesis) followed by a space is displayed. Then follow the values on the stack, when BASE contains 10, as signed numbers, unsigned otherwise. At the end a ’)’ (right parenthesis) is displayed.

.S is implemented using pictured numeric output words. Its use will corrupt the transient region identified by #> .

`.SEG` "dot-segment" EXTRA

(u --)

Display u as a four character string if it corresponds to a segment in CHForth else as a four digit hexadecimal string.

`/STRING` "slash-string" FORTH

(c-addr1 u1 n -- c-addr2 u2)

Adjust the character string at c-addr1 by n characters. The resulting character string, specified by c-addr2 u2, begins at c-addr1 plus n characters and is u1 minus n characters long.

`2LITERAL` "two-literal" FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (x1 x2 --)

Append the run-time semantics defined below to the current definition.

Run-time: (-- x1 x2)

Place cell pair x1 x2 on the stack.

`<#` "less-number-sign" FORTH

(--)

Initialize the pictured numeric output conversion process.
See also: # #> #S

`><` "flip" EXTRA

(x1 -- x2)

See: FLIP

`>NUMBER` "to-number" FORTH

(ud1 c-addr1 u1 -- ud2 c-addr2 u2)

ud2 is the unsigned result of converting the characters within the string specified by c-addr1 u1 into digits, using the number in BASE , and adding each into ud1 after multiplying ud1 by the number in BASE . Conversion continues left-to-right until a character that is not convertible, including any "+" or "-" is encountered or the string is entirely converted. c-addr2 is the location of the first unconverted character or the first character past the end of the string if the string was entirely converted. u2 is the number of unconverted characters in the string. An ambiguous condition exists if ud2 overflows during the conversion.

>UPC "to-u-p-c" EXTRA

(char1 -- char2)

Convert char1 to uppercase.

? "question" FORTH

(a-addr --)

Display the value stored at a-addr.

APPEND EXTRA

(c-addr1 u c-addr2 --)

Add u to the numerical value of the character at c-addr2. Store the string specified by c-addr1 u at the character address given by the sum of c-addr2 and the incremented numerical value of the character at c-addr2.

APPEND-CHAR EXTRA

(char c-addr --)

Increment the numerical value of the character at c-addr by one. Store char at the character address given by the sum of the incremented numerical value of the character at c-addr and c-addr.

B. "b-dot" EXTRA

(u --)

Display u as a two digit hexadecimal number with a trailing space.

See also: H.

BASE FORTH

(-- a-addr)

a-addr is the address of a cell containing the current number conversion radix {{2..36}}.

BL "b-l" FORTH

(-- char)

char is the character value for a space.

C" "c-quote" FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("ccc<quote>" --)
Parse ccc delimited by " (double-quote). Append the run-time semantics given below to the current definition.

Run-time: (-- c-addr)
Return c-addr, a counted string consisting of the characters ccc. A standard program shall not alter the returned string.
See also: S"

CHAR "char" FORTH
("name" -- char)
Skip leading space delimiters, Parse name delimited by a space. Put the value of its first character on the stack.
See also: [CHAR]

CMOVE "c-move" FORTH
(c-addr1 c-addr2 u --)
If u is greater than zero, copy u consecutive characters, character-by-character and left-to-right, from c-addr1 to c-addr2. If c-addr2 lies within the source region, memory propagation occurs. (c-addr2 lies within the source region if c-addr2 is not less than c-addr1 and c-addr2 is less than the quantity c-addr1 u CHARS +).
See also: CMOVE> MOVE

CMOVE> "c-move-up" FORTH
(c-addr1 c-addr2 u --)
If u is greater than zero, copy u consecutive characters, character-by-character and right-to-left, from c-addr1 to c-addr2. If c-addr1 lies within the destination region, memory propagation occurs. (c-addr1 lies within the destination region if c-addr1 is greater than or equal to c-addr2 and if c-addr2 is less than the quantity c-addr1 u CHARS +).
See also: CMOVE MOVE

CMOVEX "c-move-x" EXTRA
(x-addr1 x-addr2 u --)
If u is greater than zero, copy u consecutive characters, character-by-character and left-to-right, from extended address

x-addr1 to extended address x-addr2. If x-addr2 lies within the source region, memory propagation occurs. (x-addr2 lies within the source region if x-addr2 is not less than x-addr1 and x-addr2 is less than the quantity x-addr1 u CHARS +).
See also: CMOVE CMOVEX>

CMOVEX> "c-move-x-up" EXTRA
(x-addr1 x-addr2 u --)
If u is greater than zero, copy u consecutive characters, character-by-character and right-to-left, from extended address x-addr1 to extended address x-addr2. If x-addr2 lies within the source region, memory propagation occurs. (x-addr2 lies within the source region if x-addr2 is not less than x-addr1 and x-addr2 is less than the quantity x-addr1 u CHARS +).
See also: CMOVE CMOVEX

COMPARE FORTH
(c-addr1 u1 c-addr2 u2 -- flag)
Compare the string specified by c-addr1 u2 to the string specified by c-addr2 u2. The strings are compared, beginning at the given addresses, character by character, up to the length of the shorter string or until a difference is found. If the two strings are identical up to the length of the shorter string, n is zero if both strings are of equal length, minus-one of u1 is less than u2, and one otherwise. If the two strings are not identical up to the length of the shorter string, n is minus-one if the first non-matching character in the string specified by c-addr1 u1 has a lesser numerical value than the corresponding character in the string specified by c-addr2 u2 and one otherwise.
See also: COMPARE-UPPERCASE

COMPARE-UPPERCASE EXTRA
(c-addr1 u1 c-addr2 u2 -- flag)
Compare the string specified by c-addr1 u2 to the string specified by c-addr2 u2. The strings are compared, beginning at the given addresses, character by character, up to the length of the shorter string or until a difference is found. If the two strings are identical, where lower case characters are considered equal to upper case characters, up to the length of the shorter string, n is zero if both strings are of equal length, minus-one of u1 is less than u2, and one otherwise. If the two strings are not identical up to the length of the shorter string, n is

minus-one if the first non-matching character in the string specified by c-addr1 u1 has a lesser numerical value, where the value of lower case characters are converted to their upper case equivalent values without affecting the strings themselves, than the corresponding character in the string specified by c-addr2 u2 and one otherwise.
See also: COMPARE

CONVERT OBSOLETE
(ud1 c-addr1 -- ud2 c-addr2)
ud2 is the result of converting the characters within the text beginning at the first character after c-addr1 into digits, using the number in BASE , and adding each digit to ud1 after multiplying by the number in BASE . Conversion continues until a character that is not convertible is encountered. c-addr2 is the location of the first unconverted character. An ambiguous condition exists if ud2 overflows.

Note: this word is obsolescent and is included as a concession to existing implementations. Its function is superseded by >NUMBER .

COUNT FORTH
(c-addr1 -- c-addr2 char)
Return the character string specification for the counted string stored at c-addr1. c-addr2 is the address of the first character after c-addr1. u is the contents of the character at c-addr1, which is the length in characters of the string at c-addr2.

COUNTX "count-x" EXTRA
(x-addr1 -- x-addr2 char)
Fetch char from extended address x-addr1 and add 1 CHARS to x-addr1 giving x-addr2.

CTRL "control" EXTRA
("name" -- char)
Skip leading space delimiters, Parse name delimited by a space. Put the value of the control character defined by its first character on the stack. Exception -531 occurs when the character is not in the range {'@'..'_'}.
See also: CHAR [CTRL]

D. "d-dot" FORTH

(d --)
Display d in free field format.

D.R "d-dot-r" FORTH
(d n --)
Display d right aligned in a field n characters wide. If the number of characters required to display d is greater than n, all digits are displayed with no leading spaces in a field as wide as necessary.

DECIMAL FORTH
(--)
Set the numeric conversion radix to ten (decimal).

DIGIT EXTRA
(char +n -- n1 true | char false)
Try to convert char to a digit n1 with number base +n. If the conversion succeeds, return a true flag. Otherwise a false flag.

DPL "d-p-l" EXTRA
(-- a-addr)
a-addr is the address of a cell. When the last interpreted number contained a decimal point, it will contain the number of digits after the decimal point in that number; otherwise the contents are -1.

EXPAND EXTRA
(c-addr1 u1 c-addr2 -- c-addr2 u2)
Copy any non-tab characters in the string specified by c-addr u1 to a string specified by c-addr2 u2. Tab characters are expanded to spaces with a tab distance of 8 positions.

FLIP EXTRA
(x1 -- x2)
Exchange the high and low bytes of x1 giving x2.

H. "h-dot" EXTRA
(u --)
Display u as a four digit hexadecimal number with a trailing space.
See also: .HEX B.

HEX FORTH

(--)
Set the contents of BASE to sixteen.

HOLD FORTH
(char --)
Add char to the beginning of the pictured numeric output string. An ambiguous condition exists if HOLD executes outside of a <# #> delimited number conversion.

INLINE# "inline-number" EXTRA
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(-- x)
Return the inline compiled number, system use only.

INLINE$ "inline-string" EXTRA
(-- l-addr)
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

l-addr is the list address of an inline compiled string. System use only.

JOIN EXTRA
(char1 char2 -- x)
char1 is the low byte of x and char2 is the high byte of x.

KB. "k-b-dot" EXTRA
(u --)
Display the result of division of u by 1024 with one digit after the decimal point followed by a space, the string "Kb" and a space.

LITERAL FORTH
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (x --)
Compile x as a literal. Append the run-time syntax given below

to the current definition.

Run-time: (-- x)
Place x on the stack.

LITERALS EXTRA
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (x1 .. xn n --)
Append the execution semantics defined below to the current definition.

Executing:
(-- x1 .. xn)
Place x1 to xn on the stack.

NUMBER? "number-question" EXTRA
(c-addr u -- 0 | n 1 | d 2)
A word that normally executes (NUMBER?) .

PACK EXTRA
(c-addr1 u c-addr2 -- c-addr2)
Place the string specified by c-addr1 u as a counted string at c-addr2.

PAD FORTH
(-- c-addr)
c-addr is the address of a transient region that can be used to hold data for intermediate processing.

PLACE EXTRA
(c-addr1 u c-addr2 --)
Place the string specified by c-addr1 u as a counted string at c-addr2.

S" "s-quote" FORTH
Interpretation: ("ccc<quote>" -- c-addr u)
Parse ccc delimited by " (double quote). Store the resulting string ccc at a temporary location. The maximum length of the temporary buffer is 255 characters. CHForth allows for the storing of more such strings before new strings start to

overwrite the buffer. A standard program shall not alter the returned string.

Compilation: ("ccc<quote>" --)
Parse ccc delimited by " (double quote). Append the run-time semantics given below to the current definition.

Run-time: (-- c-addr u)
Return c-addr and u describing a string consisting of the characters ccc. A standard program shall not alter the returned string.
See also: C"

SCAN EXTRA
(c-addr1 u1 char -- c-addr2 u2)
Scan the string specified by c-addr1 u1 for an occurrence of char and return the part of the string starting with the found char as a string specified by c-addr2 u2. If the string specified by c-addr1 u1 does not contain char, u2 is zero.

If char is the character for space, control characters are considered equal to char.

SEARCH FORTH
(c-addr1 u1 c-addr2 u2 -- c-addr3 u3 flag)
Search the string specified by c-addr1 u1 for the string specified by c-addr2 u2. If flag is true, a match was found at c-addr3 with u3 characters remaining. If flag is false there was no match and c-addr3 is c-addr1 and u3 is u1.

SIGN FORTH
(n --)
If n is negative, add a minus sign to the beginning of the pictured numeric output string. An ambiguous condition exists if SIGN executes outside of a <# #> delimited number conversion.

SKIP EXTRA
(c-addr1 u1 char -- c-addr2 u2)
Skip leading occurrences of char in the string specified by c-addr1 u1 and return the remaining string specified by c-addr2 u2. If the string specified by c-addr1 u1 contains only occurrences of char, u2 is zero.

If char is the character for space, control characters are considered equal to char.

SLITERAL FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (c-addr1 u --)
Append the run-time semantics given below to the current definition.

Run-time: (-- c-addr2 u)
Return c-addr2 u describing a string consisting of the characters specified by c-addr1 u during compilation. A Standard Program shall not alter the returned string.

SPLIT EXTRA

(x -- char1 char2)
char1 is the low byte of x and char2 is the high byte of x.

SRCSEG "source-segment" EXTRA

(-- a-addr)
a-addr is the address of a cell containing the segment address of the first string in COMPARE and SEARCH . The user is responsible to restore the default value (CSEG) after using an alternative value in COMPARE and SEARCH .

STYPE "s-type" EXTRA

(c-addr u --)
If u is greater than zero, display the character string specified by c-addr and u. The characters are displayed as with SEMIT .

STYPEX "s-type-x" EXTRA

(x-addr u --)
If u is greater than zero, display the character string at the extended address x-addr for a total of u characters. The characters are displayed as with SEMIT .

TYPE FORTH

(c-addr u --)
If u is greater than zero, display the character string specified by c-addr and u.

See also: EMIT

TYPEX "type-x" EXTRA

(x-addr u --)

If u is greater than zero, display the character string at the extended address x-addr for a total of u characters.

TYPEZ "type-z" EXTRA

(x-addr --)

While the character at the extended address x-addr is not zero, display the character and increment x-addr.

U. "u-dot" FORTH

(u --)

Display u in free field format.

U.R "u-dot-r" FORTH

(u n --)

Display u right aligned in a field n characters wide. If the number of characters required to display u is greater than n, all digits are displayed with no leading spaces in a field as wide as necessary.

UD. "u-d-dot" EXTRA

(ud --)

Display ud in free field format.

UD.R "u-d-dot-r" EXTRA

(ud n --)

Display ud right aligned in a field n characters wide. If the number of characters required to display ud is greater than n, all digits are displayed with no leading spaces in a field as wide as necessary.

UPPER EXTRA

(c-addr u --)

Convert the lowercase characters in the string specified by c-addr u to uppercase.

[CHAR] "bracket-char" FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition.

Run-time: (-- char)
Place char char, the value of the first character of name, on the stack.
See also: CHAR

[CTRL] "bracket-control" EXTRA
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -531 occurs when the character is not in the range {’@’..’_’}.

Run-time: (-- char)
Place char, the value of the first character of name, after conversion to a control character, on the stack.
See also: CTRL [CHAR]

Chapter 8

Word lists

Word lists in Forth are a method to group words with specific semantics. Also the search time is reduced when internal words are places in word list that are currently not accessible.

8.1 WORDLIST and VOCABULARY

The Standard way to create a word list is by the word WORDLIST that creates an initially empty list and returns its word list identification (wid), in CHForth its address. This is the base for a more convenient word, VOCABULARY that gives a name to this list. When executed, the vocabulary replaces the wid in the search order. The wid can be obtained by placing GET for a word defined by WORDLIST .

8.2 Search order

The search order is another list. When FIND and ’ search for name, they take every wid in the search order, starting from the top, and look in the word list for name. They are ready when name matches a name in a word list. It therefore possible to have more words of the same name in different word lists with a different semantics provided the right word is in a word list that is searched earlier. The search order can be set by SET-ORDER and is returned by GET-ORDER . The search order can be extended by ALSO and diminished by PREVIOUS . The list where the compiled words are places is set by SET-CURRENT or DEFINITIONS . The first word list in the order can be set and reset by SET-CONTEXT and GET-CONTEXT respectively.

8.3 CHForth word lists

All Standard words except some that are in ONLY are placed in FORTH . Most extensions are found in EXTRA . In INTERNAL are the words that are not documented and are internally to the system. In EDITOR are the words for ACCEPT and in ASSEMBLER DECOMPILER and DISASSEMBLER are still more words that are used in their specific environment. -1 SET-ORDER will place the ONLY word list in the first and second place of the search order and the count to two. -2 SET-ORDER (not Standard) will extend this with EXTRA and FORTH on the top. The wid returned by GET-CURRENT is not changed.

8.4 Example

CHForth offers a method to find words in a word list that have special properties. For example when you want to know what words are immediate, use this:

```
INTERNAL DEFINITIONS                    \ (IMMED) is internal

ALSO FORTH                              \ New search order

: (IMMED)       ( wid -- )
        DUP BODY> >HEAD ?DUP            \ Has word list a name?
        IF      CR 8 SPACES .HEAD       \ Display it on a new line
        THEN
        CR VOC@ TEMPORARY !             \ Store wid, required for
        BEGIN   ANOTHER                 \    ANOTHER gives a flag
        WHILE   DUP HEAD>FLAGS H@       \    and a dea, check flags
                =IMMEDIATE AND          \ Is word immediate
                IF      ?HEAD           \ Display in 16 char column
                ELSE    DROP            \ Not immediate
                THEN
        REPEAT
    ;
```

```
DEFINITIONS                         \ .IMMEDIATE is FORTH

: .IMMEDIATE    ( -- )
        EVERY?                      \ Typed EVERY before?
        IF      VOC-LINK            \ All word lists
                BEGIN   REGULAR?    \ Only VOCABULARIES
                        IF      DUP (IMMED) \ Do internal word
                        THEN
                        @ ?DUP 0=   \ Every wid done
                UNTIL
        ELSE    GET-CONTEXT (IMMED) \ Only the first in order
        THEN
    ;

PREVIOUS FORTH                      \ Old search order
```

8.5 Word list glossary

.VOCNAME "dot-vocname" EXTRA

(wid --)

Display the name of the word list identification wid.

See also: .HEAD

.WORDLISTS EXTRA

(--)

Display the word lists that have a name, those who have been created with VOCABULARY .

ALSO ONLY

(--)

Transform the search order consisting of wid1 .. widn-1 widn (where widn is searched first) into wid1 .. widn-1 widn widn. An ambiguous condition exists if there are too many word lists in the search order.

ANOTHER EXTRA

(-- dea true | false)

Return the next dea in the word list. Used in words as WORDS . This word depends on the stored wid at TEMPORARY . When ANSI does not contain zero, only words marked with ANS are returned.

ASSEMBLER ASSEMBLER
(--)
Replace the first word list in the search order with the ASSEMBLER word list.

DECOMPILER DECOMPILER
(--)
Replace the first word list in the search order with the DECOMPILER word list.

DEFINITIONS ONLY
(--)
Make the compilation word list the same as the first word list in the search order. Specifies that the names of subsequent definitions will be placed in the compilation word list. Subsequent changes in the search order will not effect the compilation word list.

DISASSEMBLER DISASSEM
(--)
Replace the first word list in the search order with the DISASSEMBLER word list.

EDITOR ONLY
(--)
Make the EDITOR word list the first word list to be searched. This word list contains CHForth specific extensions to the ANSI standard for the line input editor and the block editor. Note that these words are non-standard.

EXTRA ONLY
(--)
Make the EXTRA word list the first word list to be searched. This word list contains all CHForth specific extensions to the ANSI standard. Note that these words are non-standard.

FIND FORTH
(c-addr -- c-addr 0 | xt 1 | xt -1)
Find the Forth word named in the counted string at c-addr. If the word is not found after searching all word list in the search order, return c-addr and zero. If the definition is found, return

xt. If the definition is immediate, also return 1, otherwise return -1.
See also: ' ['] POSTPONE

FORTH FORTH
(--)
Make the FORTH word list the first word list to be searched. Note that this word list contains at startup only ANSI-standard words.

FORTH-WORDLIST ONLY
(-- wid)
Return wid, the identifier of the word list that includes all standard words provided by the implementation. This word list is initially the compilation word list and is part of the initial search order.

GET EXTRA
("name" --)
Interpretation: ("name" -- wid)
Skip leading space delimiters. Parse name delimited by a space. wid is the word list identification associated with name. Exception -32 occurs if name was not defined by VOCABULARY .

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VOCABULARY .

Run-time: (-- wid)
wid is the word list identification associated with name.

GET-CONTEXT ONLY
(-- wid)
Return wid, the identifier of the first word list in the search order.

GET-CURRENT ONLY
(-- wid)
Return wid, the identifier of the compilation word list.

GET-ORDER ONLY
(-- wid1 .. widn n)

Returns the number of word lists n in the search order and the word list identifiers wid1 .. widn identifying these word lists. widn identifies the word list searched first, and wid1 the word list that is searched last. The search order is unaffected.

INTERNAL ONLY
(--)
Make the INTERNAL word list the first word list to be searched. This word list contains CHForth specific extensions to the ANSI standard that are not documented and can be changed by the author by name or action without prior consent. Note that these words are non-standard.

ORDER FORTH
(--)
Display the word lists in the search order in their search order sequence, from the first searched to the last searched. Also display the word list into which new definitions will be placed.

ORDER is implemented using pictured numeric output words. Its use will corrupt the transient region identified by #> .

REGULAR? "regular-query" EXTRA
(wid -- wid flag)
If the word list identification wid has a header (when it was created with VOCABULARY), return a true flag else a false flag.

SEARCH-CONTEXT EXTRA
(c-addr u -- 0 | xt 1 | xt -1)
Find the Forth word specified by the character string c-addr u in all word lists in the search order, including LOCAL-WORDLIST when STATE does not contain zero and there are local values. Return the execution token and 1 if the word is IMMEDIATE and -1 otherwise. If name can not be found, return a false flag. The name is internally converted to uppercase if the variable CASESENSITIVE is false.

SEARCH-WORDLIST FORTH
(c-addr u wid -- 0 | xt 1 | xt -1)
Find the Forth word identified by the string c-addr u in the word list identified by wid. If the word is not found, return zero. If the word is found, return its execution token xt and 1 if the

word is immediate, -1 otherwise.

SET-CONTEXT ONLY

(wid --)

Set the first searched word list in the search order to the word list identified by wid.

SET-CURRENT ONLY

(wid --)

Set the compilation word list to the word list identified by wid.

SET-ORDER ONLY

(wid1 .. widn n --)

Set the search order to the word lists wid1 .. widn. Subsequently, word list widn will be searched first, followed by word list widn-1 and so on, with word list wid1 searched last. If n is zero, empty the search order. If n is minus one, set the search order to the minimum search order wid(ONLY) wid(ONLY). When n is minus two, set the search order to wid(ONLY) wid(EXTRA) wid(FORTH) wid(FORTH). The maximum of n in this implementation is sixteen.

VOC! FORTH

(dea wid --)

Store the dictionary entry address dea in the word list described by the word list identifier wid.

VOC-LINK EXTRA

(-- x)

A value that links all word lists and vocabularies.

VOC@ EXTRA

(wid -- dea)

Fetch the dictionary entry address dea of the last definition from the word list described by the word list identifier wid.

VOCABULARY EXTRA

("name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. Create a new word list and store the word list identifier with the definition for name. name is referred to as a "vocabulary".

name Execution: (--)
Make the above created word list the current word list.

WORDLIST FORTH
(-- wid)
**** Wat wordt er bedoeld met dynamisch ?
Creates a new empty word list, returning its word list identifier wid. The new word list is dynamically allocated in data space. Note that other ANS systems may create the new word list in another place.

WORDS ONLY
(--)
List the word names in the first word list of the search order in columns of 16 characters wide and a count at the end.

WORDS is implemented using pictured numeric output words. Its use will corrupt the transient region identified by #> .
See also: EVERY

WORDSPEED EXTRA
(-- addr)
a-addr is the address of a cell containing the delay after WORDS SEE DIS etc. in milliseconds.

Chapter 9

Vectors

Sometimes some words may require different actions in different situations. CHForth offers a type of word, called a vector. These vectors can be changed and new actions may be appended to the normal action of it.

9.1 Vectors used by the system

KEY and EMIT are for character input and output.

COLD interprets the command line and jumps then to QUIT .

DIAGNOSE will display some information about the compiled memory sizes and the time it took, only when something is given on the command line.

START does some work before COLD is started, can be CHAINed to some other initializing word.

ATEXIT does some work before the program is stopped, like closing an open log file and resetting used interrupt vectors.

'INTERPRET 'COMPILE and NUMBER? have the actions for interpreting and compiling words and numbers in them for INTERPRET .

PROMPT may be changed.

BEEP will not probably change.

PAUSE is in EKEY . Put a word there and Forth will do this every time it waits for a key press.

9.2 Examples

```
: (BEEP)    7 EMIT ;         \ Sound the speaker long and hard
VECTOR BEEP                  \ No action attached yet
' (BEEP) IS BEEP             \ Store the action
BEEP                         \ A sound will be heard
: (TONE)    100 440 TONE ;   \ More pleasant
' (TONE) IS BEEP             \ Another sound
BEEP                         \ A new sound

: (TONE2)
        CHAIN BEEP           \ Inherit the current action: (BEEP)
        100 MS               \ Wait a while
        (BEEP) ;             \ Old sound
' (TONE2) IS BEEP            \ New action appended
BEEP                         \ Two tones

' BEEP IS PAUSE              \ A very irritating sound
' NOOP IS PAUSE              \ Do this to stop the above
```

9.3 Vector words glossary

CHAIN EXTRA

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the current execution semantics of name to the current definition. Exception -32 occurs if name was not defined by VECTOR .

IS EXTRA

Interpretation: (xt "name" --)
Skip leading space delimiters. Parse name delimited by a space. Store execution token xt in name. Exception -32 occurs if name

was not defined by VECTOR .

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VECTOR .

Run-time: (xt --)
Store execution token xt in name.

POP EXTRA

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VALUE , VARIABLE or VECTOR .

Run-time: (--) (R: x --)
Pop x associated with name from the return stack.

PUSH EXTRA

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VALUE , VARIABLE or VECTOR .

Run-time: (--) (R: -- x)
Push x associated with name on the return stack.

VECTOR EXTRA

("name" --)
Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined

below. name is referred to as a "vector".

name Execution: (i*x -- j*x)
Execute the execution token stored in the entry. The execution token can be manipulated by IS . Exception -525 occurs if no execution token is assigned to name.
See also CHAIN POP PUSH

Chapter 10

Interpreter structure

The interpreter is the interface between the user and the Forth program. QUIT asks the user to input a line of text and the interpreter parses words delimited by spaces and executes them, or it will compile them in the dictionary when STATE is not zero.

10.1 QUIT

When after a cold start and perhaps interpreting the contents of the command line, QUIT resets the stacks, displays the status line and displays the prompt on a new line. This prompt will display the name of the first word list in the search order. Because I like to have a explicit prompt, this differs from the Standard where only a flashing cursor is allowed, this can be done by: ' NOOP IS PROMPT Then QUIT waits for input. When the input is followed with a press on the enter key, the word INTERPRET is executed by CATCH . When all went right, the message 'ok' will be displayed when STATE is zero and nothing when it is not. Then the prompt will be displayed and the cycle is complete. When CATCH received a number different from zero, the appropriate error message is displayed and QUIT is reentered.

10.2 INTERPRET

INTERPRET parses a space delimited word and when STATE is zero, the parsed word is passed to 'INTERPRET else to 'COMPILE . When these have completed, the stacks are checked for over- and underflow and the process is repeated. When an exception occurs in 'INTERPRET or 'COMPILE the control is passed to the CATCH in QUIT .

10.3 'INTERPRET

'INTERPRET is a VECTOR and contains $INTERPRET that searches the words in the search order for a match with the name string it gets from INTERPRET . It tries to find the name in the word lists present in the search order and when the found word is not compile-only its execution token will be executed otherwise -14 is THROWn to the CATCH in QUIT . When name can not be found, it tries to convert the string to a number and puts the single or double number on the stack otherwise -13 is THROWn.

When the variable ANSI is not zero, words that do not have the ANS bit set in their headers will give a message just before their execution and numbers that have non-standard prefixes will also display this message.

10.4 'COMPILE

'COMPILE is a VECTOR and contains $COMPILE that searches the words in the search order for a match with the name string it gets from INTERPRET . It tries to find the name in the word lists present in the search order and when the found word is immediate its execution token will be executed otherwise its execution token is compiled in the list segment. When name can not be found, it tries to convert the string to a number and compiles the single or double number in the list segment. When the conversion to a number fails, message -518 is displayed and the name is compiled as a S" string and the word FORWARD is compiled after it.

When the variable ANSI is not zero, words that do not have the ANS bit set in their headers will give a message just before their execution or compilation and numbers that have non-standard prefixes will also display this message.

The strings compiled when message -518 is given are executed by

FORWARD , an alias for EVALUATE , when the word in which the string is compiled, so this can be used as a primitive form as forward referencing. This is not recommended, is main purpose is to continue compiling when a word during compilation can not be found, the programmer can look up the unfound words in the ERROR.LOG file and repair the source code.

10.5 Interpreter words glossary

```
$COMPILE           "string-compile"                          EXTRA
    ( c-addr u -- )
    Try to find the name c-addr u in the search order and when
    found execute it or compile it according to the flag returned
    by FIND . Else try to convert the string to a number and
    compile it.  Else issue a warning that the word can not be
    found and compile a forward reference to it.

$INTERPRET         "string-interpret"                        EXTRA
    ( c-addr u -- )
    Try to find the name c-addr u in the search order and execute
    it when found else convert the string to a number and place it
    on the stack. Else abort with an exception message.

'COMPILE           "tick-compile"                            EXTRA
    ( c-addr u -- )
    A word that normally executes $COMPILE .

'INTERPRET         "tick-interpret"                          EXTRA
    ( c-addr u -- )
    A word that normally executes $INTERPRET .

2LITERAL           "two-literal"                             FORTH
    Interpretation: ( i*x -- )
    This word is marked compile only. The default interpreter issues
    exception -14 when an attempt is made to execute this word.

    Compilation: ( x1 x2 -- )
    Append the run-time semantics defined below to the current
    definition.

    Run-time: ( -- x1 x2 )
    Place cell pair x1 x2 on the stack.
```

ANS EXTRA

(--)

Mark the most recently created definition as a standard word. When the variable ANSI does not contain zero, the default interpreter issues a warning if words that are not marked are interpreted or compiled.

ANSI EXTRA

(-- a-addr)

a-addr is the address of a cell containing true when messages will be given if non-standard words are encountered and false otherwise.

COMPILE, "compile-comma" FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Execution: (xt --)

Append the execution semantics of the definition represented by xt to the execution semantics of the current word definition.

COMPILE-ONLY EXTRA

(--)

Mark the most recently created definition as a compile-only word. The default interpreter issues exception -14 when an attempt is made to execute the definition in interpret state.

EVALUATE FORTH

(i*x c-addr u -- j*x)

Save the current input source specification. Store minus one in SOURCE-ID . Make the string described by c-addr and u both the input source and input buffer, set >IN to zero, and interpret. When the parse area is empty, restore the prior input source specification. Other stack effects are due to the words EVALUATEd.

FORWARD ERRORLOG

(c-addr u --)

Compiled when during loading an undefined word is encountered in a colon definition. As an alias of EVALUATE , it will evaluate a string with the name of the unfound word. This can be used to create forward references.

IMMEDIATE FORTH
(--)
Mark the most recently created definition as an immediate word.

INTERPRET EXTRA
(--)
Interpret the current input stream.

LITERAL FORTH
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (x --)
Compile x as a literal. Append the run-time syntax given below to the current definition.

Run-time: (-- x)
Place x on the stack.

LITERALS EXTRA
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (x1 .. xn n --)
Append the execution semantics defined below to the current definition.

Executing:
(-- x1 .. xn)
Place x1 to xn on the stack.

NOOP "no-op" EXTRA
(--)
Does nothing.

QUERY FORTH
(--)
Make the user input device the input source. Receive input into the terminal input buffer, replacing any previous contents. Make the result, whose address is returned by TIB , the input buffer.

Set >IN to zero.

Note: this word is obsolescent and is included as a concession to existing implementations.

QUIT FORTH

(--)

Empty the return stack, store zero in SOURCE-ID , make the user input device the input source, and enter interpretation state. Do not display a message. Repeat the following:

- Accept a line forth the input source into the input buffer, set >IN to zero and interpret.
- Display the implementation defined input prompt if in interpretation state, all processing has been completed, and no ambiguous condition exists.

STATE

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(-- a-addr)

a-addr is the address of a cell containing the compilation state flag. STATE is true when in compilation state, false otherwise. The true value in STATE is non-zero, but is otherwise implementation-defined. Only the following standard words alter the value in STATE : : (colon), ; (semicolon), ABORT , QUIT , :NONAME , [(left-bracket),] (right-bracket) and ;CODE .

Note: A Standard Program may not directly alter the contents of STATE .

See also: : :NONAME ; ABORT QUIT [ ]

TERMINAL EXTRA

(--)

Reset the input and output to the terminal.

Chapter 11

Error recovery

The ANSI Standard offers a consistent way for error recovery. This is done by a non-local return mechanism implemented by the words CATCH and THROW just as in LISP and the words SETJMP en LONGJMP in C.

CHForth offers also a way to define your own exception messages and redefine existing messages.

Newer books will use the word exception instead of error because it is more general and sometimes no error did occur at all.

11.1 CATCH and THROW

No word in CHForth handles its own exceptions, but the words in which an exception may occur all return a value on the stack, that differs from zero when an exception occured and is zero when nothing went wrong. Then by placing the word THROW next to the previous word, a zero on the stack will be dropped and the program will continue, but a non-zero number will look for a so-called exception frame on the return stack. This frame is pushed there by the word CATCH . The depth of the data stack before CATCH was called is restored as is the depth of the return stack. The exception number will be on the stack and execution will continue after CATCH instead after THROW . The user than can take measures when a certain exception number will appear on the stack. When no exception occured in a word that was executed by CATCH, a zero will be left on top of the stack.

In QUIT is also a CATCH . This CATCH word catches all exceptions it receives of INTERPRET , this word reads a line from the terminal and tries to interpret this. Any errors are displayed on the screen with the error number, the type (is this a standard message, a message of DOS or a message of CHForth that is not standard) and the message that is assigned to this number with the word MESS" and then follows the line where the exception occurred. When this exception occured during loading of a text file or a blocks file, this information is also written to a text file called error.log, so you can see where the errors occurred at a more convenient time.

11.2 Examples

```
: read-key
        KEY [CHAR] Q = THROW    \ Error when 'Q' pressed
        CR ." OK "              \ No error, CATCH leaves 0
    ;

: main
        CR ." Press 'Q' to stop"
        BEGIN
        ['] read-key CATCH      \ Try to catch an error
        UNTIL                   \ 'Q' stops, others continue
    ;

-13 MESS" unbekanntes Wort"     \ Redefine a message

' UNSINN                        \ May give the new message
```

11.3 Error messages

These messages are given when you type the number followed by .MESS . The address of the counted string can now be found in ERR$ and the number in ERR# . Beware that these become invalid when another exception occurs. In QUIT these are the messages that are given when any exception occurs. You can generate new messages for these numbers with MESS" . In this way you could for example make Dutch messages.

11.3.1 Standard ANS Forth messages

-3	stack overflow
-4	stack underflow
-5	return stack overflow
-6	return stack underflow
-8	dictionary overflow
-9	invalid memory address
-10	division by zero
-11	result out of range
-13	undefined word
-14	interpreting a compile-only word
-15	invalid FORGET
-16	attempt to use zero-length string as a name
-22	control structure mismatch
-25	return stack imbalance
-28	user interrupt
-29	compiler nesting
-32	invalid name argument
-33	block read exception
-34	block write exception
-35	invalid block number
-36	invalid file position
-37	file I/O exception
-38	non-existent file
-49	search-order overflow
-50	search-order underflow
-57	exception in sending or receiving a character
-58	missing terminating [ELSE] or [THEN]

11.3.2 DOS messages

First number is the exception number, second the standard DOS error number.

-511	1	function number invalid
-510	2	file not found
-509	3	path not found
-508	4	too many open files (no handles available)

-507	5	access denied
-506	6	invalid handle
-505	7	memory control blocks destroyed
-504	8	insufficient memory
-503	9	memory block address invalid
-502	10	environment invalid
-501	11	format invalid
-500	12	access code invalid
-499	13	data invalid
-497	15	invalid drive
-496	16	attempted to remove current directory
-495	17	not same device
-494	18	no more files
-493	19	disk write-protected
-492	20	unknown unit
-491	21	drive not ready
-490	22	unknown command
-489	23	data error (CRC)
-488	24	bad request structure length
-487	25	seek error
-486	26	unknown media type (non-DOS disk)
-485	27	sector not found
-484	28	printer out of paper
-483	29	write fault
-482	30	read fault
-481	31	general failure
-480	32	sharing violation
-479	33	locking violation
-478	34	disk change invalid
-477	35	FCB unavailable
-476	36	sharing buffer overflow
-451	61	print queue full
-450	62	queue not full
-449	63	not enough space to print file
-432	80	file exists
-430	82	cannot make directory
-429	83	fail on INT 24h
-428	84	too many redirections
-427	85	duplicate redirections
-426	86	invalid password
-425	87	invalid parameter

11.3.3 Messages of this Forth system

```
 -513 is not unique
 -514 execution halted
 -515 wrong use of DPSWAP
 -516 no defining word
 -517 not defining methods
 -518 is undefined, compiling forward reference
 -519 list-segment full
 -520 header-segment full
 -521 program contains errors
 -522 local stack overflow
 -523 local stack underflow
 -524 illegal opcode for this processor
 -525 unresolved forward definition
 -526 no special routine for this character
 -527 is not portable
 -528 or part of it is not yet implemented
 -529 is in a non-portable number format
 -530 already defining methods
 -531 character can not be converted
 -532 missing terminating ENDDOC
 -533 missing terminating *)
```

Message 0 will display the copyright message.

11.4 Error words glossary

!CSP "store-c-s-p" EXTRA

(--)

Save the current depth of the stack for checking with ?CSP .

.MESS EXTRA

(n --)

Display the message that is assigned to exception number n as with MESS" . If the message is not found, display the exception number and the name of the word where the exception occurred. If n is -1 or -2 nothing is displayed. Store the number in ERR# .

.WHERE EXTRA
(--)
If the last exception occurred during loading of a file, display the name of the file and the line number where the exception occurred.

?CSP "question-c-s-p" EXTRA
(--)
Check the current depth of the stack with the one stored by !CSP Exception -29 will occur when they do not match.

?ERROR "question-error" EXTRA
(x n --)
If x is not zero, exception n occurs. Else drop both numbers from the stack and continue.

?PAIRS "question-pairs" EXTRA
(x1 x2 --)
Check x1 and x2. Exception -22 occurs when they are not equal.

?STACK "question-stack" EXTRA
(--)
Check the three stack pointers and when they are too low or too high, exception -3, -4, -5, -6, -522 or -523 will occur.

ABORT FORTH
(i*x --) (R: j*x --)
Perform the function of -1 THROW . When no other exception frame is present other than the one pushed by QUIT , empty the stacks and perform QUIT . When no file is currently open, display no message. Otherwise, contrary to the Standard, display some information about the file and the line where ABORT was called. Store a zero-length string in ERR$.

ABORT" "abort-quote" FORTH
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("ccc<quote>" --)
Parse characters ccc delimited by " (double-quote). Append the run-time semantics specified below to the current definition.

Run-time: (i*x x1 -- | i*x) (R: j*x -- | j*x)
Remove x1 from the stack. If any bit of x1 is not zero, perform the function of -2 THROW . The default interpreter will display ccc. The address of the counted string ccc can be found in ERR$, but is only valid for a limited time.

CATCH FORTH
(i*x xt -- j*x 0 | i*x n)
Push an exception frame on the exception stack and then execute the execution token xt (as with EXECUTE) in such a way that control can be transferred to a point just after CATCH if THROW is executed during the execution of xt.

If the execution of xt completes normally (i.e. the exception frame pushed by this CATCH is not popped by an execution of THROW) pop the execution frame and return zero on top of the data stack, above whatever stack items would have been returned by xt EXECUTE . Otherwise, the remainder of the execution semantics are given by THROW .

ERR# "error-number" EXTRA
(-- x)
Return the number of the last exception.

ERR$ "error-string" EXTRA
(-- c-addr)
Return the address of the count of the last exception string.

ERRLINE "error-line" EXTRA
(-- a-addr)
a-addr is the address of a cell containing the line number of the file where an exception occurred.

ERRNAME "error-name" EXTRA
(-- a-addr)
a-addr is the address of a cell containing the address of the counted string representing the name of the file where an exception occurred.

ERROR-TYPE EXTRA
(--)
Show the type of the last exception number stored in ERR# by .MESS . Display nothing if ERR# equals -1 or -2.

```
MESS"               "mess-quote"                          EXTRA
    ( n "ccc<quote>" -- )
    Parse ccc delimited by a " (double-quote) and compile the string
    in the dictionary. The string is displayed when n is passed to
    .MESS or THROW .

NOT-IMPLEMENTED                                           EXTRA
    ( -- )
    Abort with exception message: not implemented, used in some
    definitions.

SHOW-ERROR                                                EXTRA
    ( n -- )
    Display the exception message and information where the exception
    with number n occurred and the type of the exception and display
    the source line with the exception word marked out.

THROW                                                     FORTH
    ( k*x n -- k*x | i*x n )
    If any bits of n are non-zero, pop the topmost exception frame
    from the exception stack, along with everything on the return
    stack above that frame. Then restore the input source
    specification in use before the corresponding CATCH and adjust
    the depths of all three stacks so that they are the same as the
    depth saved in the exception frame (i is the same number as i in
    the input arguments to the corresponding CATCH ), put n on top of
    the data stack, and transfer control to a point just after the
    CATCH that pushed that exception frame.

WARNING                                                   EXTRA
    ( -- a-addr )
    a-addr is the address of a cell containing true when the program
    will warn the user when redefinitions are encountered and false
    otherwise.
```

Chapter 12

The assembler

The assembler, if not already in memory, can be loaded by

```
    NEEDS -assembler
```

This is a full 8086 assembler that works in prefix mode about the same as the one in F-PC, that is opcodes precede the destiny and the source, different from traditional postfix Forth assembler like the one in F83.

When using CHF386 the assembler is extended with some instructions known only to 386 and 486 CPU's like 32 bit data manipulation, although no 32 bit addressing, as CHForth runs in real or virtual 8086 mode.

12.1 Register use

The BX register is always contains the top element of the stack, the other elements are addressed via SP. The return stack is addressed via BP. Both stacks reside in the stack segment, the value of it is in the register SS. The local stack is only accessible by system words.

When entering a code definition, the AX register contains its starting address, that is useful if you make defining words with ;CODE. The code segment, CS is equal to DS.

The list segment with the colon definitions is kept in ES and the offset of the Forth instruction pointer in SI.

The direction bit must be clear when returning to Forth, when you use STD always do CLD at the end of your code.

The use of CX, DX and DI is not restricted.

When the destiny is a register or a register indirect mode, always append a comma to it.

12.2 Examples

```
CODE D+         ( d1 d2 -- d3 )
                POP     CX               \ pop low word of d2
                POP     DX               \ pop high word of d1
                POP     AX               \ pop low word of d1
                ADD     AX, CX           \ add low part
                ADC     BX, DX           \ add high part
                PUSH    AX               \ push low part
                NEXT                     \ high part in BX
END-CODE                                 \ check errors
```

Remember that the 80x86 series are low end machines and ANS Forth is on word-level a high end machine: the word order is the other way round. As it happens, 2@ of an Intel pointer places the offset on top, as is necessary in -x words, so many problems do not arise in high level words. When you push or pop 32 bit data, remember to swap both halves.

The assembler for the 386 version can generate code for 32 bit data, but not for 32 bit addresses as it runs in real or virtual 86 mode. So the previous example can be rewritten as:

```
CODE D+         ( d1 d2 -- d3 )
                ROL     EBX, # #16       \ shift high word left
                POP     BX               \ pop low word
                POP     EAX              \ pop 32 bit operand
                ROL     EAX, # #16       \ swap high and low words
                ADD     EBX, EAX         \ do the operation
                PUSH    BX               \ push low word
                ROL     EBX, # #16       \ high word to BX
                NEXT                     \ return to Forth
END-CODE                                 \ check errors
```

Some addressing modes:

```
Intel:                  CHForth:

pop     ax              pop     ax
pop     [BX]            pop     0 [bx]      \ always an offset
pop     [bx+23]         pop     23 [bx]
pop     es:[bp+si-32]   pop     es: -32 [bp+si]
push    [1234]          push    1234        \ direct address
mov     ax,cs:[bx]      mov     ax, cs: 0 [bx]
mov     cs:[bx],ax      mov     cs: 0 [bx], ax
mov     ax,[1234]       mov     ax, 1234    \ direct address
mov     ax,1234         mov     ax, # 1234  \ immmediate
mov     [1234],ax       mov     1234 ax     \ direct address
mov     al,12           mov     al, # 12
mov     al,[1234]       mov     al, 1234 [] \ direct address
mov     [1234],al       mov     1234 [], al \ byte adressing
mov     byte ptr 12,'A' mov     12 [], # 'A' byte
                                    \ [], # and byte all required
mov     word ptr 12,21  mov     12 # 21
add     bx,[bx]         add     bx, 0 [bx]
mov     al,12           mov     al, # 12    \ immediate mode
sar     bx              sar     bx, # 1     \ # 1 is necessary
out     dx,al           out     dx, al
out     12,ax           out     12 #, ax    \ only use for #,
in      ax,dx           in      ax, dx
in      al,12           in      al, # 12
```

Only when #CPU contains #386 :

```
Intel:                  CHForth:

push    1234            push    # 1234      \ immediate
push    ebx             push    ebx         \ 32 bit data
pop     dword [ebx]     pop     sz: 0 [bx]  \ 16b adr, 32b data
pop     [ebx]           not implemented     \ 32b address
rcl     ebx,14          rcl     ebx, # 14   \ only 386
```

12.3 Structures

Like the compiler structures in high level Forth, the assembler also knows control flow structures. For example:

```
CODE ?DUP               ( x -- x x | x )
                TEST    BX, BX  \ When top of stack is not zero
        0<> IF
                PUSH    BX      \ Push it on the stack
        THEN
                NEXT            \ Top of stack still in BX
END-CODE
```

As you see, the jump words like IF, WHILE, UNTIL are postfix, the condition like 0= 0<> U< come before them. Without IF and THEN the example can be rewritten using labels:

```
CODE ?DUP               ( x -- x x | x )
                TEST    BX, BX  \ When top of stack is zero
                JZ      0 $     \ jump to label
                PUSH    BX      \ Push it on the stack
        0 $:    NEXT            \ Top of stack still in BX
END-CODE
```

The labels consist of $: preceded with a number in the range 0..31 and the jumps have the corresponding number followed by $.

When using the 386 version, the conditional jumps like JZ will compile a 16 bit LJZ when the offset is too large for a signed byte. You can not use the labels with 'n $' for this, use real addresses for example:

```
                JAE     $1223
                JL      $4504
```

In the 86 version you have be more inventive, like this:

```
        U>= IF
                JMP     $1223
        THEN
                JGE     4 $
                JMP     $4504
        4 $:
```

When you need to use words like , in a code definition you have to put a A; before using it because the assembler will execute opcodes after all of its parameters are defined and so the , will interfere. A better method to compile data is using words DW and DB that handle this problem.

```
CODE SWAP
            DB  $58              \ POP AX
            DB  'S'              \ PUSH BX
            DW  $D88B            \ XCHG BX, AX
            NEXT
            A; ", finished!"     \ compile an inline counted string
END-CODE
```

Use JB for JNAE (or JC) and JGE for JL etcetera, as I have deleted the JNx words.

12.4 Assembler words glossary

`#` ASSEMBLER

(x -- x)
Immediate mode for source.

`#,` ASSEMBLER

(x -- x)
Immediate mode for destiny.

`$` ASSEMBLER

(x --)
Jump to an assembler label.

`$:` ASSEMBLER

(x --)
Define an assembler label.

`$ELSE` ASSEMBLER

(--)
Jump to after $THEN .

`$IF386` ASSEMBLER

(--)
If #CPU does not contain 386 jump to after $ELSE or $THEN .
Else continue.

`$THEN` ASSEMBLER

(--)
Terminate a $IF386 directive.

;CODE ASSEMBLER

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: colon-sys --)
Append the execution semantics defined below to the current definition. End the current definition, consuming colon-sys, enter interpret state, add the ASSEMBLER word list to the search order and start interpreting the rest of the parse area and assemble machine code. If needed, refill the input buffer until END-CODE is processed.

Execution: (--) (R: nest-sys --)
Replace the execution semantics of the most recently defined word with the name execution semantics given below. Return control to the calling definition specified by nest-sys. An ambiguous condition exists if the most recently defined word was not defined with CREATE or a user-defined word that calls CREATE .

name Execution: (i*x -- j*x)
Perform the machine code sequence that was generated following ;CODE .
See also: DOERCODE DOES> END-CODE

A; ASSEMBLER

(--)
Terminate a line of assembly code.

ASSEMBLER ASSEMBLER

(--)
Replace the first word list in the search order with the ASSEMBLER word list.

CODE ASSEMBLER

("name" --)
Skip leading space delimiters. Parse name delimited by a space. Create a definition for name, called a "code definition", with the execution semantics defined below. Add the ASSEMBLER word list to the search order and start interpreting the rest of the parse area and assemble machine code. If needed, refill the input buffer until END-CODE is processed.

name Execution: (i*x -- j*x)
Execute the machine code sequence that was generated following CODE .
See also: END-CODE

DB ASSEMBLER
("ccc" --)
Assemble "ccc" as an 8 bit value.

DOERCODE ASSEMBLER
("name" --)
Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. Enter interpret state, add the ASSEMBLER word list to the search order and start interpreting the rest of the parse area and assemble machine code. If needed, refill the input buffer until END-CODE is processed.

Execution: (--) (R: nest-sys --)
Replace the execution semantics of the most recently defined word with the name execution semantics given below. Return control to the calling definition specified by nest-sys. An ambiguous condition exists if the most recently defined word was not defined with CREATE or a user-defined word that calls CREATE .

name Execution: (i*x -- j*x)
Perform the machine code sequence that was generated following DOERCODE .
See also: DOES> END-CODE

DW ASSEMBLER
("ccc" --)
Assemble "ccc" as a 16 bit value.

END-CODE ASSEMBLER
(--)
Resolve all assembler labels, terminate the current code definition and allow its name to be found in the dictionary. Remove the ASSEMBLER word list from the search order.

L$ ASSEMBLER
(-- addr)
Define a forward near label in assembler, one per definition.

```
L$:                                                                 ASSEMBLER
    ( addr -- )
    Resolve a forward near label.

[]                                                                  ASSEMBLER
    ( -- )
    Direct mode for source.

[],                                                                 ASSEMBLER
    ( -- )
    Direct mode for destiny.
```

Chapter 13

FLYER

FLYER is a way to further eliminate the implicit use of STATE in Forth. Most words that have a different action during compilation and interpretation, can now be written with the compile-time action only. Flyer is a new concept based on a technique called a co-routine. This co-routine technique is supplied by the word DIVE . It performs a switch to compiling and execution when a word like S" is used at runtime, compile time it is a noop (does nothing). All words which need a well defined action while interpreting use the word FLYER . So there is one well defined way to define these actions and the programmer needs only to write its compilation behaviour. In CHForth it is used throughout the system to give all words an interpretation behavior, with undefined interpretation semantics according the standard.

13.1 Compilation in a buffer

Words that internally use FLYER like TO and S" are compiled in a reserved buffer during interpreting. This buffer is located at the directly above the dictionary space. The default size of this buffer is 512 bytes, plus 140 bytes overshoot space. The word DPSWAP switches between user dictionary and this buffer space.

13.2 The circular buffer

The buffer is made circular, so it can be used over and over again. It will never overflow, but data which is kept there has a restricted lifetime. The word CIRCULATE keeps the buffer within its given bounds and overshoot space. The size of the buffer may be adjusted with the word FLY-BUFFER , which may result in the loss of all previous buffer data. The size of this buffer may vary from 128 to 4096 bytes.

13.3 DIVE into deep water

The co-routine call DIVE swaps two (return) addresses on the top of the return stack. Its behaviour is not so easy to understand, to understand the order of execution consider the following example:

```
: DIVING
    ." TWO "                     ( This is displayed second )
    DIVE                         ( Swap return stack addresses )
    ." FOUR " ;                  ( So this is displayed as last )

: DIVE-DEMO
    ." ONE "                     ( This is displayed first )
    DIVING                       ( Display 'TWO' and return... )
    ." THREE " ;                 ( This is displayed as third )
                                 ( And afterwards comes FOUR )
```

Than a small riddle, what is happening here ?

```
: HI        CR ." Hi "  DIVE  ." how are you? " ;  ( -- )
: GREET     HI  BL PARSE TYPE SPACE ;          ( "name" -- )
```

Describe what this code should do when GREET is executed and test the example on your system. The same mechanism is used for FLYER and for implementing local variables.

Lets implement a simple tracer:

```
: TRACE
    CR ." before" .S             ( Print stack before running )
    DIVE                         ( Back to calling routine )
    ." after" .S ;               ( Print stack after running )

: :     ( "name" -- )            ( Redefine colon to include )
    :  POSTPONE TRACE ;          ( the tracer )
```

Whenever we have any doubts about some words stack behaviour, we can use this redefinition of colon to check a words stack behaviour. One can add any features to improve TRACE ; be aware of the weird return stack behaviour, which should not interfere with handling return stack data (E.g. inline arguments) as the return address of TRACE may reside on top of such data.

13.4 Use of FLYER

Where can we use FLYER in our code? The answer is, anywhere we need the same compile and a runtime behaviour of a compiler directive. Note: The only word added to the next definition, to make it available when interpreting is the word FLYER. System words which include FLYER are:

```
    S"  ."  ABORT"  PREFIX  (and thereby all prefixes)
```

An example from the Forth source:

```
: ."    ( "ccc"<"> -- )
        FLYER                   ( Fly when interpreting     )
        POSTPONE (.")           ( Compile runtime code      )
        [CHAR] " PARSE,         ( Get & compile string      )
        ;  IMMEDIATE
```

13.5 FLYER words glossary

CIRCULATE EXTRA

(--)

If the buffer overflows, reset it to the start of the FLYER buffer space.

DIVE EXTRA

(--)

Perform a co-routine call to the calling routine. This means that the calling routine is finished first. If the calling routine is finished the called routine which included DIVE will be finished.

DPSWAP "d-p-swap" EXTRA

(--)

Exchange the dictionary pointer from the user dictionary to the

FLYER buffer.

FLYER EXTRA

(i*x -- j*x)

When interpreting, switch to the circular buffer. Set system to compilation state and execute a coroutine call to the calling routine. Compile this routine followed by an EXIT. Switch back to executing state and user dictionary, next execute the compiled routine. Only for use in a definition.

Chapter 14

Create new data types

This system offers two different ways of defining new data types. The new way described here builds named "execution interpreters". The idea is not new, the standard document, ANSI X3.215-1994 contains an example (page 176) where this concept is used. However the standard does not support it as a named datatype. This concept separates the definition of compilers and executable code. It makes the code easier to understand and the scope in an execution part is now clear.

14.1 Introduction to DOER:

The standard method for defining new data types mix the creation (compiler action) and the execution action for a new data type in a single definition.
Conceptual DOER: is a separation from the creation of a compiler for a new datatype and the execution code (interpreter) for that type. See for an example paragraph 18.3 just below.

What happens when we use the new defining word DOER: ?

- A header is created, with the name mentioned just behind DOER:
- A reference to MODIFY is compiled.
- A type data field is created.
- A call to DODOES is compiled.
- The compiler (colon) is started.

Actually a DOER: word combines : (colon) and DOES> . On its execution MODIFY replaces the execution behavior of the most recently created definition with the specified action (execution-interpreter).

14.2 Supplied words

```
    DOER:  DOERCODE  DOES>  ;CODE
```

14.3 A comparison of DOER: and DOES>

We will make a new defining word using both methods. First in the traditional way:

```
: NEWVAL
    CREATE  ,   ( x -- )    ( Create a new name and reserve data space )
    DOES> @ ;   ( -- x )    ( Push contents of data field )
```

Now with the new DOER:

```
DOER: DONEWVAL  ( -- x )    ( Push contents of data field )
    @ ;

: NEWVAL        ( x -- )
    CREATE  ,  DONEWVAL ;   ( Create a new name and reserve data space )
```

Both words create the same data type. When using DOER: there is a strict separation between the "execution-interpreter" and the build action of the data structure. There are more advantages, showed in the next example:

```
: <New-data-type>
    ( Actions executed before a header is created              )
    CREATE  ( Build empty data structure with default action   )
    ( Actions to allocate/initialise the structures data field )
    DO-new-data-type ( Replace the execution time action        )
    ( Actions required after the new datatype is installed     )
    ;
```

We get new options here, it is possible to describe in a conventional way, how the creation of a new datatype ends.

14.4 The use of DOERCODE and ;CODE

The words DOERCODE and ;CODE are the assembly counterparts of DOER: and DOES> and are used in the same way. The difference is, that the specification of the execution action is written in machine code. The programmer should take care to obtain the data address explicitly. Note: The extra indirection when creating code that runs in ROM whilst the data is in RAM. Then an extra indirection to the RAM data field is required.
This problem is solved by the assembler macro DATA-ADDR,

An example of the use:

```
: NEWVAL
    CREATE  ,   ( x -- )
    ;CODE       ( -- x )
        [R2] POP,          ( Pop data address high byte       )
        ACC: POP,          ( Pop data address low byte        )
        R0: DEC,           ( Allocate low byte on stack       )
        @R0 A: MOVX,       ( Move low byte to stack           )
        A: R2: MOV,        ( Get high address byte            )
        R0: DEC,           ( Allocate high byte on stack      )
        @R0 A: MOVX,       ( Move high byte to stack          )
        RET,               ( Ready                            )
    END-CODE

DOERCODE DONEWVAL
     [R2] POP,             ( Pop data address high byte       )
     ACC: POP,             ( Pop data address low byte        )
     R0: DEC,              ( Allocate low byte on stack       )
     @R0 A: MOVX,          ( Move low byte to stack           )
     A: R2: MOV,           ( Get high address byte            )
     R0: DEC,              ( Allocate high byte on stack      )
     @R0 A: MOVX,          ( Move high byte to stack          )
     RET,                  ( Ready                            )
END-CODE

: NEWVAL        ( x -- )
    CREATE  ,  DONEWVAL ;  ( Create a new name and reserve data space )
```

There is no difference between those words and the high level versions, except for speed.

14.5 Using prefix operators

Most data types can make use of prefix operators. They do this by building methods (prefix actions) in a so called methods word list. The words DOER: DOERCODE DOES> and ;CODE create such a word list, also called type-data-field. With the words METHODS and INHERIT these word lists can be handled. Lets make NEWVAL from the previous chapter accessible by the prefixes of VALUE :

```
INHERIT DOVAL DONEWVAL
```

All prefixes available to DOVAL (VALUE) are now available to NEWVAL . For detailed information read chapter 19.6 .

14.6 Defining words word glossary

: "colon" FORTH

(C: "name" -- colon-sys)
Skip leading delimiters. Parse "name" delimited by a space. Create a definition for name, called a "colon definition". Enter compilation state, and start current definition, producing colon-sys. Append the initiation semantics below to the current definition. The execution semantics of name will be determined by the words compiled into the body of the definition. The current definition is not findable in the dictionary until it is ended. When used within the METHODS structure, the behaviour of this word will be changed!

Initiation: (i*x -- i*x) (R: -- nest-sys)
Save nest-sys (a single cell address) of the calling definition. The stack effects i*x represents arguments to name.

name Execution: (i*x -- j*x)
Execute the definition name. The stack effects i*x and j*x represent arguments to and results from name, respectively.
See also: NONAME: DOES> ; ;CODE] [

:NONAME "colon-noname" FORTH

(C: -- colon-sys || S: -- xt)
Create an execution token xt, enter compilation state and start the current definition, producing colon-sys. Append the initiation semantics given below to the current definition. The execution semantics of xt will be determined by the word compiled into the body of the definition. This definition can be executed later by using xt EXECUTE .
Colon-sys is the topmost item on the data stack.

Initiation: (i*x -- i*x) (R: -- nest-sys)
Save nest-sys (a single cell address) of the calling definition. The stack effects i*x represent arguments to xt.

xt Execution: (i*x -- j*x)
Execute the definition specified by xt. The stack effects i*x and j*x represents argument to and results from xt, respectively.
See also: : DOES> ; ;CODE] [

CODE FORTH

("name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name, called a "code definition", with the execution semantics defined below. Append the ASSEMBLER word list to beginning of the search order to process the words between name and END-CODE. sys is balanced by the corresponding END-CODE. name is called a "code definition."

name Execution: (i*x -- j*x)

Execute the machine code sequence that was generated following code.

CONSTANT FORTH

(x "name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. name is referred to as a "constant".

name Execution: (-- x)

Place x on the stack.

CREATE FORTH

("name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. If the data-space pointer is not aligned, reserve enough data space to align it. The new data-space pointer defines name's data field. CREATE does not allocate data space in name's data field. The words ROM and RAM change the behaviour of this word!

name Execution: (-- a-addr)

a-addr is the address of name's data field. The execution semantics of name may be extended by using DOES> .
The relocatable compiling process will compile an extra indirection when building a defining word with data in ram.

DOER: "doer-colon" EXTRA

("name" -- || C: -- colon-sys)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. Enter compilation state, and start current definition.
The relocatable compiling proces will compile an extra indirection when building a defining word with data in ram.

Runtime: (--) (R: nest-sys1 --)
Replace the execution semantics of the most recent definition, referred to as name, with the name execution semantics given below. Return control to the calling definition specified by nest-sys1. Code may be damaged if the most recently defined word was not defined with CREATE or a user-defined word that calls CREATE . The words ROM and RAM change the behaviour of this word!

Initiation: (i*x -- i*x a-addr) (R: -- nest-sys2)
Save implementation-dependant information nest-sys2 about the calling definition. Place name's data field address on the stack. the stack effects i*x represents the arguments to name.

name Execution: (i*x -- j*x)
Execute the part of the definition beginning with the initiation semantics appended by the DOES> which modifies name. The stack effects i*x and j*x represent arguments to and result from name, respectively.
See also: CREATE DOES>

DOERCODE "doer-code" EXTRA
("name" --)
Parse name delimited by a space. Create a definition for "name" with the execution semantics defined below. Append the ASSEMBLER word list to the beginning of the search order.
DOERCODE is balanced by the corresponding END-CODE .

Runtime: (--) (R: nest-sys --)
Replace the execution semantics of the most recent definition.

name Execution (i*x -- j*x)
Perform the machine code sequence generated following DOERCODE.
This word does not respond at the words ROM and RAM.
See also DOES> ;CODE ROM RAM .

FLAG EXTRA
("name" --) (-- flag)
Define a bit flag with name. An error will be issued when there are no more bits left in the bit array. The programmer may define a maximum of 120 bit flags.
name Execution: (-- flag)
Expand the contents of the bit flag, leaving true or false.

See also: SET , CLEAR or TO .

LOCAL EXTRA

("name" --)
Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution and run-time semantics defined below.

Execution: (x --)
Store x in name.

name Execution: (-- x)
Place x on the stack. The value can be manipulated by TO +TO .

LOCALS| "locals-bar" FORTH

("name1" .. "namen" "|" --)
Create up to eight local variables with "name1" to "namen". The list of locals is terminated by "|". In CHForth this is not limited to eight locals, it depends on the actual name length and the size of the FLYER buffer.

Runtime: (xn .. x2 x1 --)
Initialise up to 8 local variables, each of which takes as its initial value the top stack item, removing it from the stack. Identifier name1 is initialised with x1, etc. When invoked each local will return its value. The value may be changed using TO and +TO .

MARKER FORTH

("name" --) (--)
Skip leading space delimiters. Parse name delimited by a space. Create a dictionary entry for name with the execution semantics defined below.
name Execution: (--)
Restore all dictionary allocation and search order pointers to the state they had just prior to the definition of name. Remove name and all subsequent word definitions. Restoration of any structures still existing that could refer to deleted definitions or deallocated data space is not provided in any other way then by the use of forget fields. No other contextual information such as numeric base is affected. See also: FORGET (FORGET) IS-FORGET

PREFIX EXTRA

```
    ( "name1" -- )
    Skip leading space delimiters. Parse name1 delimited by a space.
    Create a definition for name1 with the execution semantics defined
    below.

    name Execution: ( i*x "name2" -- j*x )
    Skip leading space delimiters. Parse name2 delimited by a space.
    Execute the prefix action of name1. Error -64 will be issued
    if this prefix is not valid for this word or datatype.

    name Compilation: ( i*x "name2" -- j*x )
    Skip leading space delimiters. Parse name2 delimited by a space.
    Compile the prefix action of name1. Error -64 will be issued
    if this prefix is not valid for this word or datatype.

SFR              "s-f-r"                               EXTRA
    ( byte "name" -- )
    Create a new definition with name and address byte, with the
    execution semantics defined below.
    name Executing: ( -- byte )
    Push the contents of the SFR or 'direct ram' location addressed
    by "name" <byte> on the stack.

VALUE                                                  FORTH
    ( x "name" -- )
    Skip leading space delimiters. Parse name delimited by a space.
    Create a definition for name with the execution semantics defined
    below, with an initial value equal to x. name is referred to as a
    "value". The words ROM and RAM change the behaviour of this word!

    name Execution: ( -- x )
    Place x on the stack. The value of x is that given when name was
    created, until the phrase x TO name is executed, causing a new
    value of x to be associated with name.
    See also +TO CLEAR ADR PUSH POP

VARIABLE                                               FORTH
    ( "name" -- )
    Skip leading space delimiters. Parse name delimited by a space.
    Create a definition for name with the execution semantics defined
    below. Reserve one cell of data space at an aligned address.
    name is referred to as a "variable". The words ROM and RAM change
    the behaviour of this word!
```

name Execution: (-- a-addr)
a-addr is the address of the reserved cell. A program is responsible for initialising the contents of the reserved cell.

VOCABULARY ONLY

("name" --)
Parse name delimited by a space, ignoring leading delimiters. Create a dictionary entry for name with the execution semantics defined below. Create a new word list and store the word list identifier with the new word.
name is referred to as a "vocabulary".
name Execution: (--)
Make the above created word list the context word list.

14.7 Internal structure of the basic do-types

a) The basic structure of DOER: and DOES> . When created with DOER: a header is in front of this structure. But when created by DOES> , the structure is placed in the middle of a colon definition after the described compiler action.

```
    ------------------------------------------------
    |MODIFY|forget-xt|toplfa-ptr|toplfa|DODOES|Etc.|
    ------------------------------------------------
                            \______^
```

b) The basic structure of DOERCODE and ;CODE . When created with DOERCODE a header is in front of this structure. But when created by ;CODE , the structure is placed in the middle of a colon definition after the described compiler action.

```
    --------------------------------------------------
    |MODIFY|forget-xt|toplfa-ptr|toplfa|assembly-code|
    --------------------------------------------------
                            \______^
```

Chapter 15

The TO-concept

The TO-concept is a method of accessing data and was originally developed by Paul Bartholdi (Forth Dimensions Vol 1 nr 4). All data types have the same access operator (TO etc), there is no longer a need for programmers to remember which operator belongs to a certain data type for instance:

```
2VARIABLE AAP       ( Operators for AAP : 2@ 2! 2+! )
VARIABLE DOG        ( Operators for DOG : @ ! +! )
CVARIABLE ANT       ( Operators for ANT : C@ C! C+! )
```

When we use the TO-concept all these data types are accessed with the same words (FROM TO +TO). So the programmer no longer needs to know all the different operators. Other advantages of using the TO-concept are, it is less easy to use the wrong data type and operator. Use ! on the byte variable ANT is no longer possible and the program is much more easy to read without those @ and ! operators.

15.1 How do prefixes work

A prefix operator scans the input for the next word in the input stream. If any, it checks if it is a valid prefix for that word. So prefix actions will only be accepted when they are valid for the used data type. If a prefix is not valid, an error message is given (ANSI message -32). Some examples:

```
8 TO BASE           ( Set number base to octal )
FROM BASE .DEC      ( Get number base and print it in decimal )
```

```
BASE .HEX          ( Print the address of BASE in hexadecimal )
```

The system variable BASE is used here, according to ANSI it must give its address on the stack when executed. In CHForth, these words are of the type 'system VARIABLE'. A prefix can change the behaviour of such a variable. All used system variables are of the following types:

system VALUE   : DP VOC-LINK etc.
system CONSTANT : TIB CSTART
system VECTOR   : KEY EMIT etc.
system VARIABLE : STATE BASE etc.

Each type has a number of prefixes (methods) which can act on it. A list of prefixes and the types they work on, can be found in paragraph 19.6 .

15.2 Supplied words

```
TO      +TO     FROM    CLEAR   SET     PUSH    POP     ADR     IS
```

15.3 Defining new prefixes

Defining a new prefix consist of three actions:

a) Define a new prefix operator.
b) Define a runtime action for the datatype it must act on.
c) Extending the methods WORDLIST for that datatype.

An example, a new prefix which sets all bits of a system variable. Starting with defining the 'new' prefix (a):

```
PREFIX DECR        ( That was not to difficult )
```

Note: The prefix SET already exists.
Now define the runtime action (b):

```
: (DECR)    ( inline# -- )
    -1  INLINE#  +!
    ;  C/O  TAIL
```

The runtime word is named (DECR) and it uses internally the word

INLINE# which picks up and skip an inline data cell. The inline cell is the address of the system variable to be adjusted. The word C/O marks this word as a compile only word, and TAIL marks the word as invalid for tail optimising (Words that use inline data need the return address, so the call can not be modified to a jump). They are used in the same way as the standard word IMMEDIATE .

Finally the action must be added to the type vocabulary of the system variables (c):

```
METHODS DOUVAR

: DECR  ( body -- )
    POSTPONE (DECR)  @ , ;

END-METHODS
```

All actions are done, but what are all these weird actions ? The METHOD structure is explained in chapter 21. The @ , is used here, because a system data type holds in its body a pointer to the real data address. The @ picks up the pointer and compiles it inline, just behind (DECR).

15.4 TO-concept word glossary

```
+TO                "plus-to"                            EXTRA
    ( n|u "name" -- )
    Runtime: ( n|u "name" -- )
    Skip leading spaces. Parse name delimited by a space.
    Add n|u to name. ANSI error -32 is issued if name was not
    defined by VALUE , (LOCAL) etc.

    Compilation: ( "name" -- )
    Skip leading spaces. Parse name delimited by a space.
    Append the run-time semantics below to the current definition.
    ANSI error -32 is issued if name was not defined by VALUE ,
    (LOCAL) etc.

    Run-time: ( n|u -- )
    Add n|u to name.
    See also: VALUE (LOCAL) PREFIX
```

```
ADR                                                          EXTRA
    ( -- )
    Runtime: ( "name" -- a-addr )
    Skip leading spaces. Parse name delimited by a space.
    Leave a-addr associated with name on the stack. ANSI error -32
    is issued if name was not defined by VALUE , (LOCAL) etc.

    Compilation: ( "name" -- )
    Skip leading spaces. Parse name delimited by a space.
    Append the run-time semantics below to the current definition.
    ANSI error -32 is issued if name was not defined by VALUE etc.

    Run-time: ( -- a-addr )
    Put a-addr associated with name on the stack.
    See also: VALUE PREFIX


CLEAR                                                        EXTRA
    ( -- )
    Runtime: ( "name" -- )
    Skip leading spaces. Parse name delimited by a space.
    Store zero in name. ANSI error -32 is issued if name was not
    defined by VALUE , (LOCAL) etc.

    Compilation: ( "name" -- )
    Skip leading spaces. Parse name delimited by a space.
    Append the run-time semantics below to the current definition.
    ANSI error -32 is issued if name was not defined by VALUE ,
    (LOCAL) etc.

    Run-time: ( -- )
    Store zero in name.
    See also: VALUE (LOCAL) PREFIX

FROM                                                         EXTRA
    ( -- )
    Runtime:  ( "name" -- x )
    Skip leading spaces. Parse name delimited by a space.
    Place x associated with name on the stack. ANSI error -32 is
    issued if name was not defined by VALUE , (LOCAL) etc.

    Compilation: ( "name" -- )
    Skip leading spaces. Parse name delimited by a space.
```

Append the run-time semantics below to the current definition.
ANSI error -32 is issued if name was not defined by VALUE ,
(LOCAL) etc.

Run-time: (-- x)
Place x associated with name on the stack.
See also: VALUE (LOCAL) PREFIX

IS EXTRA
(--)
Runtime: (xt "name" --)
Skip leading spaces. Parse name delimited by a space.
Store xt in name. ANSI error -32 is issued if name was not
defined by UVECTOR (Only when metacompiling).

Compilation: ("name" --)
Skip leading spaces. Parse name delimited by a space.
Append the run-time semantics below to the current definition.
ANSI error -32 is issued if name was not defined by DOUVEC etc.

Run-time: (xt --)
Store xt in name.
See also: DOUVEC PREFIX

POP EXTRA
(--)
Interpretation: ("name" --) (R: x --)
Skip leading spaces. Parse name delimited by a space.
Pop x associated with name from the return stack. ANSI error -32
is issued if name was not defined by VALUE etc.

Compilation: ("name" --)
Skip leading spaces. Parse name delimited by a space.
Append the run-time semantics below to the current definition.
ANSI error -32 is issued if name was not defined by VALUE etc.

Run-time: (--) (R: x --)
Pop x associated with name from the return stack.
See also: VALUE PREFIX etc.

PREFIX EXTRA
("name1" --)
Skip leading space delimiters. Parse name1 delimited by a space.

Create a definition for name1 with the execution semantics defined below.

name Execution: (i*x "name2" -- j*x)
Skip leading space delimiters. Parse name2 delimited by a space.
Execute the prefix action of name1. Error -64 will be issued
if this prefix is not valid for this word or datatype.

name Compilation: (i*x "name2" -- j*x)
Skip leading space delimiters. Parse name2 delimited by a space.
Compile the prefix action of name1. Error -64 will be issued
if this prefix is not valid for this word or datatype.

PUSH EXTRA
(--)
Runtime: ("name" --) (R: -- x)
Skip leading spaces. Parse name delimited by a space.
Push x associated with name on the return stack. ANSI error -32
is issued if name was not defined by VALUE etc.

Compilation: ("name" --)
Skip leading spaces. Parse name delimited by a space.
Append the run-time semantics below to the current definition.
ANSI error -32 is issued if name was not defined by VALUE etc.

Run-time: (--) (R: -- x)
Push x associated with name on the return stack.
See also: VALUE PREFIX etc.

SET EXTRA
(--)
Interpretation: ("name" --)
Skip leading spaces. Parse name delimited by a space.
Set all bits of name to ones. ANSI error -32 is issued if name
was not defined by FLAG etc.

Compilation: ("name" --)
Skip leading spaces. Parse name delimited by a space.
Append the run-time semantics below to the current definition.
ANSI error -32 is issued if name was not defined by FLAG etc.

Run-time: (--)
Set all bits of name to ones.

See also: FLAG PREFIX

```
TO                                                         FORTH
    ( -- )
    Interpretation: ( x "name" -- )
    Skip leading spaces. Parse name delimited by a space.
    Store x in name. ANSI error -32 is issued if name was not
    defined by VALUE , (LOCAL) etc.

    Compilation: ( "name" -- )
    Skip leading spaces. Parse name delimited by a space.
    Append the run-time semantics below to the current definition.
    ANSI error -32 is issued if name was not defined by VALUE ,
    (LOCAL) etc.

    Run-time: ( x -- )
    Store x in name.
    See also: VALUE (LOCAL) PREFIX
```

15.5 Internal structure of compiled prefixes

An example of the memory structure when DP and +TO DP are compiled:

```
-----------------------------------
|DP|.....|(+TO)|Data address of DP|
-----------------------------------
```

DP is compiled as a direct reference to the system value DP , and +TO DP is compiled as a reference to the runtime code (+TO) and thereafter (inline) the data (RAM) address which belongs to DP .

15.6 Prefixes (methods) for the existing types

First on the line is the name of the DOER: or DOERCODE word, which belongs to the datatype named within paren. The comment <meta only> means that the defining word for that datatype was only present when this system was built. After paren the list of valid prefixes for that datatype are listed.

DOER	BUILDING WORD	PREFIXES
DOUVEC	(UVECTOR <meta only>)	PUSH POP IS ADR
DOUVAR	(UVARIABLE <meta only>)	PUSH POP TO +TO CLEAR FROM
DOUVAL	(UVALUE <meta only>)	PUSH POP TO +TO CLEAR ADR
DOUCON	(UCONSTANT <meta only>)	No prefixes for the user
DOVAR	(VARIABLE)	PUSH POP TO +TO CLEAR FROM
DOVAL	(VALUE)	PUSH POP TO +TO CLEAR ADR
DOLOCAL	(LOCALS| and LOCAL)	TO +TO

Chapter 16

Methods mechanism

This systems offers a way of creating new data types, which use prefix words to access them. This is done with a mechanism that creates a small list of words (mini vocabulary) which is associated to a datatype. They have no name and their link is hidden in the data structure to build the new words. In these mini vocabularies the prefix actions of the defining word are stored. If the list is empty there are no prefix actions for that defining word. A prefix word like TO searches the mini vocabulary of the defining word of the used datatype. Because its a vocabulary, new prefix actions may be added to an existing datatype. If a new datatype has the same internal representation as an existing one, inheritance is possible. In this way standard data types become objects with their own sealed actions in a mini vocabulary.

The prefix operators work exactly as is described in the ANS-Forth document for the word TO. Already prefixes are used in ANS-Forth by local variables and values. So it seams naturally to make the mechanism available to the programmer. Most data types in this system can be used in conjunction with prefix operators.

16.1 Method introduction

The methods structure consist of three parts:

- Opening a methods word list (mini vocabulary)
  When opening, the hidden word list of a defining word is made the compilation word list.
- Extending the methods word list

All new definitions will be added to the this word list.
Definitions in this vocabulary will always be executed.
They behave like compiler directives.

- Closing the methods word list
  The previous compilation vocabulary is restored.

Definitions in methods word lists must describe the compile time behaviour of that word only. The system takes care of its interpret time behaviour. Things you must not forget when defining new methods:

- Define the runtime actions of methods before you open any methods word list.
- A created word in a method word list can not be found when compiling these new methods.
- When executing, a method leaves the body address of the used child on the stack.
- It describes the compile time action only.
- The name of the method must be the same as its associated prefix operator.

When a user defined method is forgotten, the method vocabulary of that type is automatically adjusted (see also forget fields chapter 16).

16.2 Supplied words

METHODS INHERIT PREFIX

16.3 Defining a new method

We will define a new prefix operator for the datatype VALUE . It must increase the contents of the value named, by one. The name chosen for this prefix is INCR . First we will define the runtime action:

```
: (INCR)    1  INLINE# +! ;    ( -- )
```

Next we define the new method for all defined VALUEs:

```
METHODS DOVAL   ( Its defined with a DOER )

    : INCR  POSTPONE (INCR)  , ;           ( childs-body -- )
```

```
END-METHODS
```

The first action in INCR will later compile the runtime code into a definition. The second action compiles the childs body address inline after (INCR) . At last the new prefix operator is defined:

```
PREFIX INCR
```

That's it.

16.4 Defining a new data type with prefix operators

First the defining word is created, in this example with a DOER: it describes the default action of the type (default means without prefix) and how a new child is created:

```
DOER: DOCOUNTER     @ ;     ( Push count on the stack )

: COUNTER           CREATE  0 ,  DOCOUNTER ;
```

Secondly we create a method for this datatype. This method uses the previously defined runtime code and prefix operator:

```
METHODS DOCOUNTER

    : INCR  POSTPONE (INCR)  , ;

END-METHODS
```

With COUNTER we define a datatype with the actions; leave the counter value or increase the counter by one. This is how:

```
COUNTER LAMPS   ( Create new counter named LAMPS )
LAMPS .         ( Printing LAMPS contents <0> )
INCR LAMPS      ( Adding one to the contents of LAMPS )
Etc.
```

16.5 Inheritance

The previous word can be defined smarter, by making use of INHERITance. This systems allows copying methods from an existing datatype, to a newly created datatype. The programmer must be aware of possible problems:

- The new datatype has another representation internally as the existing datatype (take care! The system may crash).
- It may only be used on a empty target datatype.

We can make use of old methods which fit for this type, instead of creating new methods. So we will use the methods from the value datatype. Internally they have the same representation.

Like this:

```
DOER: DOCOUNTER     @ ;     ( Push count to stack )

: COUNTER           CREATE  0 ,  DOCOUNTER ;

INHERIT DOVAL DOCOUNTER
```

Counter can use all methods defined for the VALUE datatype. See paragraph 19.6 for a summary of all data types with valid methods (prefix operators).

16.6 Methods words glossary

```
INHERIT                                                     EXTRA
    ( "name1" "name2" -- )
    Copy all word list information from the type word list addressed
    by "name1" to the one addressed by "name2". An error condition
    exists if "name2" has a non empty type word list.

METHODS                                                     EXTRA
    ( "name" -- )
    Skip leading space delimiters. Parse name delimited by a space.
    Create a dictionary entry, holding all the necessary data
    to restore a type-word list when this methods entry is removed.
    Make the type-word list of "name" the compilation word list.
    Evaluate all text until the prase END-METHODS is encountered.
    Next restore the original compilation word list (cq. vocabulary).
    If an error occurs, first the original compilation word list is
    restored, then control is given to the systems error handler.

PREFIX                                                      EXTRA
    ( "name1" -- )
    Skip leading space delimiters. Parse name1 delimited by a space.
    Create a definition for name1 with the execution semantics defined
    below.

    name Execution: ( i*x "name2" -- j*x )
    Skip leading space delimiters. Parse name2 delimited by a space.
    Execute the prefix action of name1. Error -64 will be issued
    if the prefix is not valid for this word or datatype.

    name Compilation: ( i*x "name2" -- j*x )
    Skip leading space delimiters. Parse name2 delimited by a space.
```

Compile the prefix action of name1. Error -64 will be issued if the prefix is not valid for this word or datatype.

16.7 Internal structure of methods

When new methods are to be created, a methods-child is created (with zero header). This methods-child keeps in its data-field the data to restore an extended methods word list.

```
---------------------------------------
|zeroheader|domethods|toplfa|toplfaptr|
---------------------------------------
                      ^      ^
                      |      Address of toplfa pointer of datatype
                      Old top link field of datatype
```

16.8 Methods example (a string variable)

A string variable will be defined, it uses the already existing prefix operators TO and +TO . Other technical details:

- Maximum string length 255 characters.
- Strings are not initialised, after being defined the data field contains random data.
- Default action, leave string parameters address and length on the stack.
- Store a string with TO or append a string with +TO .
- No string overflow security is present (add it or take care).

First the runtime routines:

```
: (TO$)     ( c-addr u inline# -- )
    INLINE# PLACE ;     ( Store string at inline address inline#)

: (+TO$)    ( c-addr u inline# -- )
    INLINE#  >R TUCK    ( Save string address and length       )
    R@ COUNT + SWAP MOVE ( Add string behind present string    )
    R> C+! ;            ( Adjust string length                 )
```

Define default runtime action and the defining word:

```
DOER: DO$VAR    ( -- C)  ( -- c-addr u E)
    COUNT ;             ( Push string address and length        )

: $VAR          ( +n "name" -- )
    CREATE              ( Define string var. with "name"        )
    255 UMIN 1+ CHARS ALLOT ( Reserve +n + 1 chars string space )
    DO$VAR ;            ( Install default runtime action        )
```

Finally the methods for the string variable are defined:

```
METHODS DO$VAR          ( Start defining methods for "stringvar")
: TO    POSTPONE (TO$)  , ; ( Compile code for TO "stringvar"   )
: +TO   POSTPONE (+TO$) , ; ( Compile code for +TO "stringvar"  )
END-METHODS             ( Stop defining methods for "stringvar" )
```

Define a string variable (64 $VAR TEXT) and test the prefix operators on the newly defined word. First clear the variable (S" " TO TEXT). Check the contents (TEXT TYPE). The result should be a zero string. Then fill the string (S" Hello " TO TEXT) and append the string (S" Willem " +TO TEXT). Check the new contents. It should be "Hello Willem " (TEXT TYPE).

Chapter 17

Interrupt handling

The hardware and software of the personal computer is largely controlled through the use of interrupts. Of course Forth is provided with tools to use them. Also some interrupts are used by the CHForth itself.

17.1 Used interrupts

Interrupt 0 is called when a division overflow or a division by zero occurs. CHForth redirects the vector to a routine that issues ANS Forth exception -10. This interrupt is reset to its previous value when leaving CHForth with BYE or HALT.

Interrupt 6 is called when a non-existing opcode is encountered. CHForth redirects the vector to a routine that issues CHForth exception -524. This hardware feature is implemented on the 80286 processors and newer but is harmless on a 8086. This interrupt is reset to its previous value when leaving CHForth with BYE or HALT.

Interrupt 1B is called when BIOS receives a Ctrl-Break action. CHForth redirects the vector to a routine that issues ANS Forth exception -28. This interrupt is reset to its previous value when leaving CHForth with BYE or HALT.

Interrupt 1C is called 18.2 times a second by the clock. It is currently not used in CHForth, but can be changed. This interrupt is reset to its previous value when leaving CHForth with BYE or

HALT.

Interrupt 23 is called when DOS receives a Ctrl-Break or Ctrl-C action. CHForth redirects the vector to a routine that discards the key. Directly after this BIOS takes over and issues interrupt 1B, described above. The value of this interrupt is not saved in Forth because when the program terminates, DOS itself will restore it.

17.2 Examples

To get the value of an interupt is by GET-INTERRUPT:

```
    $13 GET-INTERRUPT
```

will leave the segment and offset of the BIOS disk routines on the stack.

```
    $FF00 $0FF0 $13 SET-INTERRUPT
```

will set the address, in this case the system restart address. Whenever a disk access is needed, in a Disk Operating System very often, the computer will restart in this example without saving your code. So here you have very dangerous toys in your hands!.

The word INTVEC is also provided to use the interrupts as if it were objects like VALUEs. The last word is used in the library file CLOCK where interrupt 1C is changed. You may load it by saying `NEEDS clock` and after `CLOCKON` you will see a digital clock in the upper-right corner of the screen and with `CLOCKOFF` it is hidden.

17.3 Interrupt words glossary

GET-INTERRUPT EXTRA

(n -- x-addr)

Return the extended address x-addr of the interrupt vector n.

INTVEC "interrupt-vector" INTVEC

(x "name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. Name is referred to as an "interrupt vector".

name Executing: (-- x-addr)
Place x-addr, the extended address of the current vector assigned to interrupt number x. The value of this vector can be changed by executing 'addr TO name', can be reset to its initial value by 'CLEAR name' and the number x can be obtained by executing 'FROM name'. To get the address where the default value is stored, use 'ADR name'.

SET-INTERRUPT EXTRA
(x-addr n --)
Set interrupt vector n to extended address x-addr.

Chapter 18

The decompiler

The decompiler, if not already in memory, can be loaded by

```
NEEDS -decompiler
```

As a result of the threaded code mechanism there is a nearly one-to-one relationship between source and object code. The decompiler is a program that helps the programmer to view compiled code in a form that resembles the source for that code.

18.1 What can be decompiled

- Colon definitions with inline literals, locals and compiler structures.
- Constants
- Variables
- Definitions made with Create
- Vocabularies

18.2 What can not be decompiled

Code definitions like DROP, 2@, EXIT can not be decompiled with the standard decompiler, see chapter 19, the disassembler.

18.3 Examples

```
see space
: SPACE
        BL EMIT
        ;   ans  ok
```

As you see most words are written in capital letters, and some indentation is helpful to view the structure of the words. The decompiled text could even be placed in a log file and after some editing made ready for reloading. The last word 'ans' signifies compliance to the standard.

```
see spaces
: SPACES
        0 MAX 0
        ?do     SPACE
        Loop
        ;   ans  ok
```

Literals between -9 and 9 are printed as decimal digits. Other numbers as four digit hexadecimal numbers with a leading dollar sign. Compiler directives as ?DO and LOOP are printed with one capital followed by lower case letters.

```
' bl (see)
CONSTANT BL             $0018     32 ok
```

The word (SEE) expects an execution token on the stack and decompiles the word associated with it. The value of the constant is printed as well in hexadecimal as in decimal.

18.4 Decompiler words glossary

(SEE) DECOMPILER

(xt --)

Decompile the definition that has xt as its execution token.

ALL DECOMPILER

(--)

Decompile all words in the context word list.

BTW DECOMPILER

("name1" "name2" --)

Decompile all words in the context word list between "name1" and "name2" inclusive, the order does not matter.

DECOMPILER DECOMPILER

(--)

Set the context to the DECOMPILER word list.

NO. DECOMPILER

(--)

The decompiler shows only the names of the definitions.

SEE DECOMPILER

("name" --)

Parse "name" delimited by spaces and decompile or disassemble it.

TILL DECOMPILER

("name" --)

Decompile all words in the context word list newer than "name" and itself.

YES. DECOMPILER

(--)

Set decompiler to normal.

Chapter 19

The disassembler

This disassembler can display the assembler code from code definitions in Forth and any code in the 1 Mb of the PC and the first 64 Kb of the HMA on AT and higher machines.

The disassembler, if not already in memory, can be loaded by

```
    NEEDS -disassembler
```

19.1 What can be disassembled

Code in Forth can be disassemble by using the word DIS or, when the decompiler is loaded before the disassembler, also by SEE. Code in other segments can be decompiled by DISX.

19.2 What can not be disassembled

When you use CHForth-86 it is not possible to decompile specific 386 code. CHForth-386 can decompile some enhanced instructions, but I did not find it necessary to provide a solution for all opcodes.

Only data that belong to a known Forth word, such as a variable or a constant is displayed as data. But, although this is a symbolic disassembler, most data in Forth does not have labels, so it is shown as if it was code. so care is to be taken when interpreting what you see.

19.3 Examples

```
FORTH> see +
\   +  ANS
cseg:0828  pop     ax                          58                  X
cseg:0829  add     bx,ax                       03D8                ..
cseg:082B  next                                26ADFFE0            &...
```

When an address can be associated with a header, the name is printed along with flags as ANS, IMMEDIATE, COMPILE-ONLY and HIDDEN. The other lines start with the segment, when in Forth one of the five symbols: cseg, lseg, hseg, eseg or stac is used, otherwise the hexadecimal value. Then the offset followed with the code on that address. After the middle follows the display of bytes, first in hex, at the end with SEMIT.

The sequence

```
            LODSW   ES:
            JMP     AX
```

is displayed with the name of the macro.

```
FORTH> ' if dis
\   IF  immediate compile-only ans
cseg:11D8  jmp     docolon $031E            E9DBEFFC1E03          ......
\   ELSE  immediate compile-only ans
cseg:11DE  jmp     docolon $032A            E9D5EFFC2A03          ....*. ok
```

As CHForth generates data on aligned addresses, jumps and calls at the start of definitions are followed by a byte $FC(that is the instruction CLD that here never is executed). The address after docolon is the start of the colon definition in LSTSEG.

```
FORTH> ' bl dis
\   BL  ans
cseg:01DE  jmp     doconstant $0020         E99FFFFC2000          .... .
\   FALSE  ans
cseg:01E4  jmp     doconstant $0000         E999FFFC0000          ...... ok
```

Interrupt vectors can be dissassembler by this method:

```
        $21 GET-INTERRUPT DISX
```

19.4 Disassembler words glossary

```
DIS               "disassemble"                       DISASSEM
    ( addr -- )
    Disassemble from address addr.

DISASSEMBLER                                          DISASSEM
    ( -- )
    Replace the first word list in the search order with the
    DISASSEMBLER word list.

DISX              "dis-extended"                      DISASSEM
    ( x-addr -- )
    Disassemble from extended address x-addr.

SEE                                                   DECOMPILER
    ( "name" -- )
    Skip leading space delimiters. Parse name delimited by a space.
    Find name. If name can not be found exception -13 occurs.
    If name is high level, decompile it. Otherwise if the
    disassembler is loaded, disassemble it.
```

Chapter 20

The viewer

The viewer, if not already in memory, can be loaded by

```
    NEEDS -view
```

As a result of the threaded code mechanism there is a nearly one-to-one relationship between source and object code. The viewer is a program that helps the programmer to view compiled code in a form that resembles the source for that code albeit in less explicit form than with the decompiler. The viewer is a simple type of decompiler and takes much less room than the decompiler or disassembler.

20.1 What can be viewed

- Colon definitions with inline literals, locals and compiler structures.
- Constants
- Variables
- Definitions made with Create
- Vocabularies

20.2 What can not be viewed

Code definitions like DROP, 2@, EXIT will be shown as if they were data, see chapter 19, the disassembler.

20.3 Examples

```
FORTH> view space
cseg:23E2 : SPACE
            lseg:1ADA  01E8  BL
            lseg:1ADC  0374  EMIT
            lseg:1ADE  06A2  ;
```

The number on the far left is the address in the code segment. The colon is the type of the definition and SPACE is its name. As this is a colon definition, the next lines are indented and the address in the list segment is showed first, with the contents following it. On the right is the name of each compiled word.

```
FORTH> view spaces
cseg:23E8 : SPACES
            lseg:1AE0  06AD   $0000  False
            lseg:1AE4  0CCB  MAX
            lseg:1AE6  06AD   $0000  False
            lseg:1AEA  06FE  ?DO    lseg:1AF2
            lseg:1AEE  23E2  SPACE
            lseg:1AF0  0725  LOOP
            lseg:1AF2  06A2  ;
```

Literals printed as four digit hexadecimal numbers with a leading dollar sign. Some numbers like -1 and 0 are printed further as TRUE or FALSE. Values between 1 and 31 are printed as control characters and values between 32 and 127 as characters. Compiler directives as ?DO and ELSE are printed with a jump address after it.

```
FORTH> ’ bl (view)
cseg:01E8 CONSTANT BL
cseg:01EC   0020    ’ ’      ’ .’
```

The word (VIEW) expects an address on the stack and views the code that is there in the code segment. The hexadecimal value of the constant is printed. The value is followed by a ASCII value when it is displayable or a decimal value if not. At the and ASCII dump of the two bytes is given in single quotes.

20.4 Viewer words glossary

```
(VIEW)                                                   VIEW
    ( addr -- )
    Display data in the code segment from addr.

VIEW                                                     VIEW
    ( "name" -- )
    Find "name" in the search-order or convert it to an address.
    Display one line at the time of data with, space continues,
    other keys terminate.
```

Chapter 21

The interface with DOS

As Forth gives access to nearly every part of the computer system, words to access the internal memory and external ports is available.

21.1 The DOS environment

DOS reserves a segment to store the environment strings when a program is loaded. Its segment number in CHForth is returned by ESEG and its size in paragraphs by ELEN.
For example, if you want to know the value of the DOS PATH variable, simply use

```
    S" PATH=" SEARCH-ENVIRONMENT
```

to return the string. Remember that the string given by S" has to be in uppercase. This is just the way the CHForth string variable COMSPEC is initialized to get the name and path of the operating system command interpreter.

21.2 External ports

The eight and sixteen bit ports of the PC can be accessed by the words PC@ and PC! or P@ and P! respectively. For example,

```
    $61 PC@ 3 OR $61 PC!
```

puts the speaker on.

21.3 The screen

The text screen segment is in the constant SBASE that is initialized when the program is started and contains $B800 on color and $B000 on monochrome systems. For example

```
    $0720 SBASE 0 !X
```

puts a space (20) with black background and white foreground (07) in the leftmost position on the first line of the screen.

The textmode is set with TEXT or TEXT0 that restores the textmode to that when CHForth was started. When you have a Speedstar Pro videoboard TEXT1 gives 132x25 text screen and TEXT2 a 132x43 textscreen. The size of the screen, inclusive the attribute bytes is returned by SCREENSIZE, so you could save the screen in memory or on disk. Other modes can be set by SETMODE and asked by GETMODE. To use this consult the manual of your video adapter.

You can get the with of the screen from C/L and the height with L/SCR . On some systems the latter is always set to 25 as the byte at 40:84 is not defined for the PC/XT. The current character attribute is in the variable ATTR and the the default in the variable ATT0 . Inverse characters are emitted after INVERS , blinking occurs after BLINK , highlight after BRIGHT and the opposite is done with -INVERS -BLINK and -BRIGHT . The default value is reset by NORMAL .

The screen is addressable with HOME and AT-XY and the cursor position is returned by ?AT . You can position the cursor on the current line with HTAB .

21.4 The DOS interface glossary

?AT "question-at" EXTRA
(-- u1 u2)
Return the column x1 and row x2 of the cursor on the screen.

AT-XY "at-x-y" FORTH
(u1 u2 --)
Perform steps so that the next character displayed will appear in column u1, row u2 of the current output device, the upper left

corner of which is row zero, column zero. It is a no-op when the operation cannot be performed on the current output device with the specified parameters. Note that for other implementations the result in that case is an ambiguous condition.

ATT0 "attribute-zero" EXTRA
(-- a-addr)
a-addr is the address of a cell containing the default attribute of the characters on the screen.

ATTR "attribute" EXTRA
(-- a-addr)
a-addr is the address of a cell containing the current attribute of the characters on the screen.

BEEP EXTRA
(--)
Make an alarm sound on the speaker. As this is sometimes irritating, try CLICK .

BEEPH EXTRA
(-- a-addr)
a-addr is the address of a cell containing the frequency in Hertz of BEEP.

BEEPL EXTRA
(-- a-addr)
a-addr is the address of a cell containing the duration in milliseconds of BEEP.

BIOS-IO EXTRA
(--)
Set input and output to fast BIOS routines, redirection is not supported.
See also: MS-DOS-IO

BIOS? "bios-query" EXTRA
(-- x)
A value that is true when output goes via fast BIOS and not via slow DOS.

BLINK EXTRA
(--)

Invert the blink character attribute.

```
BLOCK-CURSOR                                          EXTRA
    ( -- )
    Set the cursor form to a block.

BRIGHT                                                EXTRA
    ( -- )
    Invert the bright character attribute.

C/L                "c-per-l"                          EXTRA
    ( -- n )
    Return the number of characters on a screen line.

CLICK                                                 EXTRA
    ( -- )
    Make a more pleasant sort of BEEP.

CONSOLE!           "console-store"                    EXTRA
    ( char -- )
    Write char to the standard output file.

CONSOLE?           "console-query"                    EXTRA
    ( -- x )
    A value that is true when screen output is enabled.

CONSOLE@           "console-fetch"                    EXTRA
    ( -- char | -1 )
    Read character char from the standard input file. If the end of
    the file is reached, return -1.

DEALLOC                                               EXTRA
    ( u -- ior )
    Return the contiguous region of memory outside the data space
    indicated by the segment address u to the system for later
    allocation. u shall indicate a region of memory outside the data
    space that was previously obtained by ALLOC or REALLOC . If no
    exception occurs ior is zero. Othewise ior is the I/O result
    code.

DFTMODE            "default-mode"                     EXTRA
    ( -- )
    Set the screen to the textmode that was current at program start.
```

`ECHO` EXTRA

(--)

When loading echo the lines read to the screen.

`ECHO?` "echo-query" EXTRA

(-- x)

A value that is true when characters are echoed during loading a textfile.

`ELEN` EXTRA

(-- n)

n is the number of paragraphs in the environment segment.

`EOL` "e-o-l" EXTRA

(--)

Emit spaces to clear the line on the screen beyond the cursor.

`ESEG` EXTRA

(-- x)

x is the value of the DOS environment segment.

`GET-DIRECTORY` EXTRA

(-- c-addr u ior)

Get the current directory as a character string specified by c-addr u. The path is preceded by the drive letter and a colon. If no exception occurs, ior is zero. Otherwise c-addr and u are unspecified and ior is the I/O result code.

`GET-INTERRUPT` EXTRA

(n -- x-addr)

Return the extended address x-addr of the interrupt vector n.

`GETDISK` EXTRA

(-- n)

n is the current drive number.

`GETMODE` EXTRA

(-- n)

n is the number of the current screen mode.

`HIDE-CURSOR` EXTRA

(--)

Hide the cursor.

```
HOME                                                       EXTRA
    ( -- )
    Set the cursor on the top left of the screen.

HTAB               "h-tab"                                 EXTRA
    ( u -- )
    If n is greater than zero, emit spaces until the cursor is at
    column u of the current user output device.

INVERS                                                     EXTRA
    ( -- )
    Exchange the character foreground and background colors.

L/SCR              "l-per-s-c-r"                           EXTRA
    ( -- n )
    Return the number of lines on the screen.

LINE-CURSOR                                                EXTRA
    ( -- )
    Set the cursor form to a line.

MS-DOS-IO                                                  EXTRA
    ( -- )
    Set input and output to slow DOS routines, redirection is
    supported.
    See also: BIOS-IO CONSOLE! CONSOLE@

NOECHO                                                     EXTRA
    ( -- )
    When loading do not echo lines read to the screen.

NORMAL                                                     EXTRA
    ( -- )
    Reset to character attribute on the screen to the default
    value.

NOSOUND                                                    EXTRA
    ( -- )
    Turn the speaker off.

OUT                                                        EXTRA
    ( -- x )
    A value that contains the number of characters printed on the
```

current screen line.

P! "p-store" EXTRA

(x1 x2 --)

Write x1 to 16 bit port x2.

P@ "p-fetch" EXTRA

(x1 -- x2)

Read the 16 bit port x1.

PAGE FORTH

(--)

Move to another page for output. Actual function depends on the output device. On a terminal, PAGE clears the screen and resets the cursor position to the upper left corner. On a printer, PAGE performs a form feed.

PC! "p-c-store" EXTRA

(char x --)

Write char to 8 bit port x.

PC@ "p-c-fetch" EXTRA

(x -- char)

Read the 8 bit port x.

PITCH EXTRA

(n --)

Set the frequency of the speaker to n.

RESTORE-METRICS EXTRA

(--)

When returning from a system call, reset some screen parameters.

SBASE "s-base" EXTRA

(-- x)

x is the segment number of the text screen.

SCREENSIZE EXTRA

(-- n)

n is the total count of characters plus attributes on the screen.

SEARCH-ENVIRONMENT EXTRA

(c-addr1 u1 -- c-addr2 u2)

Search the DOS environment strings for the string specified by c-addr1 u1. Return the character string after the first string as a character string specified by c-addr2 u2. If the string is not found, u2 is zero and c-addr2 is unspecified.

SEGMENT EXTRA
(x "name" --)
Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. Leave the dictionary pointer at an aligned address. Allocate space for 3 cells. Ask DOS for an allocation of x paragraphs and store the segment number of that allocation in the first cell. Store x in the second cell and zero in the third. The user may change the value of the third cell to a value less than or equal to x in order to save the allocated area with the program.

name Execution: (-- a-addr)
a-addr is the address of the first reserved cell of name.

SET-DIRECTORY EXTRA
(c-addr u -- ior)
Set the current directory to the string specified by c-addr u. As an extension to DOS, the default drive can also be changed if a drive letter and a colon are present at the beginning of the string. If no exception occurs, ior is zero. Otherwise ior is the I/O result code.

SET-INTERRUPT EXTRA
(x-addr n --)
Set interrupt number n to extended address x-addr.

SETDISK EXTRA
(n1 -- n2)
Set the current drive to n1. n2 is the the total number of available drives.

SETMODE EXTRA
(n --)
Set the screen to mode n.

SHOW-CURSOR EXTRA
(--)

Display the cursor.

SOUND EXTRA

(--)

Turn the speaker on.

TRAP EXTRA

(--)

Jump back the debugger program, use it when you want to step through Forth.

Chapter 22

Maintenance of program files

This chapter assumes that you have an editor, preferably SZ.COM or NE.COM at your disposal.

22.1 Generating new source files

One way to create program files is to type EDIT optionally followed with a filename. No extension is needed, as the default is .FRT set in the counted string FEXT$. The format of the files is plain ASCII and can include tabs but because in DOS this is always fixed to eight positions, this is too rigid to be useful.

A more uniform file format with headers and footers is obtained by typing PROJECT followed by the filename. The editor is entered at a place where you can start typing. The strings that can be customized are PROJ$ CAT$ and CREAT$ that are in the file CHFORTH.CFG.

22.2 Library files

Some program parts are used in other programs so it might be convenient to put them in a separate file. Words that are included in the Standard are already in CHFORTH.EXE but some words that are typed "obsolescent" in the Standard are found in the file LIB\OBSOLETE.FRT and can be loaded in with

```
    NEEDS -obsolete
```

if you need them. A decompiler, disassembler and logger are not always needed, so you can put a \ (backslash) in front of the line in CHFORTH.CFG where they are loaded with NEEDS .

There is really no difference in the files in the current directory from the files in the LIB directory, both can be loaded with IN followed by their path and name, but NEEDS does the same without a path for the files in LIB so you can place these files anywhere provided you change the line with LIBPATH in CHFORTH.CFG accordingly.

Most library files have a MARKER word in front of them, as the system of libraries is modular, the list of loaded files can be displayed by .MODULES .

22.3 Logging

All the user or the program displays on the screen can be logged to a file. Pressing F2 (when ACCEPT.FRT is loaded) or typing OPEN-LOG will create a file called FORTH.LOG or append to an existing file with that name. The status line is disabled. The logging is ended by pressing F2 again, typing CLOSE-LOG or automatically by typing BYE . The file name can be changed, it is in the counted string at LOGFILE .

Needed file: LIB\LOG.FRT

22.4 Glossary generation

Automatic glossary generation (making of help files) is possible. The word \G is an alias for \ so interpreting will skip the lines that have it in front of it. The glossary generator however, parses the following string and will put them along with the following defined word in a glossary file. With NEW-GLOSS you reset the generator. MAKE-GLOSS followed by the full filename will load and parse the file and this may be repeated until all the files are processed or the memory is full. WRITE-GLOSS will write the data to a file, extension preferably .HLP and directory DOC so the word HELP can immediately be used.

Needed file: MAKEHELP.FRT

22.5 Maintenance words glossary

```
,EDIT                                                   EDITOR
    ( u "name" -- )
    Skip leading space delimiters. Parse name delimited by a space.
    Open file name with the editor program and place the cursor at
    line u. When name is omitted, the last opened file by this
    command or EDIT LIST or WHAT is opened and name is displayed on
    the right of the status line. The default extension is taken from
    FEXT$ .

.MODULES                                                EXTRA
    ( -- )
    Display the list of words that are created by MARKER .

?DEF              "query-defined"                       EXTRA
    ( "name" -- flag )
    Skip leading space delimiters. Parse name delimited by a space.
    Find name. If name is found, flag is true, false otherwise.
    See also: ?UNDEF

?UNDEF            "query-undefined"                     EXTRA
    ( "name" -- flag )
    Skip leading space delimiters. Parse name delimited by a space.
    Find name. If name is found, flag is false, true otherwise.
    See also: ?DEF

CAT$                                                    EXTRA
    ( -- c-addr )
    c-addr is the address of a counted string containing a
    description of the category to which this file belongs.

CLOSE-LOG                                               LOG
    ( -- )
    Close the log file.

CREAT$                                                  EXTRA
    ( -- c-addr )
    c-addr is the address of a counted string containing the name of
    the creator of this file.

EDIT                                                    EDITOR
```

("name" --)
Skip leading space delimiters. Parse name delimited by a space. Open file name with the editor program and place the cursor at the first line. When name is omitted, the last opened file by this command or ,EDIT LIST or WHAT is opened and name is displayed on the right of the status line. The default extension is taken from FEXT$.

EDLIB EDITOR
("name" --)
Skip leading space delimiters. Parse name delimited by a space. Open file name in the directory given in LIBPATH with the editor program and place the cursor at the first line. The default extension is taken from FEXT$.

LIBPATH EXTRA
(-- c-addr)
c-addr is the address of a counted string containing the path to the library files.
See also: HELPPATH NEEDS

LOGFILE LOG
(-- c-addr)
Contains the name of the logfile.

GLOSS "glossary" FORTH
("fname1" "fname2" --)
Make a glossary with name2 out of the origin file name1 .

MAKE-GLOSS "make-glossary" FORTH
("name" --)
This word reads a source file and builds the glossary information for it in memory.

NEEDS EXTRA
("name" --)
Skip leading space delimiters. Parse name delimited by a space. Find name. If found continue. Otherwise, load the file with the same name (excluding an optional trailing minus sign) from the directory specified in LIBPATH .

NEW-GLOSS "new-gloss" FORTH
(--)

```
    This command starts a fresh glossary.

OPEN-LOG                                                    LOG
    ( -- )
    Open the logfile.

PROJ$                                                       EXTRA
    ( -- c-addr )
    c-addr is the address of a counted string containing a
    description of the project for which the file is created.

PROJECT                                                     PROJECT
    ( "name" -- )
    Skip leading space delimiters. Parse name delimited by a space.
    Create a text file for name with the default extension in FEXT$ .
    Write a header as defined in the strings PROJ$ CAT$ and CREAT$
    and start the editor with the cursor at a place where the
    programmer can start typing. This file can be loaded directly
    after editing by typing IN . After the header is a MARKER for
    an automatic FORGET when reloading the file.

WHAT                                                        EDITOR
    ( -- )
    Open file name with the editor program and place the cursor at
    the line number stored in ERRLINE . name is stored at the address
    stored in ERRNAME . ERRNAME and ERRLINE are valid after an
    exception that occurred during loading of file name. name is
    displayed on the right of the status line.

WRITE-GLOSS        "write-glossary"                         FORTH
    ( "name" -- )
    This word writes the glossary info from memory into a file.
    The information may be collected from several source files.

\G                                                          EXTRA
    ( "ccc<eol>" -- )
    If BLK contains zero, parse and discard the remainder of the
    parse area; otherwise parse and discard the portion of the parse
    area corresponding to the remainder of the current line. \G is an
    immediate word. Used in generating glossaries.
```

Chapter 23

Turnkey programs

With CHForth it is possible to write programs that run independent like a filter utility or a game. In such programs an interpreter or compiler is not necessary. As yet it is not possible to delete the compiler and interpreter, but it is easy to ignore them.

23.1 Trimming the system

There are three words, RESERVE LRESERVE and HRESERVE that make it possible to trim the three main segments of CHForth. If for example you need a Forth program with interpreter and compiler that needs only 4 Kb space in each segment to compile a few words after it is loaded and nothing more, you could use the following:

```
    4096 RESERVE    \ Reserve no more than 4 Kb for data
    4096 LRESERVE   \ Reserve only 4 Kb for colon definitions
    4096 HRESERVE   \ Reserve no more than 4 Kb for headers
    SAVE SMALLF     \ Make a program SMALLF.EXE
```

This is nice to be used as a normal CHForth program on systems that do not have 640 Kb conventional memory, the three segments are now each less than 64 Kb in the memory of the computer. Remember to never put data above LIMIT LLIMIT or HLIMIT as they are just beyond the last usable address in their segments. If you need an interpreter but no compiler, you could use 0 for all three RESERVE words, as there will always be some space above HERE for interpreting input.

23.2 Self running programs

A second method is the use of the word TURNKEY . First write a word that does the job that you want to do and then save the program:

```
    EMPTY                       \ Discard any unnecessary code
    : GO            ( -- )      \ Program can not use parameters
        'Z' 1+ 'A'
        DO      I EMIT
        LOOP                    \ No BYE necessary
    ;
    TURNKEY GO ALPHABET         \ Make program ALPHABET.EXE
```

After this you are in DOS, type the name of the program after the prompt and now you see the alphabet and then the DOS prompt.

This technique uses 0 RESERVE 0 LRESERVE to trim the code and list segment and wholly discard the head segment, so interpreting is not possible as headers are not present.

23.3 Examples

Some examples are to be found in \TURNKEY . All can be compiled at the DOS prompt by:

```
    CHFORTH IN filename
```

Or by executing:

```
    MAKE -ffilename
```

If you have the MAKE utility.

Most programs give some information about their workings when you type the name followed by a space and /? or -?

23.4 Turnkey glossary.

.FREE "dot-free" EXTRA

(--)

Display the value of the three dictionary pointers and the free space in their respective segments.

HLIMIT EXTRA

(-- x)
Return the address after the last usable in the head segment.

HMEMTOP EXTRA
(-- addr)
Return the address after the last physical address in the header segment.

HRESERVE EXTRA
(x --)
Reserve x address units above HHERE in the head segment to be used by the compiler in a saved program. When x is zero, all headers of the definitions are discarded in the saved program.

LIMIT EXTRA
(-- x)
Return the address after the last usable in the dictionary.

LLIMIT EXTRA
(-- x)
Return the address after the last usable in the list segment.

LMEMTOP EXTRA
(-- addr)
Return the address after the last physical address in the list segment.

LRESERVE EXTRA
(x --)
Reserve x address units above LHERE in the list segment to be used by the compiler in a saved program. When x is zero, no compiling is possible in the new program.

MEMTOP EXTRA
(-- addr)
Return the address after the last physical address in memory.

RESERVE EXTRA
(x --)
Reserve x address units above HERE to be used by ALLOT in a saved program. Some space is always available in PAD and TEMPORARY so interpreting remains possible if x is zero.

RESTART? EXTRA

(-- x)

A value that prohibits restarting of the initialization of a program. When the program is started its value is false. When Ctrl-Break is pressed, it is set to true.

SAVE EXTRA

("name" --)

Skip leading space delimiters. Parse name delimited by a space. Protect the dictionary as with EXTEND . Write the CHForth program as an executable file with this name. name may have a preceding path but no extension. The current settings of LIMIT and MEMTOP are preserved as are their equivalents in other segments.

TURNKEY EXTRA

("name1" "name2" --)

Skip leading space delimiters. Parse name1 delimited by a space. Skip leading space delimiters. Parse name2 delimited by a space. Protect the dictionary as with EXTEND . Write the CHForth program as an executable file with this name2. name2 may have a preceding path but no extension.

The saved file does not contain any headers, so interpreting in the executable file is not possible. The data space and list space will also be reduced to the minimum value that is needed to contains the current data in the data and list space. Both spaces can be enlarged before executing this word.

When this program is executed from the DOS prompt, name1 will be executed by CATCH and at the end the control will be returned to DOS. The program saved has no capability to compile and has no headers.

UNUSED FORTH

(-- u)

u is the amount of space remaining in the region addressed by HERE , in address units.

Chapter 24

CHForth internals

Traditionally Forth has been implemented as a small model, where code, data, colon definitions, stacks and headers were in one memory space, the dictionary. In CHForth some differentiation has been made.

24.1 Code space

In the codesegement are placed all data outside colon definitions and low level code routines.

Code definitions start without special entering code, as CHForth is direct threaded code, contrary to indirect threaded code, the way FIGFORTH and F83 were implemented, where a pointer preceded every definition. Returning to the next routine is done via the macro NEXT in assembler, that loads the word pointed to by ES:SI into the AX register, increments SI by two to point the next time to the next word (postincrement) and jumps to the address in AX.

Variables and words that are built with CREATE without DOES> have a jump instruction to special code that pushes the inline address on the stack. This could have been a call instruction, but as the top of the stack in CHForth is in a register and for speed a jump seemed faster. Just after the jump is a byte with value $FC, which is the code for the instruction CLD, faster than NOP, but that is of no importance, as it is never executed and only serves to keep HERE aligned. The word >BODY is simply '2 CELLS +' or '4 +' as a cell is 2 characters or bytes. The address after the CLD instruction is called the data field.

Constants and values are made in the same way, a jump to a special routine which pushes the value in the data field on the the stack and then the data itself.

Colon definitions have a jump to a special routine that pushes the current Forth instruction pointer SI on the return stack and makes the contents of the data field the next value of the Forth instruction pointer. This now points to a list of compiled execution tokens and is discussed in paragraph 24.3

The value of the code space is in the CS and DS register and its value is returned by the word CSEG. Access of this memory area is with the traditional @ ! and , etc.

24.2 Header space

To reserve more space in the code segment, a header segment is present, which has all the headers of the definitions.

A header is identified by its dictionary entry address, this is returned by the word >HEAD.

- It starts with link field, that points to the previous word in the current word list.

- Then follows a cell with flags, of which the immediate bit is the most important.

- This is followed by a cell containing a pointer to the forget code associated with this type of word, when there does not exist such a routine, it is zero.

- Next is the pointer to the execution token in the code space.

- At the end is a byte with the count of characters in the name followed with the name itself and padded with a null byte when necessary to make the dictionary entry address even.

Creating a header does not allocate space in the code segment, so making an ALIAS is very simple.

Never use knowledge about the current order of the header fields, this may change in the future, for example, I may add a hashing mechanism to speed up compilation.

The value of the header space is returned by the word HSEG. Access of its data is with H@ H! and H, etc.

24.3 List space

The value in the data field of colon definitions is a pointer to a list of compiled execution tokens (the essence of Forth) that are interpreted one by one by the NEXT macro (in native code Forths this list can be a series of machine code calls and other machine code). This list is placed in another memory space, the list segment.

In the list space are also placed literal numbers and inline strings compiled by ." .

Literal numbers have a preceding execution token that will push the inline value on the stack.

The value of the list space is in the ES register and is returned by the word LSEG. Access is with L@ L! and L, etc.

24.4 String space

Strings compiled by ABORT" S" and C" have a pointer in the list segment that points to the address of the string in the code segment, where the strings themselves are compiled, so on execution the strings can be TYPEd CMOVEd and COUNTed. Direct execution of S" will still compile the string in the a special area in the code space due to the word FLYER and execute them immediately afterwards to place the address and length on the stack. This area is 1024 bytes large and will accept up to four 256 byte long strings and more if they are shorter, there is an overflow area of 256 bytes at the end.

24.5 Stack space

CHForth has three stacks.

- The first is the data stack, normally called simply the stack where numbers, execution tokens and flags are stored. Access is with DUP SWAP DROP PICK ROLL and so forth. The number of elements on the stack is given by DEPTH .

- The second is the return stack, called so because its function is mainly to keep return addresses when nesting occurs by entering colon definitions. It also contains information for do-loops, for-next loops and can temporary be used by words as >R and R> to transfer values from the data stack to the return stack and vice versa.

- The third stack is only used to store local variables. No operators to access this stack apart from the local variables themselves are available.

The value of the stack space is in the SS register and is contained in the variable STKSEG in the INTERNAL word list. As it is not necessary to access the stacks directly, no special accessing words are given. For this the words PICK and ROLL are provided. Never use constructs from other Forths as: : EMIT SP@ 1 TYPE DROP ; This will definitely not work!

24.6 DOS space

DOS reserves a segment to store the environment strings when a program is loaded. See chapter 21.

Access of the memory outside of CHForth is provided by the word SEGMENT . It needs a number of paragraphs (16 byte chunks) on the stack and a name. Using the name gives access to an array of 3 cells of which the first gives the value of the segment in the 640 Kb that is available for DOS. In the second cell is the number of paragraphs. The area is automatically returned to DOS when you forget this word. When have filled the area with data you can place a value giving the size of the area in paragraphs and put it in the third cell. When you save the program, the data will also be saved and will later be available if you execute the program. Of course the value of the segment can then be different, but that is because DOS assigns segments to its memory allocation. Data is accessed by @X !X COUNTX etc. where the appendix -X is short for extended address. The extended address is always in the form segment-offset, where the low word is the segment and the high word is the offset.

Example:

```
    $100 SEGMENT MYDATA      \ Allocate $100*$10 (4096) bytes.
    MYDATA @                 \ Get the segment
    0                        \ An offset
    MYDATA CELL+ @           \ the size
    PARAGRAPHS               \ convert it to byte count
    DUMPX                    \ dump it
    1234 MYDATA @ 20 !X      \ store some data in it
    ( FORGET MYDATA )        \ Return the area to DOS
    700 #PARAGRAPHS          \ Convert to paragraphs
    MYDATA 2 CELLS + !       \ Keep 700 bytes when you save this
    SAVE MYPROG              \ Save CHForth along with MYDATA
```

Chapter 25

Alphabetical index of words

```
!                  "store"                                 FORTH
    ( x a-addr -- )
    Store x at a-addr.

!CSP               "store-c-s-p"                           EXTRA
    ( -- )
    Save the current depth of the stack for checking with ?CSP .

!X                 "store-x"                               EXTRA
    ( x x-addr -- )
    Store x at extended address x-addr.

",                 "quote-comma"                           EXTRA
    ( "ccc<">" -- )
    Parse ccc delimited by '"' (double quote) and compile it as a
    counted string in the dictionary. Execution of HERE just before
    the execution of ", will give the address of the string.

#                  "number-sign"                           FORTH
    ( ud1 -- ud2 )
    Divide ud1 by the number in BASE giving the quotient ud2 and the
    remainder n. (n is the least-significant digit of ud1). Convert n
    to external form and add the resulting character to the beginning
    of the pictured numeric output string. An ambiguous condition
    exists if # executes outside of a <# #> delimited number
    conversion.
    See also: #> #S <#

#>                 "number-sign-greater"                   FORTH
```

(xd -- c-addr u)

Drop xd. Make the pictured numeric output string available as a character string. c-addr and u specify the resulting character string. A Standard Program may replace characters within the string.

See also: # #S <#

#CELLS "number-cells" EXTRA

(n1 -- n2)

n2 is the minimum number of cells needed to store n1 characters.

#CHARS "number-chars" EXTRA

(n1 -- n2)

n2 is the minimum number of address units needed to store n1 characters.

#CPU "number-c-p-u" EXTRA

(-- a-addr)

a-addr is the address of a cell containing the processor type, allowed values are 86 and 386.

#LINES "number-lines" EXTRA

(-- addr)

A variable containing the number of the current line of the current file.

#PARAGRAPHS "number-paragraphs" EXTRA

(n1 -- n2)

n2 is the minimum number of paragraphs needed to store n1 characters.

#S "number-sign-s" FORTH

(ud1 -- ud2)

Convert one digit of ud1 according to the rule for # . Continue conversion until the quotient is zero. An ambiguous condition exists if #S executes outside of a <# #> delimited number conversion.

See also: # #> <#

#TIB "number-t-i-b" FORTH

(-- a-addr)

a-addr is the address of a cell containing the number of characters in the terminal input buffer.

Note: this word is obsolescent and is included as a concession to existing implementations.

```
$                                                         ASSEMBLER
    ( x -- )
    Jump to an assembler label.

$:                                                        ASSEMBLER
    ( x -- )
    Define an assembler label.

$COMPILE         "string-compile"                         EXTRA
    ( c-addr u -- )
    Try to find the name c-addr u in the search order and when
    found execute it or compile it according to the flag returned
    by FIND . Else try to convert the string to a number and
    compile it.  Else issue a warning that the word can not be
    found and compile a forward reference to it.

$ELSE                                                     ASSEMBLER
    ( -- )
    Jump to after $THEN .


$IF386                                                    ASSEMBLER
    ( -- )
    If #CPU does not contain 386 jump to after $ELSE or $THEN .
    Else continue.

$INTERPRET       "string-interpret"                       EXTRA
    ( c-addr u -- )
    Try to find the name c-addr u in the search order and execute
    it when found else convert the string to a number and place it
    on the stack. Else abort with an exception message.

$THEN                                                     ASSEMBLER
    ( -- )
    Terminate a $IF386 directive.

&EXEC:           "and-exec-colon"                         EXTRA
    ( x1 x2 -- )
    Perform a bitwise AND on the two numbers on the stack and use the
    result as an index into the inline execution array and execute
```

the execution token stored there.

' "tick" FORTH
("name" -- xt)
Skip leading space delimiters. Parse name delimited by a space. Find name and return xt, the execution token for name. Exception -13 occurs if name is not found.

When interpreting ' name EXECUTE is equivalent to name.
See also: POSTPONE [']

'ACCEPT EXTRA
(-- a-addr)
a-addr is the address of a cell that contains the execution token of the routine that is executed by ACCEPT .

'COMPILE "tick-compile" EXTRA
(c-addr u --)
A word that normally executes $COMPILE .

'INTERPRET "tick-interpret" EXTRA
(c-addr u --)
A word that normally executes $INTERPRET .

'NAME "tick-name" EXTRA
(-- addr)
Contains the name of the current file.

("paren" FORTH
("ccc<paren>" --)
Parse ccc delimited by a right parenthesis ")". (is immediate.

The number of characters in ccc may be zero to the number of characters in the parse area.

When parsing from a text file, if the end of the parse area is reached before a right parenthesis is found, refill the input buffer from the next line of the file, set >IN to zero, and resume parsing, repeating this process until either a right parenthesis is found or the end of the file is reached.

(* EXTRA

(--)
Repeatedly skip leading spaces, parse and discard space-delimited words from the parse area, until the word *) has been parsed and discarded. If the parse area becomes exhausted, it is refilled as with REFILL . (* is immediate.

An ambiguous condition exists if (* is POSTPONEd. If the end of the input stream is reached and cannot be refilled before the terminating *) is parsed, exception -533 occurs.

(.) "paren-dot" EXTRA
(n -- c-addr u)
Convert n to a numeric output string with a leading minus sign if n is negative.

(.HEAD) "paren-dot-head" EXTRA
(dea -- c-addr u)
c-addr u specify a character string that represents the name of the definition with dictionary entry address dea. If dea is zero the string contains "{NoName}" and when the name is found but the length of it is zero, the string contains "{NullName}". u is limited to 31.

(.T0) "paren-dot-t-zero" EXTRA
(-- c-addr u)
c-addr u specify a string containing the time elapsed since the last execution of TIMER-RESET in the format of a numeric string with three digits after the decimal point.
See also: .ELAPSED

(D.) "paren-d-dot" EXTRA
(d -- c-addr u)
Convert d to a numeric output string with a leading minus sign if d is negative.

(DATE) "paren-date" EXTRA
(-- c-addr u)
c-addr u specify a character string containing the date in the format "Month day, year".
See also: .DATE

(EMIT) "paren-emit" EXTRA
(char --)

Type the character on the output device, default action of EMIT .

(EXIT) "paren-exit" EXTRA

(--) (R: nest-sys --)

End the current definition, an alias for EXIT compiled by ; .

(FORGET) "paren-forget" EXTRA

(xt --)

Forget the definition with execution token xt.

(LINE) "paren-line" EXTRA

(n u1 -- c-addr u2)

Give the address c-addr and length u2 of the line n of the block u1.

(LOCAL) "paren-local-paren" FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (c-addr u --)

When executed during compilation, (LOCAL) passes a message to the Forth system that has one of two meanings. If u is non-zero, the message identifies a new local whose word name is given by the string of characters identified by c-addr u. If u is zero, the message is 'last local' and c-addr has no significance. The result of executing (LOCAL) during compilation of a definition is to create a set of named local identifiers, each of which is a word name, that have execution semantics within the scope of that definition's source only.

local Execution: (-- x)

Push the local's value, x, onto the stack. An ambiguous condition exists when (LOCAL) is executed while in interpret state.

Note: This word is not intended for direct use in a definition to declare that definition's locals. It is instead used by system or user compiling words. These compiling words in turn define their own syntax, and may be used directly in definitions to declare locals.

(NUMBER?) "paren-number-question" EXTRA

(c-addr u -- 0 | n 1 | d 2)
Convert a string to a number. If it fails, return a false flag. Otherwise return a single number with a flag of 1 and a double number with a flag of 2. The number is negative if prefixed by ’-’. CHForth allows decimal numbers to be prefixed by ’#’ , hexadecimal numbers by ’$’ and binary numbers by ’%’ . These may be followed by ’-’ to signify negative numbers. Single characters are converted to single precision number when prefixed by ’&’ or when they are enclosed by ’’’. Uppercase letters can be converted to the corresponding control characters when prefixed by ’ˆ’.

(REF) "paren-ref" REF
(addr --)
Find compiled references in colon definitions of addr in all word lists. Display the words where the references occur and the count of the words where the references are found.

(SEE) DECOMPILER
(xt --)
Decompile the definition that has xt as its execution token.

(SHORTDATE) "paren-shortdate" EXTRA
(-- c-addr u)
c-addr u specify a character string containing the date in the format "dd mmm yy".
See also: (DATE) .SHORTDATE

(TIME) "paren-time" EXTRA
(-- c-addr u)
c-addr u specify a character string containing the time in the format "hh:mm:ss".
See also: .TIME

(VIEW) VIEW
(addr --)
Display data starting from addr.

* "star" FORTH
(n1|u1 n2|u2 -- n3|u3)
Multiply n1|u1 by n2|u2 giving product n3|u3.

*/ "star-slash" FORTH
(n1 n2 n3 -- n4)

Multiply n1 by n2 producing the double-cell intermediate result d. Divide d by n3, giving the single-cell quotient n4. Exception -10 is issued if n3 is zero or if the quotient n4 lies outside the range of a single-cell signed integer. If d and n3 differ in sign the result returned will be the same as returned by the phrase >R M* R> SM/REM SWAP DROP . Note that other implementations of the ANSI standard may return the result of the phrase >R M* R> FM/MOD SWAP DROP .

*/MOD "star-slash-mod" FORTH

(n1 n2 n3 -- n4 n5)

Multiply n1 by n2 producing the double-cell intermediate result d. Divide d by n3, giving the single-cell remainder n4 and the single-cell quotient n5. Exception -10 is issued if n3 is zero or if the quotient n5 lies outside the range of a single-cell signed integer. If d and n3 differ in sign the result returned will be the same as returned by the phrase >R M* R> SM/REM . Note that other implementations of the ANSI standard may return the result of the phrase >R M* R> FM/MOD .

+ "plus" FORTH

(n1|u1 n2|u2 -- n3|u3)

Add n2|u2 to n1|u1, giving the sum n3|u3.

+! "plus-store" FORTH

(n|u a-addr --)

Add n|u to the single-cell number at a-addr.

+!X "plus-store-x" EXTRA

(n|u x-addr --)

Add n|u to the single-cell value at extended address x-addr.

+LOOP "plus-loop" FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: do-sys --)

Append the execution semantics given below to the current definition. Resolve the destination of all unresolved occurrences of LEAVE between the location given by do-sys and the next location for a transfer of control, to execute the words following +LOOP.

Execution: (n --) (R: loop-sys1 -- | loop-sys2)
Loop control parameters must be available. Add n to the loop index. If the loop index was did not cross the boundary between the loop limit minus one and the loop limit, continue execution at beginning of the loop. Otherwise discard the current loop control parameters and continue execution immediately following the loop.
See also: DO I LEAVE

+TO "plus-to" EXTRA
Interpretation: (n|u "name" --)
Skip leading space delimiters. Parse name delimited by a space. Add n|u to name. Exception -32 occurs if name was not defined by VALUE or VARIABLE .

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VALUE , VARIABLE or (LOCAL).

Run-time: (x --)
Add n|u to name.

, "comma" FORTH
(x --)
Reserve one cell of data space and store x in the cell. If the data space pointer is aligned when , begins execution, it will remain aligned when , finishes execution.

,EDIT EDITOR
(u "name" --)
Skip leading space delimiters. Parse name delimited by a space. Open file name with the editor program and place the cursor at line u. When name is omitted, the last opened file by this command or EDIT LIST or WHAT is opened and name is displayed on the right of the status line. The default extension is taken from FEXT$.

- "minus" FORTH
(n1|u1 n2|u2 -- n3|u3)
Subtract n2|u2 from n1|u1, giving the difference n3|u3.

-- EXTRA

("ccc<eol>" --)

If BLK contains zero, parse and discard the remainder of the parse area; otherwise parse and discard the portion of the parse area corresponding to the remainder of the current line. -- is an immediate word.

-R "minus-r" EXTRA

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(--) (R: x --)

Remove x from the return stack.

-ROT "minus-rote" EXTRA

(x1 x2 x3 -- x3 x1 x2)

Rotate the top three stack items. Equivalent to ROT ROT .

-S "minus-s" STACK

(--)

(S: x --)

Drop the top number of the auxiliary stack. An alias for S>DROP .

-TRAILING "dash-trailing" FORTH

(c-addr u1 -- c-addr u2)

If u1 is greater than zero, u2 is equal to u1 less the number of spaces at the end of the character string specified by c-addr u1. If u1 is zero or the entire string consists of spaces, u2 is zero.

. "dot" FORTH

(n --)

Display n in free field format.

." "dot-quote" FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("ccc<quote>" --)

Parse characters ccc delimited by " (double-quote). Append the run-time semantics specified below to the current definition.

Run-time: (--)
Display ccc.
See also: .(

.("dot-paren" FORTH
("ccc<paren>" --)
Parse and display ccc delimited by a right parenthesis ")". .(is immediate.
See also: ."

.DATE "dot-date" EXTRA
(--)
Display the date in the format "Month day, year".
See also: (DATE) .SHORTDATE .TIME

.DEC "dot-decimal" EXTRA
(n --)
Display n as a signed decimal number.
See also: .HEX

.ELAPSED "dot-elapsed" EXTRA
(--)
Display the elapsed time as specified by (.T0) followed by the string " seconds elapsed.".
See also: (.T0) .MS TIMER-RESET

.FREE "dot-free" EXTRA
(--)
Display the value of the three dictionary pointers and the free space in their respective segments.

.HEAD "dot-head" EXTRA
(dea --)
If the length of the name associated with the dictionary entry address dea does not fit on the current line, perform a CR . Type the name and wait for the time in milliseconds contained in WORDSPEED .

.HEX "dot-hex" EXTRA
(u --)

Display u as a four digit hexadecimal number with a leading '$' character and a trailing space.
See also: .DEC H.

.LINE "dot-line" EXTRA
(n u --)
Type line n of block u.

.ME "dot-me" EXTRA
(--)
Display the full path and name of the Forth program.

.MESS EXTRA
(n --)
Display the message that is assigned to exception number n as with MESS" . If the message is not found, display the exception number and the name of the word where the exception occured. If n is -1 or -2 nothing is displayed. Store the number in ERR# .

.MODULES EXTRA
(--)
Display the list of words that are created by MARKER .

.MS "dot-m-s" EXTRA
(--)
Display the elapsed time as specified by (.T0) followed by the string " seconds.".
See also: (.T0) .ELAPSED TIMER-RESET

.R "dot-r" FORTH
(n1 n2 --)
Display n1 right aligned in a field n2 characters wide. If the number of characters required to display n2 is greater than n2, all digits are displayed with no leading spaces in a field as wide as necessary.

.S "dot-s" FORTH
(--)
Copy and display the values currently on the data stack. Starting on a new line, a '(' (left parenthesis) followed by a space is displayed. Then follow the values on the stack, when BASE contains 10, as signed numbers, unsigned otherwise. At the end a ')' (right parenthesis) is displayed.

.S is implemented using pictured numeric output words. Its use will corrupt the transient region identified by #> .

```
.SEG              "dot-segment"                      EXTRA
   ( u -- )
```

Display u as a four character string if it corresponds to a segment in CHForth else as a four digit hexadecimal string.

```
.SHORTDATE        "dot-shortdate"                    EXTRA
   ( -- )
```

Display the date in the format "dd mmm yy".
See also: (SHORTDATE) MONTHS

```
.SIGNON           "dot-signon"                       EXTRA
   ( -- )
```

Display the signon message. It will contain the name of the program, the version number and the name of the author.

```
.STATUS           "dot-status"                       EXTRA
   ( -- )
```

Display the statusline at the top of the screen.

```
.TIME             "dot-time"                         EXTRA
   ( -- )
```

Display the time in the format "hh:mm:ss".
See also: (TIME) .DATE

```
.VOCNAME          "dot-vocname"                      EXTRA
   ( wid -- )
```

Display the name of the word list identification wid.
See also: .HEAD

```
.WHERE                                               EXTRA
   ( -- )
```

If the last exception occurred during loading of a file, display the name of the file and the line number where the exception occurred.

```
.WORDLISTS                                           EXTRA
   ( -- )
```

Display the word lists that have a name, those who have been created with VOCABULARY .

/ "slash" FORTH

(n1 n2 -- n3)

Divide n1 by n2, giving the single-cell quotient n3. Exception -10 is issued if n1 is zero. If n1 and n2 differ in sign the result returned will be the same as returned by the phrase >R S>D R> SM/REM SWAP DROP . Note that other implementations of the ANSI standard may return the result of the phrase >R S>D R> FM/MOD SWAP DROP .

/LINE "per-line" EXTRA

(-- n)

n is the maximum number of characters on an input line.

/MOD "slash-mod" FORTH

(x1 x2 -- x3 x4)

Divide n1 by n2, giving the single-cell remainder n3 and the single-cell quotient n4. Exception -10 is issued if n1 is zero. If n1 and n2 differ in sign the result returned will be the same as returned by the phrase >R S>D R> SM/REM . Note that other implementations of the ANSI standard may return the result of the phrase >R S>D R> FM/MOD .

/STRING "slash-string" FORTH

(c-addr1 u1 n -- c-addr2 u2)

Adjust the character string at c-addr1 by n characters. The resulting character string, specified by c-addr2 u2, begins at c-addr1 plus n characters and is u1 minus n characters long.

0< "zero-less" FORTH

(n -- flag)

flag is true if and only if n is less than zero.

0<> "zero-not-equals" FORTH

(x -- flag)

flag is true if and only if x is not equal to zero.

0= "zero-equals" FORTH

(x -- flag)

flag is true if and only if x is equal to zero.

0> "zero-greater" FORTH

(n -- flag)

flag is true if and only if n is greater than zero.

1+ "one-plus" FORTH
(n1|u1 -- n2|u2)
Add 1 to n1|u1 giving the sum n2|u2.

1- "one-minus" FORTH
(n1|u1 -- n2|u2)
Subtract 1 from n1|u1 giving the difference n2|u2.

2! "two-store" FORTH
(x1 x2 a-addr --)
Store the cell pair x1 x2 at a-addr with x2 at a-addr and x1 at the next consecutive cell. It is equivalent to the sequence SWAP OVER ! CELL+ ! .
See also: 2@

2!X "two-store-x" EXTRA
(x1 x2 x-addr --)
Store the cell pair x1 x2 at extended address x-addr with x2 at x-addr and x1 at the next consecutive cell. It is equivalent to the sequence ROT >R 2DUP R> -ROT !X CELL+ ! .
See also: 2@X

2* "two-star" FORTH
(x1 -- x2)
x2 is the result by shifting x1 one bit toward the most-significant bit, filling the vacated least-significant bit with zero.

2, "two-comma" EXTRA
(x1 x2 --)
Reserve space for two cells in the data space and store x2 in the first cell and x1 in the second.

2/ "two-slash" FORTH
(x1 -- x2)
x2 is the result of shifting x1 one bit toward the least-significant bit, leaving the most-significant bit unchanged.

2>R "two-to-r" FORTH
Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(x1 x2 --) (R: -- x1 x2)
Transfer cell pair x1 x2 to the return stack. Semantically equivalent to SWAP >R >R .
See also: >R 2R> 2R@ R> R@

2>S "two-to-s" STACK
(x1 x2 --)
(S: -- x1 x2)
Push a pair of numbers numbers on the auxiliary stack.

2@ "two-fetch" FORTH
(a-addr -- x1 x2)
Fetch the cell pair x1 x2 stored at a-addr. x2 is stored at a-addr and x1 at the next consecutive cell. It is equivalent to the sequence DUP CELL+ @ SWAP @ .
See also: 2!

2@X "two-fetch-x" EXTRA
(x-addr -- x1 x2)
Fetch the cell pair x1 x2 stored at extended address x-addr. x2 is stored at x-addr and x1 at the next consecutive cell. It is equivalent to the sequence 2DUP CELL+ @X -ROT @X .
See also: 2!X

2CONSTANT "two-constant" FORTH
(x1 x2 "name" --)
Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. name is referred to as a "two-constant."

name Execution: (-- x1 x2)
Place cell pair x1 x2 on the stack.

2DROP "two-drop" FORTH
(x1 x2 --)
Drop cell pairs x1 x2 from the stack.

2DUP "two-dupe" FORTH
(x1 x2 -- x1 x2 x1 x2)

Duplicate cell pair x1 x2.

2LITERAL "two-literal" FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (x1 x2 --)

Append the run-time semantics defined below to the current definition.

Run-time: (-- x1 x2)

Place cell pair x1 x2 on the stack.

2NIP "two-nip" EXTRA

(x1 x2 x3 x4 -- x3 x4)

Drop the first cell pair below the top cell pair of the stack.

2OVER "two-over" FORTH

(x1 x2 x3 x4 -- x1 x2 x3 x4 x1 x2)

Copy cell pair x1 x2 to the top of the stack.

2R> "two-r-from" FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(-- x1 x2) (R: x1 x2 --)

Transfer the cell pair x1 x2 from the return stack. Semantically equivalent to R> R> SWAP .

See also: >R 2>R 2R@ R> R@

2R@ "two-r-fetch" FORTH

(-- x1 x2) (R: x1 x2 -- x1 x2)

Copy cell pair x1 x2 from the returnstack. Semantically equivalent to R> R> 2DUP >R >R SWAP .

See also: >R 2>R 2R> R> R@

2ROT "two-rote" FORTH

(x1 x2 x3 x4 x5 x6 -- x3 x4 x5 x6 x1 x2)

Rotate the top three cell pairs on the stack bringing cell pair x1 x2 to the top of the stack.

2S> "two-s-from" STACK

(-- x1 x2)

(S: x1 x2 --)

Pop a pair of numbers numbers from the auxiliary stack.

2SWAP "two-swap" FORTH

(x1 x2 x3 x4 -- x3 x4 x1 x2)

Exchange the two top cell pairs.

2VARIABLE "two-variable" FORTH

("name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. Reserve two consecutive cells in data space at an aligned address. name is referred to as a "two-variable."

name Execution: (-- a-addr)

a-addr is the address of the first (lowest address) cell of two consecutive reserved cells in data space. A program is responsible for initializing the contents of the reserved cells. See also: VARIABLE

: "colon" FORTH

(C: "name" -- colon-sys)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name called a "colon definition". Enter compilation state and start the current definition, producing colon-sys. Append the execution semantics given below to the current definition.

The execution semantics of name will be determined by the words compiled into the body of the definition. The current definition definition for name is not findable in the dictionary until it is ended. If the contents of the variable POSTFIX is true, name is not parsed from the input buffer but it is taken from the c-addr/u combination on the stack. Note that this is not an ANSI required feature and is thus not portable.

Initiation: (i*x -- i*x) (R: -- nest-sys)

Save nest-sys (a single-cell address) of the calling definition. The stack effects i*x represent arguments to name.

name Execution: (i*x -- j*x)
Execute the definition name. The stack effects i*x and j*x represent arguments to and results from name, respectively.
See also: DOER: DOES> [ ] ;CODE

:NONAME "colon-no-name" FORTH

(C: -- colon-sys) (S: -- xt)
Create an execution token xt, enter compilation state and start the current definition, producing colon-sys. Append the execution semantics below to the current definition.

The execution semantics of xt will be determined by the words compiled into the body of the definition. The definition can be executed later by xt EXECUTE .
colon-sys is the topmost item on the data stack.

Initiation: (i*x -- i*x) (R: -- nest-sys)
Save nest-sys (a single cell address) of the calling definition. The stack effects i*x represent arguments to xt.

xt Execution: (i*x -- j*x)
Execute the definition specified by xt. The stack effects i*x and j*x represent arguments to and results from xt, respectively.

See also: : DOES> ; ;CODE] [

; "semicolon" FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: colon-sys --)
Append the execution semantics defined below to the current definition. End the current definition, allow it to be found in the dictionary and enter interpretation state, consuming colon-sys. The data space pointer is left aligned.

Execution: (--) (R: nest-sys --)
Return to the calling definition specified by nest-sys.

;CODE ASSEMBLER

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues

exception -14 when an attempt is made to execute this word.

Compilation: (C: colon-sys --)
Append the execution semantics defined below to the current definition. End the current definition, consuming colon-sys, enter interpret state, add the ASSEMBLER word list to the search order and start interpreting the rest of the parse area and assemble machine code. If needed, refill the input buffer until END-CODE is processed.

Execution: (--) (R: nest-sys --)
Replace the execution semantics of the most recently defined word with the name execution semantics given below. Return control to the calling definition specified by nest-sys. An ambiguous condition exists if the most recently defined word was not defined with CREATE or a user-defined word that calls CREATE .

name Execution: (i*x -- j*x)
Perform the machine code sequence that was generated following ;CODE .
See also: DOERCODE DOES> END-CODE

< "less-than" FORTH
(n1 n2 -- flag)
flag is true if and only if n1 is less than n2.
See also: U<

<# "less-number-sign" FORTH
(--)
Initialize the pictured numeric output conversion process.
See also: # #> #S

<> "not-equals" FORTH
(x1 x2 -- flag)
flag is true if and only if x1 is not bit-for-bit the same as x2.

<NL "indent-backward" EXTRA
(--)
Decrement INDENT with eight and perform NL .

= "equals" FORTH
(x1 x2 -- flag)
flag is true if and only if x1 is bit-for-bit the same as x2.

```
>                  "greater-than"                        FORTH
    ( n1 n2 -- flag )
    flag is true if and only if n1 is greater than n2.
    See also: U>

><                 "flip"                                EXTRA
    ( x1 -- x2 )
    See: FLIP

>BODY              "to-body"                             FORTH
    ( xt -- a-addr )
    a-addr is the data field address corresponding to execution token
    xt. This is only valid for words defined via CREATE .

>CALL              "to-call"                             EXTRA
    ( xt1 -- xt2 )
    xt2 is the execution token of the DOES> part of the defining word
    of an execution token xt1.

>HEAD              "to-head"                             EXTRA
    ( xt -- dea )
    dea is the dictionary entry address that is associated with
    execution token xt. If this fails, dea is zero.

>IN                "to-in"                               FORTH
    ( -- a-addr )
    a-addr is the address of a cell containing the offset in
    characters from the start of the input buffer to the start of
    the parse area.

>NL                "indent-forward"                      EXTRA
    ( -- )
    Increment INDENT with eight and perform NL .

>NUMBER            "to-number"                           FORTH
    ( ud1 c-addr1 u1 -- ud2 c-addr2 u2 )
    ud2 is the unsigned result of converting the characters within
    the string specified by c-addr1 u1 into digits, using the number
    in BASE , and adding each into ud1 after multiplying ud1 by the
    number in BASE . Conversion continues left-to-right until a
    character that is not convertible, including any "+" or "-" is
    encountered or the string is entirely converted. c-addr2 is the
```

location of the first unconverted character or the first character past the end of the string if the string was entirely converted. u2 is the number of unconverted characters in the string. An ambiguous condition exists if ud2 overflows during the conversion.

>R "to-r" FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(x --) (R: -- x)
Move x to the return stack.
See also: R> R@ 2>R 2R> 2R@

>S "to-s" STACK

(x --) (S: -- x)
Push a number on the auxiliary stack.

>UPC "to-u-p-c" EXTRA

(char1 -- char2)
Convert char1 to uppercase.

? "question" FORTH

(a-addr --)
Display the value stored at a-addr.

?AT "question-at" EXTRA

(-- u1 u2)
Return the column u1 and row u2 of the cursor on the screen.

?CSP "question-c-s-p" EXTRA

(--)
Check the current depth of the stack with the one stored by !CSP Exception -29 will occur when they do not match.

?DEF "query-defined" EXTRA

("name" -- flag)
Skip leading space delimiters. Parse name delimited by a space. Find name. If name is found, flag is true, false otherwise.
See also: ?UNDEF

?DO "question-do" FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: -- do-sys)
Place do-sys on the control flow stack. Append the execution semantics given below the current definition. The semantics are incomplete until resolved by a consumer of do-sys such as LOOP .

Execution: (n1|u1 n2|u2 --) (R: -- loop-sys)
If n1|u1 is equal to n2|u2, continue execution at the location given by the consumer of do-sys. Otherwise set up loop control parameters with index n2|u2 and limit n1|u1 and continue executing immediately following ?DO . Anything already on the return stack becomes unavailable until the loop control parameters are discarded. An ambiguous condition exists if n1|u1 and n2|u2 are not both of the same type.
See also: +LOOP DO I LEAVE LOOP UNLOOP

?DUP "question-dupe" FORTH
(x -- 0 | x x)
Duplicate x if it is non-zero.

?ERROR "question-error" EXTRA
(x n --)
If x is not zero, exception n occurs. Else drop both numbers from the stack and continue.

?HEAD "query-head" EXTRA
(dea --)
If the remaining of the current line is less than sixteen, perform CR . When the cursor is not a at column dividable by 16, emit spaces until the column is dividable by 16. Display the name associated with the dictionary entry address dea and wait for the time in milliseconds in WORDSPEED .

?LEAVE "question-leave" EXTRA
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(flag --) (R: loop-sys -- | loop-sys)
If flag is true, discard the current loop control parameters. An

ambiguous condition exists if they are unavailable. Continue execution immediately following the innermost syntactically enclosing DO ... LOOP or DO ... +LOOP . Otherwise continue.
See also LEAVE LOOP

?PAIRS "question-pairs" EXTRA
(x1 x2 --)
Check x1 and x2. Exception -22 occurs when they are not equal.

?STACK "question-stack" EXTRA
(--)
Check the three stack pointers and when they are too low or too high, exception -3, -4, -5, -6, -522 or -523 will occur.

?UNDEF "query-undefined" EXTRA
("name" -- flag)
Skip leading space delimiters. Parse name delimited by a space. Find name. If name is found, flag is false, true otherwise.
See also: ?DEF

@ "fetch" FORTH
(a-addr -- x)
Fetch x, x is the value stored at a-addr.

@+ "fetch-plus" EXTRA
(a-addr1 -- a-addr2 x)
Fetch x from a-addr1. Add 1 CELLS to a-addr1 giving a-addr2.

@X "fetch-x" EXTRA
(x-addr -- x)
Fetch x, x is the value stored at extended address x-addr.

ABORT FORTH
(i*x --) (R: j*x --)
Perform the function of -1 THROW . When no other exception frame is present other than the one pushed by QUIT , empty the stacks and perform QUIT . When no file is currently open, display no message. Otherwise, contrary to the Standard, display some information about the file and the line where ABORT was called. Store a zero-length string in ERR$.

ABORT" "abort-quote" FORTH
Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("ccc<quote>" --)
Parse characters ccc delimited by " (double-quote). Append the run-time semantics specified below to the current definition.

Run-time: (i*x x1 -- | i*x) (R: j*x -- | j*x)
Remove x1 from the stack. If any bit of x1 is not zero, perform the function of -2 THROW . The default interpreter will display ccc. The address of the counted string ccc can be found in ERR$, but is only valid for a limited time.

ABS "abs" FORTH
(n -- u)
u is the absolute value of n.

ACCEPT FORTH
(c-addr +n1 -- +n2)
Receive a string of at most +n1 characters. An ambiguous condition exists if +n1 is zero or greater than 32767. Display graphic characters as they are received. A Standard Program that depends on the presence or absence of non-graphic characters has an environmental dependancy. The editing functions, if any, that the system performs in order to construct the string are implementation defined.

Input terminates when "return" is received. When "return" is received, nothing is appended to the string, and the display is maintained in an implementation defined way.

+n2 is the length of the string stored at c-addr.

ADR "a-d-r" EXTRA
Interpretation: ("name" -- a-addr)
Skip leading space delimiters. Parse name delimited by a space. a-addr is the data field address of name. Exception -32 occurs if name was not defined by VALUE .

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VALUE

Run-time: (-- a-addr)
a-addr is the data field address of name.

AGAIN FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: dest --)
Append the execution semantics given below to the current definition, resolving the backward reference dest.

Execution: (--)
Continue execution at the location specified by dest. If no other control flow words are used, any program code after AGAIN will not be executed.
See also: BEGIN

AHEAD FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: -- orig)
Put the location of a new unresolved forward reference orig onto the control flow stack. Append the execution semantics given below to the current definition. The semantics are incomplete until orig is resolved (e.g., by THEN).

Execution: (--)
Continue execution at the location specified by the resolution of orig.

ALIAS EXTRA

(xt "name" --)
Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the semantics defined for the execution token xt. Attributes like IMMEDIATE and COMPILE-ONLY are not borrowed from xt.

ALIGN FORTH

(--)

If the address of the next available data space location is not aligned, reserve enough space to align it.

ALIGNED FORTH
(addr -- a-addr)
a-addr is the first aligned address greater than or equal to addr.

ALL DECOMPILER
(--)
Decompile all words in the context word list.
See also: STOP?

ALLOC EXTRA
(u1 -- u2 ior)
Allocate u1 paragraphs of memory outside the data space. The initial content of the allocated space is undefined. If no exception occurs u2 is the starting segment address of the allocated space and ior is zero. Otherwise u2 is unspecified and ior is the I/O result code.

ALLOT FORTH
(n --)
Reserve n address units of data space.

If the data space pointer is aligned and n is a multiple of the size of a cell when ALLOT begins execution, it will remain aligned when ALLOT finishes execution.

ALSO ONLY
(--)
Transform the search order consisting of wid1 .. widn-1 widn (where widn is searched first) into wid1 .. widn-1 widn widn. An ambiguous condition exists if there are too many word lists in the search order.

AND FORTH
(x1 x2 -- x3)
x3 is the bit-by-bit logical "and" of x1 with x2.

ANOTHER EXTRA
(-- dea true | false)
Return the next dea in the word list. Used in words as WORDS .

This word depends on the stored wid at TEMPORARY . When ANSI does not contain zero, only words marked with ANS are returned.

ANS EXTRA

(--)

Mark the most recently created definition as a standard word. When the variable ANSI does not contain zero, the default interpreter issues a warning if words that are not marked are interpreted or compiled.

ANSI EXTRA

(-- a-addr)

a-addr is the address of a cell containing true when messages will be given if non-standard words are encountered and false otherwise.

ANY SEARCHER

("ccc" --)

Skip leading space delimiters. Parse ccc delimited by a space. Search the files with extension given by HEXT$ in the directory given by HELPPATH . Display the description of the names that contain ccc. If a full screen is displayed, wait for the user to press a key. Stop if the key is the escape key.

APPEND EXTRA

(c-addr1 u c-addr2 --)

Add u to the numerical value of the character at c-addr2. Store the string specified by c-addr1 u at the character address given by the sum of c-addr2 and the incremented numerical value of the character at c-addr2.

APPEND-CHAR EXTRA

(char c-addr --)

Increment the numerical value of the character at c-addr by one. Store char at the character address given by the sum of the incremented numerical value of the character at c-addr and c-addr.

ASSEMBLER ASSEMBLER

(--)

Replace the first word list in the search order with the

ASSEMBLER word list.

AT-XY "at-x-y" FORTH
(u1 u2 --)
Perform steps so that the next character displayed will appear in column u1, row u2 of the current output device, the upper left corner of which is row zero, column zero. It is a no-op when the operation cannot be performed on the current output device with the specified parameters. Note that for other implementations the result in that case is an ambiguous condition.

ATEXIT EXTRA
(--)
A word that is executed when the program is terminated.

ATT0 "attribute-zero" EXTRA
(-- a-addr)
a-addr is the address of a cell containing the default attribute of the characters on the screen.

ATTR "attribute" EXTRA
(-- a-addr)
a-addr is the address of a cell containing the current attribute of the characters on the screen.

B. "b-dot" EXTRA
(u --)
Display u as a two digit hexadecimal number with a trailing space.
See also: H.

BASE FORTH
(-- a-addr)
a-addr is the address of a cell containing the current number conversion radix {{2..36}}.

BEEP EXTRA
(--)
Make an alarm sound on the speaker. As this is sometimes irritating, try CLICK .

BEEPH EXTRA
(-- a-addr)

```
    a-addr is the address of a cell containing the frequency in Hertz
    of BEEP.

BEEPL                                                           EXTRA
    ( -- a-addr )
    a-addr is the address of a cell containing the duration in
    milliseconds of BEEP.

BEGIN                                                           FORTH
    Interpretation: ( i*x -- )
    This word is marked compile only. The default interpreter issues
    exception -14 when an attempt is made to execute this word.

    Compilation: ( C: -- dest )
    Put the next location for a transfer of control, dest, onto the
    control flow stack.

    Execution: ( -- )
    Continue execution.
    See also: REPEAT UNTIL WHILE

BIN                                                             FORTH
    ( x1 -- x2 )
    Modify the file access method x1 to additionally select a
    "binary", i.e. not line oriented, file access method, giving
    access method x2.
    See also: R/O R/W W/O

BIOS-IO                                                         EXTRA
    ( -- )
    Set input and output to fast BIOS routines, redirection is not
    supported.
    See also: MS-DOS-IO

BIOS?              "bios-query"                                 EXTRA
    ( -- x )
    A value that is true when output goes via fast BIOS and not
    via slow DOS.

BL                 "b-l"                                        FORTH
    ( -- char )
    char is the character value for a space.
```

BLANK FORTH

(c-addr u --)

If u is greater than zero, store the character value for space in u consecutive character positions beginning at c-addr.

BLINK EXTRA

(--)

Invert the blink character attribute.

BLK "b-l-k" FORTH

(-- a-addr)

a-addr is the address of a cell containing zero or the number of the mass-storage block being interpreted. If BLK contains zero, the input source is not a block and can be identified by SOURCE-ID . A program may not directly alter the contents of BLK

BLOCK FORTH

(u -- a-addr)

a-addr is the address of the first character of the block buffer assigned to mass-storage block u. Exceptions -33, -34 or -35 will occur if u is not an available block number.

If block u is already in a block buffer: a-addr is the address of that block buffer.

If block u is not already in memory and there is an unassigned block buffer: transfer block u from mass-storage to an unassigned block buffer. a-addr is the address of that block buffer.

If block u is not already in memory and there are no unassigned block buffers: unassign a block buffer. If the block in that buffer has been UPDATEd, transfer the block to mass-storage and transfer block u from mass storage into the buffer. a-addr is the address of that block buffer.

At the conclusion of the operation the block buffer pointed to by a-addr is the current block buffer and is assigned to u.

BLOCK-CURSOR EXTRA

(--)

Set the cursor form to a block.

BODY> "body-from" EXTRA
(a-addr -- xt)
xt is execution token corresponding to the data field address a-addr. This is only valid for a word defined via CREATE .

BOUNDS EXTRA
(n1|u1 n2|u2 -- n3|u3 n1|u1)
Add n1|u1 to n2|u2 giving n3|u3. Used for setting up DO LOOPs.

BRIGHT EXTRA
(--)
Invert the bright character attribute.

BTW DECOMPILER
("name1" "name2" --)
Skip leading space delimiters. Parse name1 delimited by a space. Skip leading space delimiters. Parse name2 delimited by a space. Find name1. Find name2. If any name can not be found exception -13 occurs. Otherwise decompile all the words in the current word list between name1 inclusive and name2 inclusive starting with the last compiled. The order of name1 and name2 is indifferent.
See also: STOP?

BUFFER FORTH
(u -- a-addr)
a-addr is the address of the first character of the block buffer assigned to u. The contents of the block are unspecified. Exceptions -34 or -35 will occur if u is not an available block number.

If block u is already in a block buffer: a-addr is the address of that block buffer.

If block u is not already in memory and there is an unassigned block buffer. a-addr is the address of that block buffer.

If block u is not already in memory and there are no unassigned block buffers: unassign a block buffer. If the block in that buffer has been UPDATEd, transfer the block to mass-storage. a-addr is the address of that block buffer.

At the conclusion of the operation the block buffer pointed to by a-addr is the current block buffer and is assigned to u.
See also: BLOCK

BYE FORTH
(--)
Terminate the Forth program and return to the operating system with returncode zero.

BYTES EXTRA
(-- a-addr)
a-addr is the address of a cell containing the dictionary pointer at the last execution of SAVE or EXTEND .

C! "c-store" FORTH
(char c-addr --)
Store char at c-addr.

C!X "c-store-x" EXTRA
(c x-addr --)
Store char at extended address x-addr.

C" "c-quote" FORTH
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("ccc<quote>" --)
Parse ccc delimited by " (double-quote). Append the run-time semantics given below to the current definition.

Run-time: (-- c-addr)
Return c-addr, a counted string consisting of the characters ccc.
A standard program shall not alter the returned string.
See also: S"

C+! "c-plus-store" EXTRA
(char c-addr --)
Add char to the character at c-addr.

C+!X "c-plus-store-x" EXTRA
(char x-addr --)
Add char to the character at extended address x-addr.

```
C,                 "c-comma"                                  FORTH
    ( char -- )
    Reserve space for one character in the data space and store char
    in the space.

C/L                "c-per-l"                                  EXTRA
    ( -- n )
    Return the number of characters on a screen line.

C0!                "c-zero-store"                             EXTRA
    ( c-addr -- )
    Clear all bits of the character at c-addr.

C@                 "c-fetch"                                  FORTH
    ( c-addr -- char )
    Fetch the character stored at c-addr.

C@+                "c-fetch-plus"                             EXTRA
    ( c-addr1 -- c-addr2 char )
    Fetch char from c-addr1 and add 1 CHARS to c-addr1 giving
    c-addr2.

C@X                "c-fetch-x"                                EXTRA
    ( x-addr -- char )
    Fetch the character stored at extended address x-addr.

CALL,              "call-comma"                               EXTRA
    ( a-addr -- )
    Compile an assembler language call in the dictionary at the
    data-space pointer to the address on the stack and increment the
    data-space pointer to an aligned address after the instruction.

CASE                                                          FORTH
    Interpretation: ( i*x -- )
    This word is marked compile only. The default interpreter issues
    exception -14 when an attempt is made to execute this word.

    Compilation: ( C: -- case-sys )
    Mark the start of the CASE ... OF ... ENDOF ... ENDCASE
    structure.

    Execution: ( -- )
    Continue execution.
```

See also: ENDCASE ENDOF OF

CASESENSITIVE EXTRA
(-- a-addr)
a-addr is the address of a cell containing false when the case of characters is to be ignored and true when case is significant.

CAT$ EXTRA
(-- c-addr)
c-addr is the address of a counted string containing a description of the category to which this file belongs.

CATCH FORTH
(i*x xt -- j*x 0 | i*x n)
Push an exception frame on the exception stack and then execute the execution token xt (as with EXECUTE) in such a way that control can be transferred to a point just after CATCH if THROW is executed during the execution of xt.

If the execution of xt completes normally (i.e. the exception frame pushed by this CATCH is not popped by an execution of THROW) pop the execution frame and return zero on top of the data stack, above whatever stack items would have been returned by xt EXECUTE . Otherwise, the remainder of the execution semantics are given by THROW .

CD "change-dir" EXTRA
("ccc" --)
Skip leading space delimiters. Parse ccc delimited by a space. When ccc is the null string, display the current directory. Else change to the directory ccc. Contrary to DOS, when a drive letter and a colon are in front of the string, that drive will also be made current.

CELL+ "cell-plus" FORTH
(a-addr1 -- a-addr2)
Add the size in address units of a cell to a-addr1 giving a-addr2.

CELL- "cell-minus" EXTRA
(a-addr1 -- a-addr2)
Subtract the size in address units of a cell from a-addr1 giving a-addr2.

```
CELLS                                                   FORTH
    ( n1 -- n2 )
    n2 is the size in address units of n1 cells.

CFG               "c-f-g"                               EXTRA
    ( -- c-addr )
    c-addr is the address of a counted string containing the name of
    the configuration file.

CHAIN                                                   EXTRA
    Interpretation: ( i*x -- )
    This word is marked compile only. The default interpreter issues
    exception -14 when an attempt is made to execute this word.

    Compilation: ( "name" -- )
    Skip leading space delimiters. Parse name delimited by a space.
    Append the current execution semantics of name to the current
    definition. Exception -32 occurs if name was not defined by
    VECTOR .

CHAR              "char"                                FORTH
    ( "name" -- char )
    Skip leading space delimiters, Parse name delimited by a space.
    Put the value of its first character on the stack.
    See also: [CHAR]

CHAR+             "char-plus"                           FORTH
    ( c-addr1 -- c-addr2 )
    Add the size in address units of a character to c-addr1 giving
    c-addr2.

CHAR-             "char-minus"                          EXTRA
    ( c-addr1 -- c-addr2 )
    Subtract the size in address units of a character from c-addr1
    giving c-addr2.

CHARS             "chars"                               FORTH
    ( n1 -- n2 )
    n2 is the size in address units of n1 characters.

CHOOSE                                                  EXTRA
    ( u1 -- u2 )
```

u2 is a random number less than u2.

CIRCULAR EXTRA
(n1 n2 -- n3)
Divide n1 by n2, giving the single-cell quotient n3. Exception -10 is issued if n1 is zero. If n1 and n2 differ in sign the result returned will be the same as returned by the phrase >R S>D R> FM/MOD DROP .

CLEAR EXTRA
Interpretation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Store zero in name. Exception -32 occurs if name was not defined by VALUE or VARIABLE .

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VALUE , VARIABLE or (LOCAL).

Run-time: (--)
Store zero in name.

CLICK EXTRA
(--)
Make a more pleasant sort of BEEP.

CLOCKOFF CLOCK
(--)
Do not show a clock on the screen.

CLOCKON CLOCK
(--)
Display a clock in the upper right corner of the screen.

CLOSE-FILE FORTH
(fileid -- ior)
Close the file identified by fileid, ior is the I/O result code.

CLOSE-LOG LOG
(--)
Close the logfile.

CMOVE "c-move" FORTH
(c-addr1 c-addr2 u --)
If u is greater than zero, copy u consecutive characters, character-by-character and left-to-right, from c-addr1 to c-addr2. If c-addr2 lies within the source region, memory propagation occurs. (c-addr2 lies within the source region if c-addr2 is not less than c-addr1 and c-addr2 is less than the quantity c-addr1 u CHARS +).
See also: CMOVE> MOVE

CMOVE> "c-move-up" FORTH
(c-addr1 c-addr2 u --)
If u is greater than zero, copy u consecutive characters, character-by-character and right-to-left, from c-addr1 to c-addr2. If c-addr1 lies within the destination region, memory propagation occurs. (c-addr1 lies within the destination region if c-addr1 is greater than or equal to c-addr2 and if c-addr2 is less than the quantity c-addr1 u CHARS +).
See also: CMOVE MOVE

CMOVEX "c-move-x" EXTRA
(x-addr1 x-addr2 u --)
If u is greater than zero, copy u consecutive characters, character-by-character and left-to-right, from extended address x-addr1 to extended address x-addr2. If x-addr2 lies within the source region, memory propagation occurs. (x-addr2 lies within the source region if x-addr2 is not less than x-addr1 and x-addr2 is less than the quantity x-addr1 u CHARS +).
See also: CMOVE CMOVEX>

CMOVEX> "c-move-x-up" EXTRA
(x-addr1 x-addr2 u --)
If u is greater than zero, copy u consecutive characters, character-by-character and right-to-left, from extended address x-addr1 to extended address x-addr2. If x-addr2 lies within the source region, memory propagation occurs. (x-addr2 lies within the source region if x-addr2 is not less than x-addr1 and x-addr2 is less than the quantity x-addr1 u CHARS +).
See also: CMOVE CMOVEX

CODE ASSEMBLER
("name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name, called a "code definition", with the execution semantics defined below. Add the ASSEMBLER word list to the search order and start interpreting the rest of the parse area and assemble machine code. If needed, refill the input buffer until END-CODE is processed.

name Execution: (i*x -- j*x)
Execute the machine code sequence that was generated following CODE .
See also: END-CODE

COLD EXTRA
(--)
Restart the system. This is always done when a program starts executing from DOS when it was saved by SAVE . The first time it will process the command tail. Otherwise QUIT is performed.

COMPARE FORTH
(c-addr1 u1 c-addr2 u2 -- flag)
Compare the string specified by c-addr1 u2 to the string specified by c-addr2 u2. The strings are compared, beginning at the given addresses, character by character, up to the length of the shorter string or until a difference is found. If the two strings are identical up to the length of the shorter string, n is zero if both strings are of equal length, minus-one of u1 is less than u2, and one otherwise. If the two strings are not identical up to the length of the shorter string, n is minus-one if the first non-matching character in the string specified by c-addr1 u1 has a lesser numerical value than the corresponding character in the string specified by c-addr2 u2 and one otherwise.
See also: COMPARE-UPPERCASE

COMPARE-UPPERCASE EXTRA
(c-addr1 u1 c-addr2 u2 -- flag)
Compare the string specified by c-addr1 u2 to the string specified by c-addr2 u2. The strings are compared, beginning at the given addresses, character by character, up to the length of the shorter string or until a difference is found. If the two strings are identical, where lower case characters are considered equal to upper case characters, up to the length of the shorter string, n is zero if both strings are of equal length, minus-one

of u1 is less than u2, and one otherwise. If the two strings are not identical up to the length of the shorter string, n is minus-one if the first non-matching character in the string specified by c-addr1 u1 has a lesser numerical value, where the value of lower case characters are converted to their upper case equivalent values without affecting the strings themselves, than the corresponding character in the string specified by c-addr2 u2 and one otherwise.

See also: COMPARE

COMPILE, "compile-comma" FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Execution: (xt --)

Append the execution semantics of the definition represented by xt to the execution semantics of the current word definition.

COMPILE-ONLY EXTRA

(--)

Mark the most recently created definition as a compile-only word. The default interpreter issues exception -14 when an attempt is made to execute the definition in interpret state.

COMSPEC EXTRA

(-- c-addr)

c-addr is the address of a counted string containing the path and name of the command interpreter of DOS.

CONSOLE! "console-store" EXTRA

(char --)

Write char to the standard output file.

CONSOLE? "console-query" EXTRA

(-- x)

A value that is true when screen output is enabled.

CONSOLE@ "console-fetch" EXTRA

(-- char | -1)

Read character char from the standard input file. If the end of the file is reached, return -1.

CONSTANT FORTH

(x "name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. name is referred to as a "constant."

name Execution: (-- x)

Place x on the stack.

CONVERT OBSOLETE

(ud1 c-addr1 -- ud2 c-addr2)

ud2 is the result of converting the characters within the text beginning at the first character after c-addr1 into digits, using the number in BASE , and adding each digit to ud1 after multiplying by the number in BASE . Conversion continues until a character that is not convertible is encountered. c-addr2 is the location of the first unconverted character. An ambiguous condition exists if ud2 overflows.

Note: this word is obsolescent and is included as a concession to existing implementations. Its function is superseded by >NUMBER .

COUNT FORTH

(c-addr1 -- c-addr2 char)

Return the character string specification for the counted string stored at c-addr1. c-addr2 is the address of the first character after c-addr1. u is the contents of the character at c-addr1, which is the length in characters of the string at c-addr2.

COUNTX "count-x" EXTRA

(x-addr1 -- x-addr2 char)

Fetch char from extended address x-addr1 and add 1 CHARS to x-addr1 giving x-addr2.

CR "c-r" FORTH

(--)

Cause subsequent output to appear at the beginning of the next line.

CREAT$ EXTRA

(-- c-addr)

c-addr is the address of a counted string containing the name of the creator of this file.

CREATE FORTH

("name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. If the data-space pointer is not aligned, reserve enough data space to align it. The new data-space pointer defines name's data field. CREATE does not allocate data space in name's data field.

name Execution: (-- a-addr)

a-addr is the address of name's data field. The execution semantics of name may be extended by using DOES> or ;CODE .

See also: DOES>

CREATE-FILE FORTH

(c-addr u x1 -- x2 ior)

Create the file named in the character string specified by c-addr and u, and open it with file access method x1. If a file with the same name already exists, recreate it as an empty file.

If the file was successfully created and opened, ior is zero, x2 is the fileid, and the file has been positioned at the start of the file.

Otherwise ior is the I/O result code and x2 is an unspecified value.

CS-PICK "c-s-pick" FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(C: destu ... orig0|dest0 -- destu ... orig0|dest0 destu)

(S: u --)

Remove u. Copy destu to the top of the control-flow stack. An ambiguous condition exists if there are less than u+1 items, each of which shall be an orig or dest, on the control-flow stack before CS-PICK is executed.

The control-flow stack in CHForth is implemented on the data stack, u is the topmost item on the data stack.

CS-ROLL "c-s-roll" FORTH
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(C: origu|destu origu-1|destu-1 ... orig0|dest0 --
origu-1|destu-1 ... orig0|dest0 origu|destu)
(S: u --)
Remove u. Rotate u+1 elements on top of the control-flow stack so that origu|destu is on top of the control-flow stack. An ambiguous condition exists if there are less than u+1 items, each of which shall be an orig or dest, on the control-flow stack before CS-ROLL is executed.
The control-flow stack in CHForth is implemented on the data stack, u is the topmost item on the data stack.

CSEG EXTRA
(-- x)
x is the value of the combined code and data segment.

CTRL "control" EXTRA
("name" -- char)
Skip leading space delimiters, Parse name delimited by a space. Put the value of the control character defined by its first character on the stack. Exception -531 occurs when the character is not in the range {’@’..’_’}.
See also: CHAR [CTRL]

CURRENT-DIRECTORY EXTRA
(-- c-addr)
c-addr is the address of a counted string containing the name of the current DOS directory.

D* "d-star" EXTRA
(d1|ud1 d2|ud2 -- d3|ud3)
Multiply d1|ud1 by d2|ud2 giving product d3|ud3.

D+ "d-plus" FORTH
(d1|ud1 d2|ud2 -- d3|ud3)
Add d2|ud2 to d1|ud1, giving the sum d3|ud3.

D+! "d-plus-store" EXTRA
(d|ud a-addr --)

```
    Add d|ud to the double-cell number at a-addr.

D+!X              "d-plus-store-x"                      EXTRA
    ( d|ud x-addr -- )
    Add d|ud to the double-cell value at extended address x-addr.

D-                "d-minus"                             FORTH
    ( d1|ud1 d2|ud2 -- d3|ud3 )
    Subtract d2|ud2 from d1|ud1, giving the difference d3|ud3.

D.                "d-dot"                               FORTH
    ( d -- )
    Display d in free field format.

D.R               "d-dot-r"                             FORTH
    ( d n -- )
    Display d right aligned in a field n characters wide. If the
    number of characters required to display d is greater than n, all
    digits are displayed with no leading spaces in a field as wide as
    necessary.

D0!               "d-zero-store"                        EXTRA
    ( a-addr -- )
    Clear all bits of the double-cell value at a-addr.

D0<               "d-zero-less"                         FORTH
    ( d -- flag )
    flag is true if and only if d is less than zero.

D0=               "d-zero-equals"                       FORTH
    ( xd -- flag )
    flag is true if and only if xd is equal to zero.

D2*               "d-two-star"                          FORTH
    ( xd1 -- xd2 )
    xd2 is the result by shifting xd1 one bit toward the
    most-significant bit, filling the vacated least-significant bit
    with zero.

D2/               "d-two-slash"                         FORTH
    ( xd1 -- xd2 )
    xd2 is the result of shifting xd1 one bit toward the
    least-significant bit, leaving the most-significant bit
```

unchanged.

D< "d-less-than" FORTH
(d1 d2 -- flag)
flag is true if and only if d1 is less than d2.

D= "d-equals" FORTH
(xd1 xd2 -- flag)
flag is true if and only if xd1 is equal to xd2.

D> "d-greater-than" EXTRA
(d1 d2 -- flag)
flag is true if and only if d1 is greater than d2.

D>S "d-to-s" FORTH
(d -- n)
n is the equivalent of d. An overflow occurs if d lies outside the range of a signed single-cell number.

DABS "d-abs" FORTH
(d -- ud)
ud is the absolute value of d.

DATE EXTRA
(-- +n1 +n2 +n3)
Return the current date. +n1 is the day {1..31}, +n2 is the month {1..12}, and +n3 is the year (e.g. 1991).

DB ASSEMBLER
("ccc" --)
Assemble "ccc" as an 8 bit value.

DEALLOC EXTRA
(u -- ior)
Return the contiguous region of memory outside the data space indicated by the segment address u to the system for later allocation. u shall indicate a region of memory outside the data space that was previously obtained by ALLOC or REALLOC . If no exception occurs ior is zero. Otherwise ior is the I/O result code.

DEBUG TRACER

(-- a-addr)
A variable used in the tracer. When zero, no trace information is shown on the screen. Else a stack diagram is shown along with the name of the next to be executed word or the word that was executed by the compiler (immediate words). See TRACE .

DECIMAL FORTH
(--)
Set the numeric conversion radix to ten (decimal).

DECOMPILER DECOMPILER
(--)
Replace the first word list in the search order with the DECOMPILER word list.

DECR "decrement" EXTRA
(a-addr --)
Subtract 1 from the single-cell value at a-addr.

DEFINITIONS ONLY
(--)
Make the compilation word list the same as the first word list in the search order. Specifies that the names of subsequent definitions will be placed in the compilation word list. Subsequent changes in the search order will not effect the compilation word list.

DELETE-FILE FORTH
(c-addr u -- ior)
Delete the file named in the character string specified by c-addr u. ior is the I/O result code.

DEPRIVE EXTRA
(--)
Hide all the words that are marked with PRIVATE .

DEPTH FORTH
(-- +n)
+n is the number of single-cell values on the data stack before +n was placed on the stack.

DFTMODE "default-mode" EXTRA
(--)

Set the screen to the text mode that was current at program start.

DIAGNOSE EXTRA
(--)
Display some information over the compiled bytes since processing the command tail.

DIGIT EXTRA
(char +n -- n1 true | char false)
Try to convert char to a digit n1 with number base +n. If the conversion succeeds, return a true flag. Otherwise a false flag.

DIS "disassemble" DISASSEM
(addr --)
Disassemble from address addr.

DISASSEMBLER DISASSEM
(--)
Replace the first word list in the search order with the DISASSEMBLER word list.

DISPOSE LOADHIGH
(--)
Reclaim the temporary space where the file loaded with LOADHIGH was compiled. All the words loaded with LOADHIGH are no longer available to the Forth system. Be sure not to use references to the words in that file.
See also: LOADHIGH MARK

DISX "dis-extended" DISASSEM
(x-addr --)
Disassemble from extended address x-addr.

DLOCAL DLOCALS
(d "name" --)
Create a dictionary entry with name "name" and initial value d.

Executing:
(-- d)
Place the value on the stack. The value can be manipulated by TO +TO and CLEAR .

DMAX "d-max" FORTH

(d1 d2 -- d3)

d3 is the greater of d1 and d2.

DMIN "d-min" FORTH

(d1 d2 -- d3)

d3 is the lesser of d1 and d2.

DNEGATE "d-negate" FORTH

(d1 -- d2)

d1 is the negation of d1.

DO FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: -- do-sys)

Place do-sys on the control flow stack. Append the execution semantics given below the current definition. The semantics are incomplete until resolved by a consumer of do-sys such as LOOP .

Execution: (n1|u1 n2|u2 --) (R: -- loop-sys)

Set up loop control parameters with index n2|u2 and limit n1|u1. An ambiguous condition exists if n1|u1 and n2|u2 are not both the same type. Anything already on the return stack becomes unavailable until the loop control parameters are discarded.

See also: +LOOP LOOP

DOC EXTRA

(--)

Repeatedly skip leading spaces, parse and discard space-delimited words from the parse area, until the word ENDDOC has been parsed and discarded. If the parse area becomes exhausted, it is refilled as with REFILL . DOC is immediate.

An ambiguous condition exists if DOC is POSTPONEd. If the end of the input stream is reached and cannot be refilled before the terminating ENDDOC is parsed, exception -532 occurs.

DOER: EXTRA

("name" --) (C: -- colon-sys)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the run-time semantics defined below. Enter compilation state, and start current definition.

Run-time: (--) (R: nest-sys1 --)
Replace the execution semantics of the most recent definition, referred to as name, with the name execution semantics given below. Return control to the calling definition specified by nest-sys1. Code may be damaged if the most recently defined word was not defined with CREATE or a user-defined word that calls CREATE .

Initiation: (i*x -- i*x a-addr) (R: -- nest-sys2)
Save implementation-dependant information nest-sys2 about the calling definition. Place name's data field address on the stack. the stack effects i*x represents the arguments to name.

name Execution: (i*x -- j*x)
Execute the portion of the definition that begins with the initiation semantics appended by the DOES> which modifies name. The stack effects i*x and j*x represent arguments to and results from name, respectively.
See also: CREATE DOES>

DOERCODE ASSEMBLER

("name" --)
Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. Enter interpret state, add the ASSEMBLER word list to the search order and start interpreting the rest of the parse area and assemble machine code. If needed, refill the input buffer until END-CODE is processed.

Execution: (--) (R: nest-sys --)
Replace the execution semantics of the most recently defined word with the name execution semantics given below. Return control to the calling definition specified by nest-sys. An ambiguous condition exists if the most recently defined word was not defined with CREATE or a user-defined word that calls CREATE .

name Execution: (i*x -- j*x)
Perform the machine code sequence that was generated following DOERCODE .
See also: DOES> END-CODE

DOES> "does" FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: colon-sys1 -- colon-sys2)
Append the run-time semantics below to the current definition. The current definition is not made findable by DOES> . Consume colon-sys1 and produce colon-sys2. Append the initiation semantics defined below to the current definition.

Run-time: (--) (R: nest-sys1 --)
Replace the execution semantics of the most recently definition, referred to as name, with the name execution semantics given below. Return control to calling definition specified by nest-sys1. Code may be damaged if the most recently defined word was not defined with CREATE or a user-defined word that calls CREATE .

Initiation: (i*x -- i*x a-addr) (R: -- nest-sys2)
Save implementation-dependant information nest-sys2 about the calling definition. Place name's data field address on the stack. The stack effects i*x represent arguments to name.

name Execution: (i*x -- j*x)
Execute the portion of the definition that begins with the initiation semantics appended by DOES> which modified name. The stack effects i*x and j*x represent arguments to and results from name, respectively.
See also: CREATE DOER:

DOS: "dos-colon" EXTRA

(c-addr u "name" --)
Skip leading space delimiters. Parse name delimited by a space. Create a dictionary entry for name with the execution semantics defined below.

name Executing: ("ccc" --)
Execute the DOS command specified by the character string c-addr u and parameters ccc, terminated by the end of the line or the character in SEPARATOR .

DPL "d-p-l" EXTRA

(-- a-addr)

a-addr is the address of a cell. When the last interpreted number contained a decimal point, it will contain the number of digits after the decimal point in that number; otherwise the contents are -1.

DRIVE EXTRA

(n "name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a dictionary entry for name with the execution semantics defined below.

name Executing: (--)

Change the default drive number to n, n is zero for drive A:.

DROP FORTH

(x --)

Remove x from the stack.

DU< "d-u-less" FORTH

(ud1 ud2 -- flag)

flag is true if and only if ud1 is less than ud2.

DU> "d-u-greater" EXTRA

(ud1 ud2 -- flag)

flag is true if and only if ud1 is greater than ud2.

DUMP FORTH

(addr u --)

Display the contents of u consecutive addresses starting at addr. At the beginning of the line the address is displayed, preceded with the name of the segment, followed with the hexadecimal contents of 16 characters and then the same characters are displayed with SEMIT .

DUMP is implemented using pictured numeric output words. Its use will corrupt the transient region identified by #> .

DUMPX "dump-extended" EXTRA

(x-addr u --)

Display the contents of u consecutive addresses starting at extended address x-addr. At the beginning of the line the extended address is displayed, followed with the hexadecimal contents of 16 characters and then the same characters are displayed with SEMIT .

DUMPX is implemented using pictured numeric output words. Its use will corrupt the transient region identified by #> .
See also: DUMP

DUP "dupe" FORTH
(x -- x x)
Duplicate x.

DUP>R "dupe-to-r" EXTRA
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(x -- x) (R: -- x)
Copy x to the return stack.

DUP>S "dupe-to-s" STACK
(x -- x)
(S: -- x)
Duplicate a number and push it on the auxiliary stack.

DW ASSEMBLER
("ccc" --)
Assemble "ccc" as a 16 bit value.

ECHO EXTRA
(--)
When loading echo the lines read to the screen.

ECHO? "echo-query" EXTRA
(-- x)
A value that is true when characters are echoed during loading a text file.

EDIT EDITOR
("name" --)
Skip leading space delimiters. Parse name delimited by a space. Open file name with the editor program and place the cursor at

the first line. When name is omitted, the last opened file by this command or ,EDIT LIST or WHAT is opened and name is displayed on the right of the status line. The default extension is taken from FEXT$.

EDITOR ONLY

(--)

Make the EDITOR word list the first word list to be searched. This word list contains CHForth specific extensions to the ANSI standard for the line input editor and the block editor. Note that these words are non-standard.

EDLIB EDITOR

("name" --)

Skip leading space delimiters. Parse name delimited by a space. Open file name in the directory given in LIBPATH with the editor program and place the cursor at the first line. The default extension is taken from FEXT$.

EKEY "e-key" FORTH

(-- u)

Receive one keyboard event u. ASCII keys have bits 7 to 15 set to zero; other keys have the scan code in bits 8 to 15 and the lower bits set to zero. Key codes made by holding the ALT-key down and using the numeric pad give a 8 bit code.

EKEY>CHAR "e-key-to-char" FORTH

(u -- u false | char true)

If the keyboard event u corresponds a valid 8 bit character, return that character and true, otherwise return u and false.

EKEY? "e-key-question" FORTH

(-- flag)

If a keyboard event is available, returns true. Otherwise returns false.

After EKEY? returns with a value of true, subsequent executions of EKEY? prior to the execution of KEY , KEY? or EKEY also return true, referring to the same event. The next execution of EKEY will return the same event without indefinite delay.

ELEN EXTRA

(-- n)

n is the number of paragraphs in the environment segment.

ELSE FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: orig1 -- orig2)
Put the location of a new unresolved forward reference orig2 onto the control flow stack. Append the execution semantics given below to the current definition. The semantics will be incomplete until orig2 is resolved (e.g. by THEN). Resolve the forward reference orig1 using the location following the appended execution semantics.

Execution: (--)
Continue execution at the location given by the resolution of orig2.
See also: IF THEN

EMIT FORTH

(x --)
If x is a graphic character in the implementation-defined character set, display x. The effect of EMIT for all other values of x is implementation-defined.

Standard programs that use control characters to perform specific functions have an environmental dependency. Each EMIT deals with one character.
See also: TYPE

EMIT? "emit-question" FORTH

(-- flag)
flag is true if the user output device is ready to accept data and the execution of EMIT in place of EMIT? would not have suffered an indefinite delay. If the device status is indeterminate, flag is true.

EMPTY EXTRA

(--)
Perform the function of FORGET on all definitions that were compiled after the last execution of EMPTY , EXTEND or SAVE .

```
EMPTY-BUFFERS                                               FORTH
   ( -- )
   Unassign all block buffers. Do not transfer the contents of any
   UPDATEd block buffer to mass storage.
   See also: BLOCK

END-CODE                                                    ASSEMBLER
   ( -- )
   Resolve all assembler labels, terminate the current code
   definition and allow its name to be found in the dictionary.
   Remove the ASSEMBLER word list from the search order.

END-LOCAL                                                   EXTRA
   Interpretation: ( i*x -- )
   This word is marked compile only. The default interpreter issues
   exception -14 when an attempt is made to execute this word.

   Compilation: ( -- )
   Terminate creation of local values.

END-METHODS                                                 EXTRA
   ( -- )
   Terminate defining methods.
   See also: METHODS

ENDCASE                                                     FORTH
   Interpretation: ( i*x -- )
   This word is marked compile only. The default interpreter issues
   exception -14 when an attempt is made to execute this word.

   Compilation: ( C: case-sys -- )
   Mark the end of the CASE ... OF ... ENDOF ... ENDCASE structure.
   Use case-sys to resolve the entire structure. Append the
   execution semantics given below to the current definition.

   Execution: ( x -- )
   Discard the case selector x and continue execution.
   See also: CASE ENDOF OF

ENDIF                                                       EXTRA
   ( orig -- )
   Interpretation: ( i*x -- )
   This word is marked compile only. The default interpreter issues
```

exception -14 when an attempt is made to execute this word.

Compilation: (C: orig --)
Resolve the forward reference orig using the location of the execution semantics.

Execution: (--)
Continue execution.
See also: ELSE IF THEN

ENDOF FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: case-sys1 of-sys -- case-sys2)
Mark the end of the ... OF ... ENDOF ... part of the CASE structure. The next location for a transfer of control resolves the reference given by of-sys. Append the execution semantics given below to the current definition. Replace case-sys1 with case-sys2 on the control flow stack, to be resolved by ENDCASE .

Execution: (--)
Continue execution at the location specified by the consumer of case-sys2.
See also: CASE ENDCASE OF

ENVIRONMENT? "environment-query" FORTH

(c-addr u -- false | i*x true)
c-addr is the address of a character string and u is the string's character count. u may have a value in the range up to 255. The character string should contain a keyword from Environmental Queries or the optional word sets to be checked for correspondence with an attribute of the present environment. If the system treats the attribute as unknown, the returned flag is false; otherwise, the flag is true and i*x returned is of the type specified in the table for the attribute queried.

EOL "e-o-l" EXTRA

(--)
Emit spaces to clear the line on the screen beyond the cursor.

ERASE FORTH

(c-addr u --)

If u is greater than zero, clear all bits in each of u consecutive address units of memory beginning at c-addr.

ERR# "error-number" EXTRA

(-- x)

Return the number of the last exception.

ERR$ "error-string" EXTRA

(-- c-addr)

Return the address of the count of the last exception string.

ERRLINE "error-line" EXTRA

(-- a-addr)

a-addr is the address of a cell containing the line number of the file where an exception occurred.

ERRNAME "error-name" EXTRA

(-- a-addr)

a-addr is the address of a cell containing the address of the counted string representing the name of the file where an exception occurred.

ERROR-TYPE EXTRA

(--)

Show the type of the last exception number stored in ERR# by .MESS . Display nothing if ERR# equals -1 or -2.

ERRORLOG ERRORLOG

(-- c-addr)

Contains the name of the logfile for compilation errors.

ESEG EXTRA

(-- x)

x is the value of the DOS environment segment.

EVALUATE FORTH

(i*x c-addr u -- j*x)

Save the current input source specification. Store minus one in SOURCE-ID . Make the string described by c-addr and u both the input source and input buffer, set >IN to zero, and interpret. When the parse area is empty, restore the prior input source specification. Other stack effects are due to the words EVALUATEd.

EVERY EXTRA
(--)
Set a flag so that the next execution of WORDS and such words will act on every vocabulary.

EVERY? "every-query" EXTRA
(-- flag)
flag is true if EVERY was typed in. Subsequent execution without executing EVERY gives a false flag.

EXEC EXTRA
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compiling: (--)
Append the execution semantics below to the current definition.

Executing: (c-addr a-addr --)
Load and execute the file with name specified as a zero terminated string at c-addr and a parameter block at a-addr.

EXEC: "exec-colon" EXTRA
(x --)
Use x as an index into the inline execution array and execute the execution token stored there.

EXECUTE FORTH
(i*x xt -- j*x)
Execute the definition specified by xt. Other stack effects are due to the word EXECUTEd.
See also: ' [']

EXIT FORTH
Interpretation:
Does nothing.

Execution: (--) (R: nest-sys --)
Return control to the calling definition specified by nest-sys.
Before executing EXIT within do-loops, the loop-control

parameters for each loop shall be discarded.
See also: UNLOOP

EXPAND EXTRA
(c-addr1 u1 c-addr2 -- c-addr2 u2)
Copy any non-tab characters in the string specified by c-addr u1 to a string specified by c-addr2 u2. Tab characters are expanded to spaces with a tab distance of 8 positions.

EXPECT OBSOLETE
(c-addr +n --)
Receive a string of at most +n1 characters. Display graphic characters as they are received. A Standard Program that depends on the presence or absence of non-graphic characters in the string has an environmental dependancy. The editing functions, if any, that the system performs in order to construct the string are implementation defined.

Input terminates when "return" is received or when the string is +n characters long. When "return" is received, nothing is appended to the string, and the display is maintained in an implementation defined way.

Store the string at c-addr and its length in SPAN .

Note: this word is obsolescent and is included as a concession to existing implementations. Its function is superseded by ACCEPT .
See also: ACCEPT

EXTEND EXTRA
(--)
Mark all definition so that they can not be forgotten.

EXTRA ONLY
(--)
Make the EXTRA word list the first word list to be searched. This word list contains all CHForth specific extensions to the ANSI standard. Note that these words are non-standard.

FALSE FORTH
(-- false)

Return a false flag.

FAPPEND OUTFILE
(c-addr u --)
Open an existing file and append text to it with FTYPE FCR and FEMIT . This file is on a stack, manipulated by FOPEN / FAPPEND and FCLOSE .

FCHARS OUTFILE
(char u --)
Write a number of chars to a file opened by FOPEN or FAPPEND .

FCLOSE OUTFILE
(--)
Close a file and return to the last one, if any, on the FOPEN or FAPPEND and FCLOSE stack.

FCR OUTFILE
(--)
Write CR to a file opened by FOPEN / FAPPEND .

FEMIT OUTFILE
(c --)
Write a character to a file opened by FOPEN / FAPPEND .

FENCE EXTRA
(-- a-addr)
a-addr is the adress of a cell containing the dictionary pointer since the last SAVE or EXTEND . Forgetting of words created when the dictionary pointer was less than this value is not possible.

FEXT$ "f-ext-string" EXTRA
(-- c-addr)
c-addr is the address of a counted string containing the default extension of Forth text files.

FILE-POSITION FORTH
(fileid -- d ior)
ud is the current file position for the file identified by fileid. ior is the I/O result code.

FILE-SIZE FORTH
(fileid -- ud ior)

ud is the size, in characters, of the file identified by fileid. ior is the I/O result code. This operation does not effect the value returned by FILE-POSITION .

FILE-STATUS FORTH
(c-addr u -- x ior)
Return the status of the file identified by the character string c-addr u. If the file exists, ior is zero; otherwise ior is the I/O result code. x contains the DOS attribute of the file.

FILL FORTH
(c-addr u char --)
If u is greater than zero, store char in each of u consecutive characters of memory beginning at c-addr.

FILLP "fill-p" PARAGRAPHS
(x u char --)
If u is greater than zero, store char in each of u consecutive paragraphs of characters of memory beginning at segment x.

FILLX "fill-x" EXTRA
(x-addr1 u char --)
If u is greater than zero, store char in each of u consecutive characters of memory beginning at extended address x-addr.

FIND FORTH
(c-addr -- c-addr 0 | xt 1 | xt -1)
Find the Forth word named in the counted string at c-addr. If the word is not found after searching all word list in the search order, return c-addr and zero. If the definition is found, return xt. If the definition is immediate, also return 1, otherwise return -1.
See also: ’ [’] POSTPONE

FIND-ATTRIBUTE EXTRA
(-- x)
A value containing the attribute of files to find with FIND-FIRST-FILE . It is reset to zero after execution of FIND-FIRST-FILE .

FIND-FIRST-FILE EXTRA
(c-addr u -- ior)

Find the first file name matching the string specified by c-addr u. Reset the value in FILE-ATTRIBUTE to zero. The name of the file will be returned by FOUND-FILE . If no exception occurs, ior is zero. Otherwise ior is the I/O result code.

FIND-METHODS EXTRA

("name" -- wid)

Skip leading space delimiters. Parse name delimited by a space. Return the word list identification wid of the methods of name. See also: METHODS

FIND-NEXT-FILE EXTRA

(-- ior)

Find the next file name matching the string given to FIND-FIRST-FILE . The name of the file will be returned by FOUND-FILE . If no exception occurs, ior is zero. Otherwise ior is the I/O result code.

FLIP EXTRA

(x1 -- x2)

Exchange the high and low bytes of x1 giving x2.

FLUSH FORTH

(--)

Perform the function of SAVE-BUFFERS and unassign all block buffers.

FLUSH-FILE FORTH

(fileid -- ior)

Attempt to force any buffered information written to the file referred to by fileid to be written to mass storage, and the size information for the file to be recorded in the storage directory if changed. If the operation is successful, ior is zero. Otherwise ior is the I/O result code.

FLYER EXTRA

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (--)

Append the run-time semantics given below to the current definition.

Run-time: (i*x -- j*x)
If STATE contains not zero, continue. Change the dictionary pointer and list dictionary pointer to a temporary area and compile the next words. Reset the dictionary pointers to their prior values and execute the routine just compiled.

FM/MOD "f-m-slash-mod" FORTH
(d n1 -- n2 n3)
Divide d by n1, giving the floored quotient n3 and the remainder n3. Input and output stack arguments are signed. Exception -10 is issued if n1 is zero or the quotient lies outside the range of a double-cell unsigned integer.
See also: SM/REM UM/MOD

FOPEN OUTFILE
(c-addr u --)
Create a file and append text to it with FTYPE FCR and FEMIT . Uses the file stack created with FOPEN / FAPPEND and FCLOSE .

FORGET FORTH
("name" --)
Skip leading space delimiters. Parse name delimited by a space. Find name in the compilation word list, then delete name from the dictionary along with all words added to the dictionary after name. Exception -13 occurs if name cannot be found. Exception -15 occurs if FORGET removes a word required for correct execution.

Note: this word is obsolescent and is included as a concession to existing implementations.

Note: In CHForth words can be protected against FORGET with EXTEND and SAVE .

FORTH FORTH
(--)
Make the FORTH word list the first word list to be searched. Note that this word list contains at startup only ANSI-standard words.

FORTH-WORDLIST ONLY
(-- wid)
Return wid, the identifier of the word list that includes all standard words provided by the implementation. This word list is

initially the compilation word list and is part of the initial search order.

FORWARD ERRORLOG
(c-addr u --)
Compiled when during loading an undefined word is encountered in a colon definition. As an alias of EVALUATE , it will evaluate a string with the name of the unfound word. This can be used to create forward references.

FOUND-ATTRIBUTE EXTRA
(-- char)
Return the file attribute of the last file found.

FOUND-FILE EXTRA
(-- c-addr u)
c-addr u specifies a character string containing the file name found by the last execution of FIND-FIRST-FILE or FIND-NEXT-FILE

FROM EXTRA
Interpretation: ("name" -- x)
Skip leading space delimiters. Parse name delimited by a space. x is the value of name. Exception -32 occurs if name was not defined by VARIABLE .

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VARIABLE .

Run-time: (-- x)
x is the value of name.

FTYPE OUTFILE
(c-addr u --)
Write a string to a file opened by FOPEN / FAPPEND .

FUDGE EXTRA
(-- a-addr)
a-addr is the address of a cell containing a delay to tune MS . The value is set in the file CHFORTH.CFG and can be changed by the user to account for the type of CPU and the clock frequency.

Not available in CHForth-386.

GET EXTRA

("name" --)

Interpretation: ("name" -- wid)

Skip leading space delimiters. Parse name delimited by a space. wid is the word list identification associated with name. Exception -32 occurs if name was not defined by VOCABULARY .

Compilation: ("name" --)

Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VOCABULARY .

Run-time: (-- wid)

wid is the word list identification associated with name.

GET-CONTEXT ONLY

(-- wid)

Return wid, the identifier of the first word list in the search order.

GET-CURRENT ONLY

(-- wid)

Return wid, the identifier of the compilation word list.

GET-DIRECTORY EXTRA

(-- c-addr u ior)

Get the current directory as a character string specified by c-addr u. The path is preceded by the drive letter and a colon. If no exception occurs, ior is zero. Otherwise c-addr and u are unspecified and ior is the I/O result code.

GET-FILE-TIME EXTRA

(fileid -- n1 n2 ior)

Return the time n1 and date n2 of creation of the file identified by fileid. If no exception occurs, ior is zero. Otherwise n1 and n2 are unspecified and ior is the I/O result code.

GET-INTERRUPT EXTRA

(n -- x-addr)

Return the extended address x-addr of the interrupt vector n.

GET-ORDER ONLY

(-- wid1 .. widn n)

Returns the number of word lists n in the search order and the word list identifiers wid1 .. widn identifying these word lists. widn identifies the word list searched first, and wid1 the word list that is searched last. The search order is unaffected.

GETDISK EXTRA

(-- n)

n is the current drive number.

GETMODE EXTRA

(-- n)

n is the number of the current screen mode.

GETNAME EXTRA

("name" -- c-addr u)

Skip leading space delimiters. Parse name delimited by zero and when the length is not zero, store it in a special location and append the extension in FEXT$ to it. Return c-addr u of that string. If the length of name is zero, return the string that was stored in the location by a previous call of GETNAME .

GETTIME EXTRA

(-- d)

d is the number of milliseconds elapsed since midnight.

GLOSS "glossary" FORTH

("fname1" "fname2" --)

Make a glossary with name2 out of the origin file name1 .

H! "h-store" EXTRA

(x h-addr --)

Store x at header address h-addr.

H, "h-comma" EXTRA

(x --)

Reserve one cell of header space and store x in the cell.

H. "h-dot" EXTRA

(u --)
Display u as a four digit hexadecimal number with a trailing space.
See also: .HEX B.

H@ "h-fetch" EXTRA
(h-addr -- x)
Fetch x, x is the value stored at header address h-addr.

HALT EXTRA
(n --)
Terminate the Forth program and return to the operating system with returncode n.

HBYTES "h-bytes" EXTRA
(-- a-addr)
a-addr is the address of a cell containing the header dictionary pointer at the last execution SAVE or EXTEND .

HDUMP "head-dump" EXTRA
(h-addr u --)
Display the contents of u consecutive addresses starting at header address x-addr. At the beginning of the line the extended address is displayed, followed with the hexadecimal contents of 16 characters and then the same characters are displayed with SEMIT .

HDUMP is implemented using pictured numeric output words. Its use will corrupt the transient region identified by #> .
See also: DUMP DUMPX HTYPE

HEAD, "head-comma" EXTRA
(c-addr u --)
Create a dictionary entry named in the character string specified by c-addr u, u may be zero. The name is not known to the Forth system until REVEAL is executed. When WARNING does not contain zero, give a warning when the name is not unique.

HEAD> "head-from" EXTRA
(dea -- xt)
xt is the the execution token that is associated with the dictionary entry address dea.

HEAD>FLAGS "head-to-flags" EXTRA
(dea -- h-addr)
h-addr is the flag field address of the dictionary entry dea.

HEAD>FORGET "head-to-forget" EXTRA
(dea -- h-addr)
h-addr is the forget field address of the dictionary entry dea.

HEAD>NAME "head-to-name EXTRA
(dea -- h-addr)
h-addr is the name field address of the dictionary entry dea.

HEADER EXTRA
("name" | c-addr u --)
If POSTFIX is zero, skip leading space delimiters and parse name delimited by a space; otherwise name is specified by the character string c-addr u. Create a dictionary entry for name. If the data-space pointer is not aligned, reserve enough data space to align it.

HELP HELP
("name" --)
Skip leading space delimiters. Parse name delimited by a space. Look up name in the files with extension given in HEXT$ in the directory given by HELPPATH and display the description of name. As a binary search on the sorted file is performed, only one description per file is displayed. When a full screen is displayed, wait for the user to press any key, escape stops. Otherwise convert name to a number (the prefixes % $ # & etc. are permitted) and display its type and decimal value and the character if it can be displayed or display the exception message if it is defined for the number.

HELPPATH EXTRA
(-- c-addr)
c-addr is the address of a counted string containing the path to the help files.
See also: HELP LIBPATH

HERE FORTH
(-- addr)
addr is the data-space pointer.

HEX FORTH
(--)
Set the contents of BASE to sixteen.

HEXT$ "h-ext-string" EXTRA
(-- c-addr)
c-addr is the address of a counted string containing the default extension of Forth help files.

HHERE "h-here" EXTRA
(-- h-addr)
h-addr is the header-space pointer.

HIDDEN EXTRA
(--)
Mark the most recently created definition as a hidden word.
It can not be found by words like ’ FIND and [’] .

HIDE-CURSOR EXTRA
(--)
Hide the cursor.

HIGHEST EXTRA
(-- wid dea)
Return the dictionary entry address of the newest definition with dictionary entry address dea and the word list identification wid in which it is compiled. Used in FORGET .

HLIMIT EXTRA
(-- h-addr)
Return the address after the last usable in the head segment.

HMEMTOP EXTRA
(-- addr)
Return the address after the last physical address in the header segment.

HOLD FORTH
(char --)
Add char to the beginning of the pictured numeric output string.
An ambiguous condition exists if HOLD executes outside of a <# #> delimited number conversion.

HOME EXTRA

(--)

Set the cursor on the top left of the screen.

HRESERVE EXTRA

(x --)

Reserve x address units above HHERE in the head segment to be used by the compiler in a saved program. When x is zero, all headers of the definitions are discarded in the saved program.

HSEG EXTRA

(-- x)

x is the value of the header segment.

HTAB "h-tab" EXTRA

(u --)

If n is greater than zero, emit spaces until the cursor is at column u of the current user output device.

HTYPE "head-type" EXTRA

(h-addr u --)

If u is greater than zero, display the character string at the header address h-addr for a total of u characters. The characters are displayed as with SEMIT .

See also: HDUMP LTYPE

I FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(-- n|u) (R: loop-sys -- loop-sys)

n|u is a copy of the current (innermost) loop index. An ambiguous condition exists if the loop control parameters are unavailable.

IF FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: -- orig)

Put the location of a new unresolved forward reference orig onto the control flow stack. Append the execution semantics given below to the current definition. The semantics are incomplete until orig is resolved, e.g., by THEN or ELSE .

Execution: (x --)
If all bits of x are zero, continue execution at the location specified by the resolution of orig.
See also: ELSE THEN

IMMEDIATE FORTH
(--)
Mark the most recently created definition as an immediate word.

IN EXTRA
("name" --)
Skip leading space delimiters. Parse name delimited by a space and load the file with that name. If the length of name is zero, load the file that was previously load with IN .

INCLUDE EXTRA
("name" --)
Skip leading delimiters. Parse name delimited by a space and load the file with that name. The appropriate extension must be included in name.

INCLUDE-FILE FORTH
(fileid --)
Remove fileid from the stack. Save the current input source specification, including the current value of SOURCE-ID . Store fileid in SOURCE-ID . Make the file specified by fileid the input source. Store zero in BLK . Other stack effects are due to the words INCLUDEd.

Repeat until end of file: read a line from the file, fill the input buffer from the contents of that line, set >IN to zero, and interpret.

Interpretation begins at the file position where the next file read would occur.

When the end of the file is reached, close the file and restore the input source specification to its saved value.

An ambiguous condition exists if fileid is invalid, if an I/O exception occurs reading fileid, or an I/O exception occurs while closing fileid. When an ambiguous condition exists, the status (open or closed) of any files that were being interpreted is implementation defined.

INCLUDED FORTH

(c-addr u --)

Remove c-addr u from the stack. Save the current input source specification, including the current value of SOURCE-ID . Open the file specified by c-addr u, store the resulting fileid in SOURCE-ID and make it the input source. Store zero in BLK . Other stack effects are due to the words INCLUDEd.

Repeat until end of file: read a line from the file, fill the input buffer from the contents of that line, set >IN to zero, and interpret.

Interpretation begins at the file position where the next file read would occur.

When the end of the file is reached, close the file and restore the input source specification to its saved value.

An ambiguous condition exists if the named file can not be opened, if an I/O exception occurs reading the file, or an I/O exception occurs closing the file. When an ambiguous condition exists, the status (open or closed) of any files that were being interpreted is implementation defined.
See also: INCLUDE-FILE

INCR "increment" EXTRA

(a-addr --)

Add 1 to the single-cell value at a-addr.

IND++ "indent-increment" EXTRA

(--)

Increment the current value of the indentation with eight.

IND-- "indent-decrement" EXTRA

(--)

Decrement the current value of the indentation with eight.

INDENT EXTRA

(-- a-addr)
a-addr is the address of a cell containing the current value of indentation for the decompiler.

INHERIT EXTRA
("name1" "name2" --)
Skip leading space delimiters. Parse name1 delimited by a space.
Skip leading space delimiters. Parse name2 delimited by a space.
Copy the methods list of name1 to the methods list of name2.
Any methods defined for name2 are lost.
See FIND-METHODS METHODS

INLINE# "inline-number" EXTRA
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(-- x)
Return the inline compiled number, system use only.

INLINE$ "inline-string" EXTRA
(-- l-addr)
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

l-addr is the list address of an inline compiled string. System use only.

INTERNAL ONLY
(--)
Make the INTERNAL word list the first word list to be searched. This word list contains CHForth specific extensions to the ANSI standard that are not documented and can be changed by the author by name or action without prior consent. Note that these words are non-standard.

INTERPRET EXTRA
(--)
Interpret the current input stream.

INTVEC "interrupt-vector" INTVEC
(x "name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. Name is referred to as an "interrupt vector".

name Executing: (-- x-addr)
Place x-addr, the extended address of the current vector assigned to interrupt number x. The value of this vector can be changed by executing 'addr TO name', can be reset to its initial value by 'CLEAR name' and the number x can be obtained by executing 'FROM name'. To get the address where the default value is stored, use 'ADR name'.

INVERS EXTRA
(--)
Exchange the character foreground and background colors.

INVERT FORTH
(x1 -- x2)
Invert all bits of x1, giving its logical inverse x2.
See also: 0= NEGATE

IS EXTRA
Interpretation: (xt "name" --)
Skip leading space delimiters. Parse name delimited by a space. Store execution token xt in name. Exception -32 occurs if name was not defined by VECTOR .

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VECTOR .

Run-time: (xt --)
Store execution token xt in name.

IS-FORGET EXTRA
(xt "name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the semantics of execution token xt to the forget method of name.

J FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(-- n|u) (R: loop-sys -- loop-sys)
n|u is a copy of the index of the next outer loop. An ambiguous condition exists if the loop control parameters of the next outer loop are unavailable.

JOIN EXTRA
(char1 char2 -- x)
char1 is the low byte of x and char2 is the high byte of x.

JUMP, "jump-comma" EXTRA
(addr --)
Compile an assembler language jump in the dictionary at the data-space pointer to the address on the stack and increment the data-space pointer to an aligned address after the instruction.

K EXTRA
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(-- n|u) (R: loop-sys -- loop-sys)
n|u is a copy of the index of the second next outer loop. An ambiguous condition exists if the loop control parameters of the second next outer loop are unavailable.

KB. "k-b-dot" EXTRA
(u --)
Display the result of division of u by 1024 with one digit after the decimal point followed by a space, the string "Kb" and a space.

KEY FORTH
(-- char)
Receive one character char, a member of the implementation defined character set. Keyboard events that do not correspond to such characters are discarded until a valid character is received, and those events are subsequently unavailable.

All standard characters can be received. Characters received by

KEY are not displayed.

Standard programs that require the ability to receive control characters have an environmental dependency.
See also: EKEY KEY?

KEY? "key-question" FORTH
(-- flag)
If a character is available, return true. Otherwise return false. If non-8 bit keyboard events are available before the first valid character, they are discarded and subsequently unavailable.

After KEY? returns with a value of true, subsequent executions of KEY? prior to the execution of KEY or EKEY also return true, without discarding keyboard events. The next execution of KEY will return the character without indefinite delay.

L! "l-fetch" EXTRA
(x l-addr --)
Store x at list address l-addr.

L$ ASSEMBLER
(-- addr)
Define a forward near label in assembler, one per definition.

L$: ASSEMBLER
(addr --)
Resolve a forward near label.

L, "l-comma" EXTRA
(x --)
Reserve one cell of list space and store x in the cell.

L/SCR "l-per-s-c-r" EXTRA
(-- n)
Return the number of lines on the screen.

L@ "l-fetch" EXTRA
(l-addr -- x)
Fetch x, x is the value stored at list address l-addr.

LAST EXTRA

(-- a-addr)

a-addr is the address of a double cell containing the last dictionary entry address and its word list identification.

LBYTES "l-bytes" EXTRA

(-- a-addr)

a-addr is the address of a cell containing the list dictionary pointer at the last execution of SAVE or EXTEND .

LC! "l-c-store" EXTRA

(c l-addr --)

Store char at list address l-addr.

LC, "l-c-comma" EXTRA

(char --)

Reserve space for one character in the list space and store char in the space.

LC@ "l-c-fetch" EXTRA

(l-addr -- char)

Fetch the character stored at list address l-addr.

LDUMP "list-dump" EXTRA

(l-addr u --)

Display the contents of u consecutive addresses starting at list address l-addr. At the beginning of the line the extended address is displayed, followed with the hexadecimal contents of 16 characters and then the same characters are displayed with SEMIT .

LDUMP is implemented using pictured numeric output words. Its use will corrupt the transient region identified by #> .
See also: DUMP DUMPX LTYPE

LEAVE FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(--) (R: loop-sys --)

Discard the current loop control parameters. An ambiguous condition exists if they are unavailable. Continue execution

immediately following the innermost syntactically enclosing DO ... LOOP or DO ... +LOOP .
See also: +LOOP LOOP

LHERE "l-here" EXTRA
(-- l-addr)
l-addr is the list-space pointer.

LIBPATH EXTRA
(-- c-addr)
c-addr is the address of a counted string containing the path to the library files.
See also: HELPPATH NEEDS

LIMIT EXTRA
(-- addr)
Return the address after the last usable in the dictionary.

LINE-CURSOR EXTRA
(--)
Set the cursor form to a line.

LINESREAD EXTRA
(-- a-addr)
a-addr is the address of a cell containing the number of file lines read since loading the configuration file at the start of the program.

LIST FORTH
(u --)
Display block u in an implementation-defined format. Store u in SCR .
See also: BLOCK

LITERAL FORTH
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (x --)
Compile x as a literal. Append the run-time syntax given below to the current definition.

Run-time: (-- x)
Place x on the stack.

LITERALS EXTRA

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (x1 .. xn n --)
Append the execution semantics defined below to the current definition.

Executing:
(-- x1 .. xn)
Place x1 to xn on the stack.

LLIMIT EXTRA

(-- l-addr)
Return the address after the last usable in the list segment.

LMEMTOP EXTRA

(-- addr)
Return the address after the last physical address in the list segment.

LOAD FORTH

(i*x u -- j*x)
Save the current input source specification. Store u in BLK , thus making block u the input source and setting the input buffer to encompass its contents, set >IN to zero, and interpret. When the parse area is exhausted, restore the prior input source specification. Other stack effects are due to the words LOADed.

Exceptions -33, -34 or -35 will occur if u is zero, or is not valid block number.

LOADHIGH LOADHIGH

("name" --)
Skip leading space delimiters. Parse name delimited by a space. Allocate temporary space at the top of the dictionary and compile the library name in that space. When this word has been executed, the dictionary space pointers have the same value as before the execution, with the difference that the words in the loaded

library are known to the Forth system.
See also: DISPOSE MARK

LOCAL EXTRA

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution and run-time semantics defined below.

Execution: (x --)
Store x in name.

name Execution: (-- x)
Place x on the stack. The value can be manipulated by TO +TO and CLEAR .

LOCAL-WORDLIST ONLY

(-- wid)
Return the wid of the LOCAL-WORDLIST .

LOCALS| "locals-bar" FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("namen" .. "name2" "name1" "|" --)
Define up to 8 local variables with "name1" to "namen". The list of locals to be defined is terminated with "|". The actual number in CHForth may be greater, depending on the length of the input line. Append the run-time semantics for name given below.

name Run-time: (-- x)
Place x on the stack. The value can be manipulated by TO +TO and CLEAR .

LOGFILE LOG

(-- c-addr)
Contains the name of the logfile.

LOGGING? "logging-query" EXTRA

(-- x)

A value that is true when logging is currently active.

LOOK SEARCHER

("name" "ccc" ---)

Skip leading space delimiters. Parse name delimited by a space. Skip leading SEPARATOR delimiters. Parse ccc delimited by SEPARATOR . Search file name with optional extension given by FEXT$. Find ccc in the file. Display the number of the lines found, the line number and the line containing ccc depending on the width of the screen. If a full screen is displayed, wait for the user to press a key. Stop if the key is the escape key.

LOOP FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: do-sys --)

Append the execution semantics given below to the current definition. Resolve the destination of all unresolved occurrences of LEAVE between the location given by do-sys and the next location for a transfer of control, to execute the words following LOOP.

Execution: (--) (R: loop-sys1 -- | loop-sys2)

Loop control parameters must be available. Add one to the loop index. If the loop index is then equal to the loop limit, discard the loop parameters and continue execution immediately following the loop. Otherwise continue execution at the beginning of the loop.

See also: DO I LEAVE

LRESERVE EXTRA

(x --)

Reserve x address units above LHERE in the list segment to be used by the compiler in a saved program. When x is zero, no compiling is possible in the new program.

LSEG EXTRA

(-- x)

x is the value of the list segment.

LSHIFT "l-shift" FORTH
(x1 u -- x2)
Perform a logical left shift of u bit-places on x1, giving x2. Put zero in the least significant bits vacated by the shift.

LTYPE "list-type" EXTRA
(l-addr u --)
If u is greater than zero, display the character string at the list address l-addr for a total of u characters. The characters are displayed as with SEMIT .
See also: HTYPE LDUMP

M* "m-star" FORTH
(n1 n2 -- d)
d is the signed product of n1 times n2.

M*/ "m-star-slash" FORTH
(d1 n1 +n2 -- d2)
Multiply d1 by n1 producing the triple-cell intermediate result t. Divide t by +n2, giving the double-cell quotient n3. Exception -10 is issued if +n2 is zero or if the quotient lies outside the range of a double-cell signed integer.

Note: The restriction in the Standard to postive values for +n2 is not maintained.
See also: */ */MOD M*/MOD

M*/MOD "m-star-slash-mod" EXTRA
(d1 n1 +n2 -- n3 d2)
Multiply d1 by n1 producing the triple-cell intermediate result t. Divide t by +n2, giving the single-cell remainder n3 and the double-cell quotient n4. Exception -10 is issued if +n2 is zero or the quotient lies outside the range of a double-cell signed integer.

Note: The restriction in the Standard to postive values for +n2 is not maintained.
See also: */ */MOD M*/

M+ "m-plus" FORTH
(d1|ud1 n -- d2|ud2)

Add n to d1|ud1, giving the sum d1|ud2.

MAKE-GLOSS "make-glossary" FORTH

("name" --)

This word reads a source file and builds the glossary information for it in memory.

MANY EXTRA

(--)

Execute the text before on the same line repeatedly until a keypress.

See also: TIMES

MARK LOADHIGH

("name" --)

Skip leading space delimiters. Parse name delimited by a space. Find name. Name is the first word compiled after loading a file with LOADHIGH .

See also: DISPOSE LOADHIGH

MARKER FORTH

("name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a dictionary for name with the execution semantics defined below.

name Executing: (--)

Restore all dictionary allocation and search pointers to the state they had just prior to the definition of name. Remove the definition of name and all subsequent definitions. Restoration of any structures still existing that could refer to deleted definitions or deallocated data space is not necessarily provided. No other contextual information such as numeric base is affected.

MAX FORTH

(n1 n2 -- n3)

n3 is the greater if n1 and n2

MEMTOP EXTRA

(-- addr)

Return the address after the last physical address in memory.

```
MESS"               "mess-quote"                            EXTRA
    ( n "ccc<quote>" -- )
    Parse ccc delimited by a " (double-quote) and compile the string
    in the dictionary. The string is displayed when n is passed to
    .MESS or THROW .

METHODS                                                     EXTRA
    ( "name" -- )
    Skip leading space delimiters. Parse name delimited by a space.
    Start defining methods for name.
    See also: END-METHODS INHERIT

MICROSECONDS                                                EXTRA
    ( d -- )
    Wait for d microseconds, limited to about 1000 by the operating
    system. Not available in CHForth-86.

MIN                                                         FORTH
    ( n1 n2 -- n3 )
    n3 is the lesser if n1 and n2

MINIACCEPT                                                  EDITOR
    ( c-addr u1 -- u2 )
    A mini version of ACCEPT for the kernel.

MOD                 "mod"                                   FORTH
    ( n1 n2 -- n3 )
    Divide n1 by n2, giving the single-cell remainder n3. Exception
    -10 is issued if n1 is zero. If n1 and n2 differ in sign the
    result returned will be the same as returned by the phrase >R S>D
    R> SM/REM DROP . Note that other implementations of the ANSI
    standard may return the result of the phrase >R S>D R> FM/MOD
    DROP .

MONTHS                                                      EXTRA
    ( -- c-addr )
    Array of three letter month names. Months in normal order, but
    letters reversed. To be used in pictured number strings.
    See also: .SHORTDATE

MOVE                                                        FORTH
    ( c-addr1 c-addr2 u -- )
    If u is greater than zero, copy the contents of u consecutive
```

address units at addr1 to the u consecutive address units at addr2. After MOVE completes, the u consecutive address units at addr2 contain exactly what the u consecutive address units at addr1 contained before the move.
See also: CMOVE CMOVE>

MOVEP "move-p" PARAGRAPHS
(x1 x2 u --)
If u is greater than zero, copy the contents of u consecutive paragraph units at segment x1 to the u consecutive paragraph units its at segment x2. After MOVE completes, the u consecutive paragraph units at x2 contain exactly what the u consecutive paragraph units at x1 contained before the move.

MS FORTH
(u --)
Wait u milliseconds.

MS-DOS-IO EXTRA
(--)
Set input and output to slow DOS routines, redirection is supported.
See also: BIOS-IO CONSOLE! CONSOLE@

MU/MOD "m-u-slash-mod" EXTRA
(ud1 u1 -- u2 ud2)
Divide ud1 by u1, giving the quotient ud2 and the remainder u2. All values and arithmetic are unsigned. Exception -10 is issued if u1 is zero or if the quotient lies outside the range of a double-cell unsigned integer.

NEEDS EXTRA
("name" --)
Skip leading space delimiters. Parse name delimited by a space. Find name. If found continue. Otherwise, load the file with the same name (excluding an optional trailing minus sign) from the directory specified in LIBPATH .

NEGATE FORTH
(n1 -- n2)
Negate n1, giving its aritmetic inverse n2.
See also: 0= INVERT

NEW DEBUG

(--)

Enable the use of DEBUG and TRACE .

NEW-GLOSS "new-gloss" FORTH

(--)

This command starts a fresh glossary.

NIP FORTH

(x1 x2 -- x2)

Drop the first item below the top of the stack.

NL "new-line" EXTRA

(--)

Display a new line and emit the number of spaces contained in INDENT .

NO. DECOMPILER

(--)

The decompiler shows only the names of the definitions.
See also: YES.

NOECHO EXTRA

(--)

When loading do not echo lines read to the screen.

NOOP "no-op" EXTRA

(--)

Does nothing.

NORMAL EXTRA

(--)

Set the attribute of the characters on the screen to the default value.

NOSOUND EXTRA

(--)

Turn the speaker off.

NOT-IMPLEMENTED EXTRA

(--)

Abort with exception message: not implemented, used in some definitions.

NUMBER? "number-question" EXTRA

(c-addr u -- 0 | n 1 | d 2)

A word that normally executes (NUMBER?) .

OF FORTH

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: -- of-sys)

Put of-sys on the compilation stack. Append the execution semantics given below to the current definition. The semantics are incomplete uitil resolved by a consumer of of-sys such as ENDOF .

Execution: (x1 x2 -- | x1)

If the two values on the stack are not equal, discard the top value and continue execution at the location specified by the consumer of of-sys (e.g., following the next ENDOF). Otherwise, discard both values and continue execution in line.

See also: CASE ENDCASE ENDOF

OFF EXTRA

(a-addr --)

Clear all bits of the single-cell value at a-addr.

OLD DEBUG

(--)

Disable the use of DEBUG and TRACE .

ON EXTRA

(a-addr --)

Set all bits of the single-cell value at a-addr.

ONLY ONLY

(--)

Set the search order to the minimum search order. The minimum search order includes the ability to interpret the words FORTH-WORDLIST and SET-ORDER .

OPEN-FILE FORTH

(c-addr u x1 -- x2 ior)

Open the file named in the character string specified by c-addr u, with file access indicated by x1.

If the file was successfully opened, ior is zero, x2 is the fileid, and the file has been positioned at the start of the file. Otherwise ior is the I/O result code and x2 is an unspecified value.

OPEN-LOG LOG
(--)
Open the logfile.

OR FORTH
(x1 x2 -- x3)
x3 is the bit-by-bit logical inclusive-or of x1 with x2.

ORDER FORTH
(--)
Display the word lists in the search order in their search order sequence, from the first searched to the last searched. Also display the word list into which new definitions will be placed.

ORDER is implemented using pictured numeric output words. Its use will corrupt the transient region identified by #> .

OUT EXTRA
(-- x)
A value that contains the number of characters printed on the current screen line.

OVER FORTH
(x1 x2 -- x1 x2 x1)
Place a copy of x1 on top of the stack.

P! "p-store" EXTRA
(x1 x2 --)
Write x1 to 16 bit port x2.

P@ "p-fetch" EXTRA
(x1 -- x2)
Read the 16 bit port x1.

PACK EXTRA
(c-addr1 u c-addr2 -- c-addr2)

Place the string specified by c-addr1 u as a counted string at c-addr2.

PAD FORTH

(-- c-addr)

c-addr is the address of a transient region that can be used to hold data for intermediate processing.

PAGE FORTH

(--)

Move to another page for output. Actual function depends on the output device. On a terminal, PAGE clears the screen and resets the cursor position to the upper left corner. On a printer, PAGE performs a form feed.

PARAGRAPH-ALIGNED EXTRA

(addr -- a-addr)

a-addr is the first paragraph-aligned address greater than or equal to addr.

PARAGRAPHS EXTRA

(n1 -- n2)

n2 is the size in address units of n1 paragraphs.

PARSE FORTH

(char "ccc<char>" -- c-addr u)

Parse ccc delimited by the delimiter char.

c-addr is the address (within the input buffer) and u is the length of the parsed string. If the parse area was empty, the resulting string has zero length.

If char is the character for space, control characters are considered equal to char.

PARSE-WORD EXTRA

(char "<chars>ccc<char>" -- c-addr u)

Skip leading char delimiters. Parse ccc delimited by the delimiter char.

c-addr is the address (within the input buffer) and u is the length of the parsed string. If the parse area was empty, the resulting string has zero length.

If char is the character for space, control characters are considered equal to char.

PARSED-WORD EXTRA
(-- c-addr u)
c-addr u specifies the character string that was the last string parsed by PARSE-WORD or WORD . A program may not change the contents of the string.

PAUSE EXTRA
(--)
A word that normally contains NOOP . Used in EKEY only.

PC! "p-c-store" EXTRA
(char x --)
Write char to 8 bit port x.

PC@ "p-c-fetch" EXTRA
(x -- char)
Read the 8 bit port x.

PHANDLE "p-handle" EXTRA
(-- fileid)
A value containing the file identification fileid of the print file; otherwise zero.

PICK FORTH
(xu .. x0 u -- xu .. x0 xu)
Remove u. Copy the xu to the top of the stack. An ambiguous condition exists if there are less than u+2 items on the stack before PICK is executed.

PITCH EXTRA
(n --)
Set the frequency of the speaker to n.

PLACE EXTRA
(c-addr1 u c-addr2 --)
Place the string specified by c-addr1 u as a counted string at c-addr2.

PLUCK EXTRA

(x1 x2 x3 -- x1 x2 x3 x1)
Copy the third stack item to the top of the stack.

POP EXTRA

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VALUE , VARIABLE or VECTOR .

Run-time: (--) (R: x --)
Pop x associated with name from the return stack.

POSTFIX EXTRA

(-- x)
A value that is true when HEADER wants the name on the stack. Normally false as HEADER wants the name in the inputstream.

POSTPONE FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the compilation semantics of name to the current definition. Exception -13 occurs if name is not found.

Execution: (--)
Perform the compilation semantics of name.

PREFIX EXTRA

("name" --)
Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the interpretation and compilation semantics defined below. name is referred to as a "prefix".

Interpretation: (i*x "name1" -- j*x)

Skip leading space delimiters. Parse name1 delimited by a space. Execute the prefix action of name1. Exception -32 occurs if this prefix is not valid for this word or datatype.

Compilation: ("name2" --)
Skip leading space delimiters. Parse name2 delimited by a space. Compile the prefix action of name1. Exception -32 occurs if this prefix is not valid for this word or datatype.

PREVIOUS ONLY
(--)
Transform the search order consisting of wid1 .. widn-1 widn (where widn is searched first) into wid1 .. widn-1. An ambiguous condition exists if the search order was empty before PREVIOUS was executed.

PRINTER EXTRA
(--)
Set the output to the printer.

PRINTING? "printing-query" EXTRA
(-- x)
A value that is true when printer output is enabled.

PRIVATE EXTRA
(--)
Mark the most recently created definition as a private word. This word can not be found after the execution of DEPRIVE .

PRIVATES EXTRA
(--)
Start beginning of a PRIVATES .. DEPRIVE block.

PROJ$ EXTRA
(-- c-addr)
c-addr is the address of a counted string containing a description of the project for which the file is created.

PROJECT PROJECT
("name" --)
Skip leading space delimiters. Parse name delimited by a space. Create a text file for name with the default extension in FEXT$. Write a header as defined in the strings PROJ$ CAT$ and CREAT$

and start the editor with the cursor at a place where the programmer can start typing. This file can be loaded directly after editing by typing IN . After the header is a MARKER for an automatic FORGET when reloading the file.

PROMPT EXTRA

(--)

A word that displays the prompt.

PUSH EXTRA

Interpretation: (i*x --)

This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("name" --)

Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VALUE , VARIABLE or VECTOR .

Run-time: (--) (R: -- x)

Push x associated with name on the return stack.

QUERY FORTH

(--)

Make the user input device the input source. Receive input into the terminal input buffer, replacing any previous contents. Make the result, whose address is returned by TIB , the input buffer. Set >IN to zero.

Note: this word is obsolescent and is included as a concession to existing implementations.

QUIT FORTH

(--)

Empty the return stack, store zero in SOURCE-ID , make the user input device the input source, and enter interpretation state. Do not display a message. Repeat the following:

- Accept a line forth the input source into the input buffer, set >IN to zero and interpret.
- Display the implementation defined input prompt if in interpretation state, all processing has been completed, and no ambiguous condition exists.

R" "r-quote" EXTRA
(-- x1) (R: x1 x2 x3 -- x1 x2 x3)
Copy x1 from the return stack to the data stack.

R' "r-tick" EXTRA
(-- x1) (R: x1 x2 -- x1 x2)
Copy x1 from the return stack to the data stack.

R/O "r-o" FORTH
(-- x)
x is the value for selecting the "read-only" file access method.
See also: CREATE-FILE OPEN-FILE

R/W "r-w" FORTH
(-- x)
x is the value for selecting the "read-write" file access method.
See also: CREATE-FILE OPEN-FILE

R> "r-from" FORTH
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(-- x) (R: x --)
Move x from the return stack to the data stack.
See also: >R 2>R 2R> 2R@ R@

R>DROP "r-from-drop" EXTRA
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(--) (R: x --)
Remove x from the return stack.

R@ "r-fetch" FORTH
(-- x) (R: x -- x)
Copy x from the return stack to the data stack.
See also: >R 2>R 2R> 2R@ R>

RANDOM EXTRA
(-- u)

Return a random number.

RANDOMIZE EXTRA

(d --)

Initialize the random number generator.

READ-FILE FORTH

(c-addr u1 fileid -- u2 ior)

Read u1 consecutive characters to c-addr from the current position of the file identified by fileid.

If u1 characters are read without an exception, ior is zero and u2 is equal to u1.

If the end of the file is reached before u1 characters are read, ior is zero and u2 is the number of characters actually read.

If the operation is initiated when the value of FILE-POSITION is equal to the value returned by FILE-SIZE for the file identified by fileid, ior is zero and u2 is zero.

If an exception occurs, ior is the I/O result code and u2 is the number of characters transferred to c-addr without an exception.

An ambiguous condition exists if the operation is initiated when the value returned by FILE-POSITION is greater than the value returned by FILE-SIZE for the file identified by fileid, or if the requested operation attempts to read portions of the file not written.

At the conclusion of the operation FILE-POSITION returns a value past the characters consumed by the operation.

READ-LINE FORTH

(c-addr u1 fileid -- u2 flag ior)

Read the next line from the file specified by fileid into memory at the address c-addr. At most u1 characters are read. Up to two line terminating characters may be read into memory at the end of the line, but are not included in the count u2. The line buffer provided by c-addr should be at least u1+2 characters long.

If the operation succeeded, flag is true and ior is zero. If a line terminator was received before u1 characters were read, then u2 is the number of characters, not including the line terminator, actually read (0 <= u2 <= u1). When u1 = u2 the line terminator has yet to be reached.

If the operation is initiated when the value returned by FILE-POSITION is equal to the value returned by FILE-SIZE for the file identified by fileid, flag is false, ior is zero, and u2 is zero. If ior is non-zero, an exception occurred during the operation and ior is the I/O result code.

An ambiguous condition exists if the operation is initiated when the value returned by FILE-POSITION is greater than the value returned by FILE-SIZE for the file identified by fileid, or the requested operation attempts to read portions of the file not yet written.

At the conclusion of the operation, FILE-POSITION returns a value past the characters consumed by the operation.

READX-FILE "read-x-file" EXTRA

(x-addr u1 fileid -- u2 ior)
Read u1 consecutive characters to extended address x-addr from the file specified by fileid. If no exception occurs, u2 is the number of characters read and ior is zero. Otherwise u2 is unspecified and ior is the I/O result code.

REALLOC EXTRA

(u1 u2 -- ior)
Change the allocation of the contiguous region of memory outside the data space starting at the segment address u1, previously allocated by ALLOC or REALLOC , to u2 paragraphs. u2 may be either larger or smaller than the current size of the region. The starting segment address u1 is not changed. If no exception occurs, ior is zero. Otherwise ior is the I/O result code.

RECOVER-SCREEN SCREENSV

(--)
Restore the former contents of the screen and the position of the cursor from memory that is saved there by SAVE-SCREEN . Do not discard this data so this word can be executed more times to restore the same screen. See also RESTORE-SCREEN .

RECURSE FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (--)
Append the execution semantics of the current definition to the current definition. The same description is valid if RECURSE is used in a definition after DOES> .
See also: DOES>

RECURSIVE EXTRA
(--)
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (--)
Makes the current definition available to the system. Normally this happens automatically when executing ; . When the current word is available to the system a reference to its name produces a recursive call to the definition. If RECURSIVE is not executed a reference to that name will result in calling a previous definition with the same name, if one exists.

REF REF
("name" --)
Skip leading space delimiters. Parse name delimited by a space. Find compiled references in colon definitions of name in all word lists. Display the words where the references occur and the count of the words where the references are found.

REFILL FORTH
(-- flag)
Attempt to fill the current input stream, returning a true flag if successful. The action depends on the source of the current input stream.
If the input-stream source is a string from EVALUATE , REFILL returns false and performs no other action.
Otherwise, REFILL attempts to receive input into the text-input buffer whose address is given by TIB , making the result the current input stream and returning a true flag if successful. Receipt of a line containing no characters is

considered successful. A false flag is returned only when there is no input available from the current input-stream source.
If the input source is a block, REFILL makes the next block the current input source and input buffer, by adding one to the value of BLK and setting >IN to zero. True is returned if the new value of BLK is a valid block number, false otherwise.
If the input-stream source is a text file, REFILL attempts to read the next line from the text-input file, making the result the current input stream and returning true if the read succeeded, and returning false otherwise.

REGULAR? "regular-query" EXTRA
(wid -- wid flag)
If the word list identification wid has a header (when it was created with VOCABULARY), return a true flag else a false flag.

RENAME-FILE FORTH
(c-addr1 u1 c-addr2 u2 -- ior)
Rename the file named by character string c-addr1 u1 to the name in the character string c-addr2 u2. ior is the I/O result code.

REPEAT FORTH
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: orig dest --)
Append the execution semantics given below to the current definition, resolving the backward reference dest. Resolve the forward reference orig using the location following the appended execution semantics.

Execution: (--)
Continue execution at the location given by dest.
See also: BEGIN WHILE

REPOSITION-FILE FORTH
(ud fileid -- ior)
Reposition the file identified by fileid to ud. ior is the I/O result code. An ambiguous condition exists if the file is positioned outside the file boundaries.

RESERVE EXTRA

(x --)

Reserve x address units above HERE to be used by ALLOT in a saved program. Some space is always available in PAD and TEMPORARY so interpreting remains possible if x is zero.

RESIZE-FILE FORTH

(ud fileid -- ior)

Set the size of the file identified by fileid to ud. ior is the I/O result code.

If the resultant file is larger than the file before the operation, the portion of the file added as a result of the operation may not have been written.

At the conclusion of the operation FILE-SIZE returns the value ud and FILE-POSITION returns an unspecified value.
See also: READ-FILE READ-LINE

RESTART? EXTRA

(-- x)

A value that prohibits restarting of the initialization of a program. When the program is started its value is false. When Ctrl-Break is pressed, it is set to true.

RESTORE-INPUT FORTH

(x1 .. xn n -- flag)

Attempt to restore the input source specification to the state described by x1 through xn, flag is true if the input source specification can not be so restored.

An ambiguous condition exists if the input source represented by the arguments is not the same as the current input source.
See also: SAVE-INPUT

RESTORE-METRICS EXTRA

(--)

When returning from a system call, reset some screen parameters.

RESTORE-SCREEN SCREENSV

(--)

Restore the former contents of the screen and the position of the cursor from memory that is saved there by SAVE-SCREEN and delete the saved data. See also SAVE-SCREEN and RECOVER-SCREEN

RETCODE "return-code" EXTRA

(-- a-addr)

a-addr is the address of a cell used to count exceptions when the file ERRORLOG is loaded. RETCODE @ HALT gives a return code that can be handled in DOS with ERRORLEVEL.

REVEAL EXTRA

(--)

Make the last made dictionary entry known to the Forth system.

ROLL FORTH

(xu xu-1 .. x0 u -- xu-1 .. x0 xu)

Remove u. Rotate u+1 items on the top of the stack. An ambiguous condition exists if there are less than u+2 items on the stack before ROLL is executed.

ROT "rote" FORTH

(x1 x2 x3 -- x2 x3 x1)

Rotate the top three stack items.

RSHIFT "r-shift" FORTH

(x1 u -- x2)

Perform a logical right shift of u bit-places on x1, giving x2. Put zero in the most significant bits vacated by the shift.

S STACK

(-- x)

(S: x -- x)

Copy the top number on the auxiliary stack to the data stack.

S" "s-quote" FORTH

Interpretation: ("ccc<quote>" -- c-addr u)

Parse ccc delimited by " (double quote). Store the resulting string ccc at a temporary location. The maximum length of the temporary buffer is 255 characters. CHForth allows for the storing of more such strings before new strings start to overwrite the buffer. A standard program shall not alter the returned string.

Compilation: ("ccc<quote>" --)

Parse ccc delimited by " (double quote). Append the run-time semantics given below to the current definition.

Run-time: (-- c-addr u)
Return c-addr and u describing a string consisting of the characters ccc. A standard program shall not alter the returned string.
See also: C"

S> "s-from" STACK
(-- x)
(S: x --)
Pop a number from the auxiliary stack.

S>D "s-to-d" FORTH
(n -- d)
Convert the number n to the double-cell number d with the same numerical value.

S>DROP "s-drop" STACK
(--)
(S: x --)
Drop the top number of the auxiliary stack.

SAVE EXTRA
("name" --)
Skip leading space delimiters. Parse name delimited by a space. Protect the dictionary as with EXTEND . Write the CHForth program as an executable file with this name. name may have a preceding path but no extension. The current settings of LIMIT and MEMTOP are preserved as are their equivalents in other segments.

SAVE-BUFFERS FORTH
(--)
Transfer the contents of each UPDATEd block buffer to mass storage. Mark all buffers as unmodified.

SAVE-INPUT FORTH
(-- x1 .. xn n)
x1 through xn describe the current state of the input source specification for later use by RESTORE-INPUT .

SAVE-SCREEN SCREENSV

(--)
Keep the contents of the screen and the cursor position in memory. There is room for 4 screens. See also RESTORE-SCREEN and RECOVER-SCREEN .

SBASE "s-base" EXTRA
(-- x)
x is the segment number of the text screen.

SCAN EXTRA
(c-addr1 u1 char -- c-addr2 u2)
Scan the string specified by c-addr1 u1 for an occurrence of char and return the part of the string starting with the found char as a string specified by c-addr2 u2. If the string specified by c-addr1 u1 does not contain char, u2 is zero.

If char is the character for space, control characters are considered equal to char.

SCAN-ANY EXTRA
(-- xt)
Skip leading space delimiters. Parse name delimited by a space. Find name. If found return the execution token xt of that word. Otherwise refill the input buffer with REFILL and repeat. Exception -58 will occur if refilling the input buffer fails.

SCR "s-c-r" FORTH
(-- a-addr)
a-addr is the address of a cell containing the block number of the block most recently LISTed.

SCREENSIZE EXTRA
(-- n)
n is the total count of characters plus attributes on the screen.

SEARCH FORTH
(c-addr1 u1 c-addr2 u2 -- c-addr3 u3 flag)
Search the string specified by c-addr1 u1 for the string specified by c-addr2 u2. If flag is true, a match was found at c-addr3 with u3 characters remaining. If flag is false there was no match and c-addr3 is c-addr1 and u3 is u1.

SEARCH-CONTEXT EXTRA

(c-addr u -- 0 | xt 1 | xt -1)

Find the Forth word specified by the character string c-addr u in all word lists in the search order, including LOCAL-WORDLIST when STATE does not contain zero and there are local values. Return the execution token and 1 if the word is IMMEDIATE and -1 otherwise. If name can not be found, return a false flag. The name is internally converted to uppercase if the variable CASESENSITIVE is false.

SEARCH-ENVIRONMENT EXTRA

(c-addr1 u1 -- c-addr2 u2)

Search the DOS environment strings for the string specified by c-addr1 u1. Return the character string after the first string as a character string specified by c-addr2 u2. If the string is not found, u2 is zero and c-addr2 is unspecified.

SEARCH-WORDLIST FORTH

(c-addr u wid -- 0 | xt 1 | xt -1)

Find the Forth word identified by the string c-addr u in the word list identified by wid. If the word is not found, return zero. If the word is found, return its execution token xt and 1 if the word is immediate, -1 otherwise.

SEE DECOMPILER

("name" --)

Skip leading space delimiters. Parse name delimited by a space. Find name. If name can not be found exception -13 occurs. If name is high level, decompile it. Otherwise if the disassembler is loaded, disassemble it.

SEGMENT EXTRA

(x "name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. Leave the dictionary pointer at an aligned address. Allocate space for 3 cells. Ask DOS for an allocation of x paragraphs and store the segment number of that allocation in the first cell. Store x in the second cell and zero in the third. The user may change the value of the third cell to a value less than or equal to x in order to save the allocated area with the program.

name Execution: (-- a-addr)

a-addr is the address of the first reserved cell of name.

SEMIT "s-emit" EXTRA
(char --)
If char is a printable ASCII character in the range {32 .. 127}, use EMIT to display char. Otherwise use EMIT to display a ’.’ (full stop).
See also: EMIT

SEPARATOR EXTRA
(-- char)
A constant character that can be used as a line separator for some commands, like SF DIR etc. Normally 254, ’’.

SET-CONTEXT ONLY
(wid --)
Set the first searched word list in the search order to the word list identified by wid.

SET-CURRENT ONLY
(wid --)
Set the compilation word list to the word list identified by wid.

SET-DIRECTORY EXTRA
(c-addr u -- ior)
Set the current directory to the string specified by c-addr u. As an extension to DOS, the default drive can also be changed if a drive letter and a colon are present at the beginning of the string. If no exception occurs, ior is zero. Otherwise ior is the I/O result code.

SET-FILE-TIME EXTRA
(n1 n2 fileid -- ior)
Set the time n1 and date n2 of creation of the file identified by fileid. If no exception occurs, ior is zero. Otherwise ior is the I/O result code.

SET-INTERRUPT EXTRA
(x-addr n --)
Set interrupt vector n to extended address x-addr.

SET-ORDER ONLY

(wid1 .. widn n --)

Set the search order to the word lists wid1 .. widn. Subsequently, word list widn will be searched first, followed by word list widn-1 and so on, with word list wid1 searched last. If n is zero, empty the search order. If n is minus one, set the search order to the minimum search order wid(ONLY) wid(ONLY). When n is minus two, set the search order to wid(ONLY) wid(EXTRA) wid(FORTH) wid(FORTH). The maximum of n in this implementation is sixteen.

SET-SOURCE EXTRA

(c-addr u --)

Set the source to the string c-addr u and set >IN to zero.

SETDISK EXTRA

(n1 -- n2)

Set the current drive to n1. n2 is the the total number of available drives.

SETMODE EXTRA

(n --)

Set the screen to mode n.

SF "search-forth" SEARCHER

("ccc" --)

Skip leading SEPARATOR delimiters. Parse ccc delimited by SEPARATOR . Search the files with extension given by FEXT$ in the current directory. Find ccc in the files. Display the number of lines found, the name of the file, the line number and the line depending on the width of the screen. If a full screen is displayed, wait for the user to press a key. Stop if the key is the escape key.

SHOW EDITOR

("name" --)

Skip leading space delimiters. Parse name delimited by a space. Open file name with list program. When name is omitted, the last opened file by this command or ,EDIT EDIT or WHAT is opened and name is displayed on the right of the status line. The default extension is taken from FEXT$.

SHOW-CURSOR EXTRA

(--)
Display the cursor.

SHOW-ERROR EXTRA
(n --)
Display the exception message and information where the exception with number n occurred and the type of the exception and display the source line with the exception word marked out.

SHOWHELP EDITOR
("name" --)
Skip leading space delimiters. Parse name delimited by a space. Open file name in the directory given in HELPPATH with the list program. The default extension is taken from HEXT$.

SHOWLIB EDITOR
("name" --)
Skip leading space delimiters. Parse name delimited by a space. Open file name in the directory given in LIBPATH with the list program. The default extension is taken from FEXT$.

SIGN FORTH
(n --)
If n is negative, add a minus sign to the beginning of the pictured numeric output string. An ambiguous condition exists if SIGN executes outside of a <# #> delimited number conversion.

SIGNON EXTRA
(-- a-addr)
a-addr is the address of a cell containing true to display the signon message at startup and false otherwise.

SILENT EXTRA
(--)
Suppress output to screen or printer.

SKIP EXTRA
(c-addr1 u1 char -- c-addr2 u2)
Skip leading occurrences of char in the string specified by c-addr1 u1 and return the remaining string specified by c-addr2 u2. If the string specified by c-addr1 u1 contains only occurrences of char, u2 is zero.

If char is the character for space, control characters are considered equal to char.

SL "search-libraries" SEARCHER
("ccc" --)
Skip leading SEPARATOR delimiters. Parse ccc delimited by SEPARATOR . Search the files with extension given by FEXT$ in the directory given by LIBPATH . Find ccc in the files. Display the number of lines found, the name of the file, the line number and the line depending on the width of the screen. If a full screen is displayed, wait for the user to press a key. Stop if the key is the escape key.

SLITERAL FORTH
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (c-addr1 u --)
Append the run-time semantics given below to the current definition.

Run-time: (-- c-addr2 u)
Return c-addr2 u describing a string consisting of the characters specified by c-addr1 u during compilation. A Standard Program shall not alter the returned string.

SM/REM "s-m-slash-rem" FORTH
(d n1 -- n2 n3)
Divide d by n1, giving the symmetric quotient n3 and the remainder n3. Input and output stack arguments are signed. Exception -10 is issued if n1 is zero or the quotient lies outside the range of a double-cell unsigned integer.
See also: FM/MOD UM/MOD

SOUND EXTRA
(--)
Turn the speaker on.

SOURCE FORTH
(-- c-addr u)
c-addr is the address of, and u is the number of characters in, the input buffer.

SOURCE-ID FORTH

(-- x)

Identifies the source of the non-block input stream (i.e., when BLK is zero) as follows:

SOURCE-ID	Input stream source
0	Keyboard
-1	String (via EVALUATE)
fileid	Text file "fileid"

An ambiguous condition exists if SOURCE-ID is used when BLK contains a non-zero value.

SPACE FORTH

(--)

Display one space.

SPACES FORTH

(n --)

If n is greater than zero, display n spaces.

SPAN OBSOLETE

(-- a-addr)

a-addr is the address of a cell containing the count of characters stored by the last execution of EXPECT .

Note: this word is obsolescent and is included as a concession to existing implementations.

SPLIT EXTRA

(x -- char1 char2)

char1 is the low byte of x and char2 is the high byte of x.

SRCSEG "source-segment" EXTRA

(-- a-addr)

a-addr is the address of a cell containing the segment address of the first string in COMPARE and SEARCH . The user is reponsible to restore the default value (CSEG) after using an alternative value in COMPARE and SEARCH .

START EXTRA

(--)

A word that is executed at the start of the program before executing COLD .

STATE

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

(-- a-addr)
a-addr is the address of a cell containing the compilation state flag. STATE is true when in compilation state, false otherwise. The true value in STATE is non-zero, but is otherwise implementation-defined. Only the following standard words alter the value in STATE : : (colon), ; (semicolon), ABORT , QUIT , :NONAME , [(left-bracket),] (right-bracket) and ;CODE .

Note: A Standard Program may not directly alter the contents of STATE .
See also: : :NONAME ; ABORT QUIT [ ]

STATOFF EXTRA

(--)
Disable the display of the statusline.

STATON EXTRA

(--)
Enable the display of the statusline.

STATUS? "status-query" EXTRA

(-- x)
A value that is true when the statusline is enabled.

STATUSATTR "status-attribute" EXTRA

(-- a-addr)
a-addr is the address of a cell containing the attribute of the characters on the status line.

STOP? "stop-question" EXTRA

(-- flag)
Return false is no key is pressed. Exception -28 occurs when the escape key was pressed. If the key was not space, return true. Wait for a second keypress and return true if it was not space, false otherwise. Exception -28 occurs when the escape key

was pressed.

STRINGS? VIEW

(-- x)

When this value is true, inline strings are displayed as with DUMP using VIEW .

STYPE "s-type" EXTRA

(c-addr u --)

If u is greater than zero, display the character string specified by c-addr and u. The characters are displayed as with SEMIT .

STYPEX "s-type-x" EXTRA

(x-addr u --)

If u is greater than zero, display the character string at the extended address x-addr for a total of u characters. The characters are displayed as with SEMIT .

SWAP FORTH

(x1 x2 -- x2 x1)

Exchange the top two stack items.

SYSTEM EXTRA

(c-addr u --)

Execute the DOS command specified by the character string c-addr u. When the screen mode or the current direcotory are changed, they will be restored.

T STACK

(-- x1)

(S: x1 x2 -- x1 x2)

Copy the second number on the auxiliary stack to the data stack.

TEMPORARY EXTRA

(-- c-addr)

c-addr is the address of a transient region that is used to hold data for intermediate processing. This region is used by some system words.

TERMINAL EXTRA

(--)

Reset the input and output to the terminal.

TEXT EXTRA
(--)
Reset the display to the same textmode as at startup.
See also: TEXT0 TEXT?

TEXT0 "text-zero" EXTRA
(--)
Set the display to 80 x 25 color text mode.
See also: TEXT TEXT0 TEXT1 TEXT2 TEXT?

TEXT1 "text-one" EXTRA
(--)
Set the display to 132 x 25 text mode. Only available with Speedstar Pro ?
See also: TEXT0 TEXT2

TEXT2 "text-two" EXTRA
(--)
Set the display to 132 x 43 text mode. Only available with Speedstar Pro ?
See also: TEXT0 TEXT1

TEXT? "text-query" EXTRA
(-- x)
A value that is true when the display is in textmode.

THEFILE EXTRA
(-- c-addr)
c-addr is the address of a counted string containing the name of the current file.

THEN FORTH
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: orig --)
Resolve the forward reference orig using the location of the execution semantics.

Execution: (--)
Continue execution.

See also: ELSE IF

THROW FORTH

(k*x n -- k*x | i*x n)

If any bits of n are non-zero, pop the topmost exception frame from the exception stack, along with everything on the return stack above that frame. Then restore the input source specification in use before the corresponding CATCH and adjust the depths of all three stacks so that they are the same as the depth saved in the exception frame (i is the same number as i in the input arguments to the corresponding CATCH), put n on top of the data stack, and transfer control to a point just after the CATCH that pushed that exception frame.

THRU FORTH

(i*x u1 u2 -- j*x)

LOAD the mass storage blocks numbered u1 through u2 in sequence. Other stack effects are due to the words LOADed.

TIB "t-i-b" FORTH

(-- c-addr)

c-addr is the address of the terminal input buffer.

TILL DECOMPILER

("name" --)

Skip leading space delimiters. Parse name delimited by a space. Find name. If name can not be found exception -13 occurs. Otherwise decompile all the words in the current word list starting with the last compiled until name is decompiled.

See also: STOP?

TIME EXTRA

(-- +n1 +n2 +n3)

Return the current time. +n1 is the second {0..59}, +n2 is the minute {0..59}, and +n3 is the hour {0..23}.

TIME&DATE FORTH

(-- +n1 +n2 +n3 +n4 +n5 +n6)

Return the current time and date. +n1 is the second {0..59}, +n2 is the minute {0..59}, +n3 is the hour {0..23}, +n4 is the day {1..31}, +n5 is the month {1..12}, and +n6 is the year (e.g. 1991).

TIMER-RESET EXTRA

(--)

Reset the Forth timer.

TIMES EXTRA

(n --)

Execute the text before on the same line repeatedly for n times.

See also: MANY

TIMESAVE EXTRA

(-- a-addr)

a-addr the the address of a double cell used by TIMER-RESET to store the current value of the timer.

TO FORTH

Interpretation: (x "name" --)

Skip leading space delimiters. Parse name delimited by a space. Store x in name. Exception -32 occurs if name was not defined by VALUE or VARIABLE .

Compilation: ("name" --)

Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -32 occurs if name was not defined by VALUE , VARIABLE or (LOCAL).

Run-time: (x --)

Store x in name.

See also: (LOCAL) VALUE

TONE EXTRA

(n1 n2 --)

Make a sound for the duration of n1 milliseconds with a frequency of n2.

TRACE TRACER

(-- a-addr)

A variable used in the tracer. When not zero, trace information is compiled in the next compiled colon definition. See DEBUG .

TRAP EXTRA

(--)

Jump back the debugger program, use it when you want to step through Forth.

TRUE FORTH

(-- true)

Return a true flag, a single-cell value with all bits set.

TUCK FORTH

(x1 x2 -- x2 x1 x2)

Copy the first (top) stack item below the second stack item.

TURNKEY EXTRA

("name1" "name2" --)

Skip leading space delimiters. Parse name1 delimited by a space. Skip leading space delimiters. Parse name2 delimited by a space. Protect the dictionary as with EXTEND . Write the CHForth program as an executable file with this name2. name2 may have a preceding path but no extension.

The saved file does not contain any headers, so interpreting in the executible file is not possible. The data space and list space will also be reduced to the minimum value that is needed to containt the current data in the data and list space. Both spaces can be enlarged before executing this word.

When this program is executed from the DOS prompt, name1 will be executed by CATCH and at the end the control will be returned to DOS. The program saved has no capability to compile and has no headers.

TYPE FORTH

(c-addr u --)

If u is greater than zero, display the character string specified by c-addr and u.

See also: EMIT

TYPEP "type-paragraphs" PARAGRAPHS

(x u --)

If u is greater than zero, display the character string at paragraph address x for a total of u paragraphs. The characters are displayed as with SEMIT .

TYPEX "type-x" EXTRA

(x-addr u --)
If u is greater than zero, display the character string at the extended address x-addr for a total of u characters.

TYPEZ "type-z" EXTRA
(x-addr --)
While the character at the extended address x-addr is not zero, display the character and increment x-addr.

U STACK
(-- x1)
(S: x1 x2 x3 -- x1 x2 x3)
Copy the third number on the auxiliary stack to the data stack.

U. "u-dot" FORTH
(u --)
Display u in free field format.

U.R "u-dot-r" FORTH
(u n --)
Display u right aligned in a field n characters wide. If the number of characters required to display u is greater than n, all digits are displayed with no leading spaces in a field as wide as necessary.

U2/ "u-two-slash" EXTRA
(x1 -- x2)
x2 is the result by shifting x1 one bit toward the least-significant bit, filling the vacated most-significant bit with zero.

U< "u-less-than" FORTH
(u1 u2 -- flag)
flag is true if and only if u1 is less than u2.
See also: <

U> "u-greater-than" FORTH
(u1 u2 -- flag)
flag is true if and only if u1 is greater than u2.
See also: >

U>D "u-to-d" EXTRA

(u -- ud)
ud is the equivalent of u.

UD. "u-d-dot" EXTRA
(ud --)
Display ud in free field format.

UD.R "u-d-dot-r" EXTRA
(ud n --)
Display ud right aligned in a field n characters wide. If the number of characters required to display ud is greater than n, all digits are displayed with no leading spaces in a field as wide as necessary.

UM* "u-m-star" FORTH
(u1 u2 -- ud)
Multiply u1 by u2 giving the unsigned double-cell product ud. All values and arithmetic are unsigned.

UM/MOD "u-m-slash-mod" FORTH
(ud u1 -- u2 u3)
Divide ud by u1, giving the quotient u3 and the remainder u2. All values and arithmetic are unsigned. Exception -10 is issued if u1 is zero or if the quotient lies outside the range of a single-cell unsigned integer.
See also: FM/MOD SM/REM

UMAX "u-max" EXTRA
(u1 u2 -- u3)
u3 is the greater if u1 and u2

UMIN "u-min" EXTRA
(u1 u2 -- u3)
u3 is the lesser if u1 and u2

UNLOOP FORTH
Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Excecution: (--) (R: loop-sys)
Discard the loop-control parameters for the current nesting level. An UNLOOP is required for each nesting level before the

definition may be EXITed. An ambiguous condition exists if the loop-control parameters are not available.

UNTIL FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: dest --)
Append the execution semantics given below to the current definition, resolving the backward reference dest.

Execution: (x --)
If all bits of x are zero, continue execution at the location specified by dest.
See also: BEGIN

UNUSED FORTH

(-- u)
u is the amount of space remaining in the region addressed by HERE , in address units.

UPDATE FORTH

(--)
Mark the current block buffer as modified.

UPDATE does not immediate cause I/O.
See also: BLOCK BUFFER FLUSH SAVE-BUFFERS

UPPER EXTRA

(c-addr u --)
Convert the lowercase characters in the string specified by c-addr u to uppercase.

VALUE FORTH

(x "name" --)
Skip leading space delimiters. Parse name delimited by a space. Create a defimition for name with the execution semantics defined below with an initial value equal to x. name is referred to as a "value".

name Execution: (-- x)
Place x on the stack. The value of x is that given when name was

is created, until the phrase x TO name is executed, causing a new value of x to be associated with name.
See also +TO ADR CLEAR POP PUSH

VARIABLE FORTH
("name" --)
Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. Reserve one cell of data space at an aligned address. name is referred to as a "variable."

name Execution: (-- a-addr)
a-addr is the address of the reserved cell. A program is responsible for initializing the contents of the reserved cell.

VECTOR EXTRA
("name" --)
Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. name is referred to as a "vector".

name Execution: (i*x -- j*x)
Execute the execution token stored in the entry. The execution token can be manipulated by IS . Exception -525 occurs if no execution token is assigned to name.
See also CHAIN POP PUSH

VERSION EXTRA
(-- n)
n is the three decimal digit version number of this CHForth system.

VID+PRN EXTRA
(--)
The output will go to both the screen and the printer.

VIDEO EXTRA
(--)
Set the output to the screen.

VIEW VIEW
("name" --)
Find "name" in the search-order or convert it to an address.

Display one line at the time of data with, space continues, other keys terminate.

VOC! FORTH

(dea wid --)

Store the dictionary entry address dea in the word list described by the word list identifier wid.

VOC-LINK EXTRA

(-- x)

A value that links all word lists and vocabularies.

VOC@ EXTRA

(wid -- dea)

Fetch the dictionary entry address dea of the last definition from the word list described by the word list identifier wid.

VOCABULARY EXTRA

("name" --)

Skip leading space delimiters. Parse name delimited by a space. Create a definition for name with the execution semantics defined below. Create a new word list and store the word list identifier with the definition for name. name is referred to as a "vocabulary".

name Execution: (--)

Make the above created word list the current word list.

W/O "w-o" FORTH

(-- x)

x is the value for selecting the "write-only" file access method. See also: CREATE-FILE OPEN-FILE

WARNING EXTRA

(-- a-addr)

a-addr is the address of a cell containing true when the program will warn the user when redefinitions are encountered and false otherwise.

WHAT EDITOR

(--)

Open file name with the editor program and place the cursor at the line number stored in ERRLINE . name is stored at the address stored in ERRNAME . ERRNAME and ERRLINE are valid after an exception that occured during loading of file name. name is displayed on the right of the status line.

WHILE FORTH

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: (C: dest -- orig dest)
Put the location of a new unresolved forward reference orig onto the control flow stack, under the existing dest. Append the execution semantics given below to the current definition. The semantics are incomplete until orig and dest are resolved (e.g., by REPEAT).

Execution: (x --)
If all bits of x are zero, continue execution at the location specified by the resolution of orig.

WITH EXTRA

("name" --)
Skip leading space delimiters. Parse name delimited by a space. Display the words from every vocabulary containing name. Case is significant.

WITHIN FORTH

(n1|u1 n2|u2 n3|u3 -- flag)
Perform a comparison of a test value n1|u1 with a lower limit n2|u2 and an upper limit n3|u3, returning true if either (n2|u2<n3|u3 and (n2|u2<=n1|u1 and n1|u1<n3|u3)) or (n2|u2>n3|u3 and (n2|u2<=n1|u1 or n1|u1<n3|u3)) are true, returning false otherwise. An ambiguous condition exists if n1|u1, n2|u2, and n3|u3 are not all the same type.

WORD FORTH

(char "<chars>ccc<char>" -- c-addr)
Skip leading delimiters. Parse characters ccc delimited by char. An ambiguous condition exists if the length of the parsed string is greater then the implementation defined length of a counted string.

c-addr is the address of a transient region containing the parsed word as a counted string. If the parse area was empty or contained no characters other than the delimiter, the resulting string has zero length. A space, not included in the length, follows the string. A Standard Program may replace characters within the string.

If char is the character for space, control characters are considered equal to char.

Note: the requirement to follow the string with a space is obsolescent and is included as a concession to existing programs that use CONVERT . A Standard Program shall not depend on the existance of the space.

WORDLIST FORTH

(-- wid)

Creates a new empty word list, returning its word list identifier wid. The new word list is dynamically allocated in data space. Note that other ANS systems may create the new word list in another place.

WORDS ONLY

(--)

List the word names in the first word list of the search order in colums of 16 characters wide and a count at the end.

WORDS is implemented using pictured numeric output words. Its use will corrupt the transient region identified by #> .
See also: EVERY

WORDSPEED EXTRA

(-- addr)

a-addr is the address of a cell containing the delay after WORDS SEE DIS etc. in millseconds.

WRITE-FILE FORTH

(c-addr u fileid -- ior)

Write u characters from c-addr to the file identified by fileid starting at its current position. ior is the I/O result code.

At the conclusion of the operation FILE-POSITION returns a value past the characters written to the file and FILE-SIZE returns a

value greater than or equal to the value returned by
FILE-POSITION .
See also: READ-FILE WRITE-LINE

WRITE-GLOSS "write-glossary" FORTH
("name" --)
This word writes the glossary info from memory into a file.
The information may be collected from several source files.

WRITE-LINE FORTH
(c-addr u fileid -- ior)
Write u characters from c-addr followed by the line terminators
to the file identified by fileid starting at its current
position. ior is the I/O result code.

At the conclusion of the operation, FILE-POSITION returns a value
past the characters written to the file and FILE-SIZE returns a
value greater then or equal to the value returned by
FILE-POSITION .
See also: READ-FILE READ-LINE

WRITEX-FILE "write-x-file" EXTRA
(x-addr u fileid -- ior)
Write u characters from extended address x-addr to the file
specified by fileid. If no exception occurs, ior is zero.
Otherwise ior is the I/O result code.

X. "x-dot" EXTRA
(x-addr --)
Display the extended address x-addr as a four character segment
name or number as in .SEG , a colon and a four digit hexadecimal
number and a space.

XOR "x-or" FORTH
(x1 x2 -- x3)
x3 is the bit-by-bit logical exclusive-or of x1 with x2.

YES. DECOMPILER
(--)
Set the decompiler to normal.
See also: NO.

`[                 "left-bracket"                         FORTH`

(--)
Enter interpretation state. [is an immediate word.
See also:]

`[']               "bracket-tick"                         FORTH`

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics below to the current definition. Exception -13 occurs if name is not found.

Run-time: (-- xt)
Place name's execution token xt on the stack. The execution token compiled by the phrase " ['] X " is the same value returned by " ' X " outside of compilation state.
See also: ' POSTPONE

`[CHAR]            "bracket-char"                         FORTH`

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition.

Run-time: (-- char)
Place char char, the value of the first character of name, on the stack.
See also: CHAR

`[COMPILE]         "bracket-compile"                      FORTH`

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space.

If name has compilation semantics specified, append them to the current definition; otherwise append the execution semantics of name. Exception -13 occurs if name is not found.

[CTRL] "bracket-control" EXTRA

Interpretation: (i*x --)
This word is marked compile only. The default interpreter issues exception -14 when an attempt is made to execute this word.

Compilation: ("name" --)
Skip leading space delimiters. Parse name delimited by a space. Append the run-time semantics given below to the current definition. Exception -531 occurs when the character is not in the range {’@’..’_’}.

Run-time: (-- char)
Place char, the value of the first character of name, after conversion to a control character, on the stack.
See also: CTRL [CHAR]

[ELSE] "bracket-else" FORTH

(--)
Repeatedly skip leading spaces, parse and discard space-delimited words from the parse area, including nested occurences of [IF] ... [THEN] and [IF] ... [ELSE] ... [THEN] , until the word [THEN] has been parsed and discarded. If the parse area becomes exhausted, it is refilled as with REFILL . If the refilling of the input buffer fails, exception -58 occurs. [ELSE] is immediate.

[IF] "bracket-if" FORTH

(flag --)
If the flag is true, do nothing. Otherwise repeatedly skip leading spaces, parse and discard space-delimited words from the parse area, including nested occurences of [IF] ... [THEN] and [IF] ... [ELSE] ... [THEN] , until either the word [ELSE] or the word [THEN] has been parsed and discarded. If the parse area becomes exhausted, it is refilled as with REFILL . [IF] is immediate.

An ambiguous condition exists if [IF] is POSTPONEd. If the end of the input stream is reached and cannot be refilled before the terminating [ELSE] or [THEN] is parsed exception -58 occurs.

[THEN] "bracket-then" FORTH
(--)
Does nothing. [THEN] is immediate.

[]CELL "cell-array" EXTRA
(x a-addr1 -- a-addr2)
Multiply x by the size in address units of a cell and add it to a-addr1 giving a-addr2.

[]CHAR "char-array" EXTRA
(x c-addr1 -- c-addr2)
Multiply x by the size in address units of a character and add it to c-addr1 giving c-addr2.

[]DOUBLE "double-array" EXTRA
(x a-addr1 -- a-addr2)
Multiply x by the size in address units of a double-cell and add it to a-addr1 giving a-addr2.

[]KEY "key-array" EXTRA
(char | x -- addr)
Return the address that is associated with control keys and extended keys. Used to store an execution token that will be executed when that particular key is pressed during ACCEPT .

\ "backslash" FORTH
("ccc<eol>" --)
If BLK contains zero, parse and discard the remainder of the parse area; otherwise parse and discard the portion of the parse area corresponding to the remainder of the current line. \ is an immediate word.

\G EXTRA
("ccc<eol>" --)
If BLK contains zero, parse and discard the remainder of the parse area; otherwise parse and discard the portion of the parse area corresponding to the remainder of the current line. \G is an immediate word. Used in generating glossaries.

] "right-bracket" FORTH
(--)
Enter compilation state.

See also: [

{{ CRITERR

(--)

Redirect the DOS critical error handler to a harmless routine. Use this word only temporary to perform dangerous functions.

}} CRITERR

(--)

Restore the redirection of the DOS critical error handler. Use this word as soon as possible after {{ .