FORTH-83 STANDARD

A PUBLICATION OF THE FORTH STANDARDS TEAM

AUGUST 1983

The existence of a FORTH Standard does not in any respect preclude anyone, whether the individual has approved this Standard or not, from implementing, marketing, purchasing or using products, processes, or procedures not conforming to the Standard. FORTH Standards are subject to periodic review and users are cautioned to obtain the latest editions.

ISBN 0-914699-03-2

FORTH STANDARDS TEAM
P.O. BOX 4545
MOUNTAIN VIEW, CA 94040
USA

TABLE OF CONTENTS

APPENDICES

1. FOREWORD

FORTH is an integrated programming approach and computer language. FORTH was invented by Mr. Charles Moore specifically to increase programmer productivity in the development of computer related applications without sacrificing machine efficiency. FORTH is a layered environment containing the elements of a computer language as well as those of an operating system and a machine monitor. This extensible, layered environment provides for highly interactive program development and testing.

In the interests of transportability of application software written in FORTH, standardization efforts began in the mid-1970s by the European FORTH User's Group (EFUG). This effort resulted in the FORTH-77 Standard. As the language continued to evolve, an interim FORTH-78 Standard was published by the FORTH Standards Team. Following FORTH Standards Team meetings in 1979 the FORTH-79 Standard was published in 1980.

The FORTH Standards Team is comprised of individuals who have a great variety of experience and technical expertise with FORTH. The FORTH Standards Team consists of both users and implementers. Comments, proposals, and correspondence should be mailed to: FORTH Standards Team, P.O. Box 4545, Mountain View, CA 94040 USA.

FORTH's extensibility allows the language to be expanded and adapted to special needs and different hardware systems. A programmer or vendor may choose to strictly adhere with the standard, but the choice to deviate is acknowledged as beneficial and sometimes necessary. If the standard does not explicitly specify a requirement or restriction, a system or application may utilize any choice without sacrificing compliance to the standard provided that the system or application remains transportable and obeys the other requirements of the standard.

2.   PURPOSE

The purpose of this standard is to allow transportability of FORTH-83 Standard Programs in source form among FORTH-83 Standard Systems. A standard program shall execute equivalently on all standard systems.

3.   SCOPE

This standard shall apply to any FORTH-83 Standard Program executing on any FORTH-83 Standard System, provided sufficient computer resources (memory, mass storage) are available.

4. TRADEOFFS

When conflicting choices are made, the following order guides the Standards Team:

1) Functional correctness - known bounds, non-ambiguous;

2) Portability - repeatable results when programs are transported among Standard Systems;

3) Simplicity;

4) Naming clarity - uniformity of expression using descriptive rather than procedure names, i.e., [COMPILE] rather than 'C, and ALLOT rather than DP+! ;

5) Generality;

6) Execution speed;

7) Memory compactness;

8) Compilation speed;

9) Historical continuity;

10) Pronounceability;

11) Teachability.

5. DEFINITIONS OF TERMS

These are the definitions of the terms used within this Standard.

address, byte

An unsigned 16-bit number that locates an 8-bit byte in a standard FORTH address space over the range {0..65,535}. It may be a native machine address or a representation on a virtual machine, locating the addr-th byte within the virtual byte address space. Addresses are treated as unsigned numbers. See: "arithmetic, two's complement"

address, compilation

The numerical value compiled for a FORTH word definition which identifies that definition. The address interpreter uses this value to locate the machine code corresponding to each definition.

address, native machine

The natural address representation of the host computer.

address, parameter field

The address of the first byte of memory associated with a word definition for the storage of compilation addresses (in a colon definition), numeric data, text characters, etc.

arithmetic, two's complement

Arithmetic is performed using two's complement integers within a field of either 16 or 32 bits as indicated by the operation. Addition and subtraction of two's complement integers ignore any overflow condition. This allows numbers treated as unsigned to produce the same results as if the numbers had been treated as signed.

block

The 1024 bytes of data from mass storage which are referenced by block numbers in the range {0..the number of blocks available -1}. The actual amount of data transferred and the translation from block number to device and physical record is a function of the implementation. See: "block buffer" "mass storage"

block buffer

A 1024-byte memory area where a block is made temporarily available for use. Block buffers are uniquely assigned to blocks. See: "9.7 Multiprogramming Impact"

byte

An assembly of 8 bits. In reference to memory, it is the storage capacity for 8 bits.

5. DEFINITIONS OF TERMS

character

A 7-bit number the significance of which is given by the ASCII standard. When contained in a larger field, the higher order bits are zero. See: "6. REFERENCES"

compilation

The action of converting text words from the input stream into an internal form suitable for later execution. When in the compile state, the compilation addresses of FORTH words are compiled into the dictionary for later execution by the address interpreter. Numbers are compiled to be placed on the data stack when later executed. Numbers are accepted from the input stream unsigned or negatively signed and converted using the value of BASE . See: "number" "number conversion" "interpreter, text"

defining word

A word that, when executed, creates a new dictionary entry in the compilation vocabulary. The new word name is taken from the input stream. If the input stream is exhausted before the new name is available, an error condition exists. Example of defining words are: : CONSTANT CREATE

definition

See: "word definition"

dictionary

A structure of word definitions in computer memory which is extensible and grows toward higher memory addresses. Entries are organized in vocabularies to aid location by name. See: "search order"

display

The process of sending one or more characters to the current output device. These characters are typically displayed or printed on a terminal. The selection of the current output device is system dependent.

division, floored

Integer division in which the remainder carries the sign of the divisor or is zero, and the quotient is rounded to its arithmetic floor. Note that, except for error conditions, n1 n2 SWAP OVER /MOD ROT * + is identical to n1. See: "floor, arithmetic"

Examples:

dividend	divisor	remainder	quotient
10	7	3	1
-10	7	4	-2
10	-7	-4	-2
-10	-7	-3	1

equivalent execution

A standard program will produce the same results, exclusive of timing dependencies, when given the same inputs on any Standard System which has sufficient resources to execute the program. Only standard source programs are transportable.

error condition

An exceptional condition which requires action by the system which may be other than the expected function. Refer to the section "10. Error Conditions".

false

A zero number represents the false state of a flag.

flag

A number that may have one of two logical states, false or true. See: "false" "true"

floor, arithmetic

If z is any real number, then the floor of z is the greatest integer less than or equal to z.

The floor of +.6 is 0
The floor of -.4 is -1

free field format

Numbers are converted using the value of BASE and then displayed with no leading zeros. A trailing space is displayed. The number of characters displayed is the minimum number of characters, at least one, to uniquely represent the number. See: "number conversion"

glossary

A set of explanations in natural language to describe the corresponding computer execution of word definitions.

immediate word

A word which executes when encountered during compilation or interpretation. Immediate words handle special cases during compilation. See, for example, IF LITERAL ." etc.

input stream

A sequence of characters available to the system, for processing by the text interpreter. The input stream conventionally may be taken from the current input device (via the text input buffer) and mass storage (via a block buffer). BLK , >IN , TIB and #TIB specify the input stream. Words using or altering BLK , >IN , TIB and #TIB are responsible for maintaining and restoring control of the input stream.

The input stream extends from the offset value of >IN to the size of the input stream. If BLK is zero the input stream is contained within the area addressed by TIB and is #TIB bytes long. If BLK is non-zero the input stream is contained within the block buffer specified by BLK and is 1024 bytes long. See: "11.8 Input Text"

interpreter, address

The machine code instructions, routine or other facilities that execute compiled word definitions containing compilation addresses.

interpreter, text

The word definitions(s) that repeatedly accepts a word name from the input stream, locates the corresponding compilation address and starts the address interpreter to execute it. Text from the input stream interpreted as a number leaves the corresponding value on the data stack. Numbers are accepted from the input stream unsigned or negatively signed and converted using the value of BASE . See: "number" "number conversion"

layers

The grouping of word names of each Standard word set to show like characteristics. No implementation requirements are implied by this grouping.

layer, compiler

Word definitions which add new procedures to the dictionary or which aid compilation by adding compilation addresses or data structures to the dictionary.

layer, devices

Word definitions which allow access to mass storage and computer peripheral devices.

layer, interpreter

Word definitions which support vocabularies, terminal output, and the interpretation of text from the text input buffer or a mass storage device by executing the corresponding word definitions.

layer, nucleus

Word definitions generally defined in machine code that control the execution of the fundamental operations of a virtual FORTH machine. This includes the address interpreter.

load

Redirection of the text interpreter's input stream to be from mass storage. This is the general method for compilation of new definitions into the dictionary.

mass storage

Storage which might reside outside FORTH's address space. Mass storage data is made available in the form of 1024-byte blocks. A block is accessible within the FORTH address space in a block buffer. When a block has been indicated as UPDATEed (modified) the block will ultimately be transferred to mass storage.

number

When values exist within a larger field, the most-significant bits are zero. 16-bit numbers are represented in memory by addressing the first of two bytes at consecutive addresses. The byte order is unspecified by this Standard. Double numbers are represented on the stack with the most-significant 16 bits (with sign) most accessible. Double numbers are represented in memory by two consecutive 16-bit numbers. The address of the least significant 16 bits is two greater than the address of the most significant 16 bits. The byte order within each 16-bit field is unspecified. See: "arithmetic, two's complement" "number types" "9.8 Numbers" "11.7 Stack Parameters"

number conversion

Numbers are maintained internally in binary and represented externally by using graphic characters within the ASCII character set. Conversion between the internal and external forms is performed using the current value of BASE to determine the digits of a number. A digit has a value ranging from zero to the value of BASE-1. The digit with the value zero is represented by the ASCII character "0" (position 3/0 with the decimal equivalent of 48). This representation of digits proceeds through the ASCII character set to the character "(" corresponding to the decimal value 9. For digits with a value exceeding 9, the ASCII graphic characters beginning with the character "A" (position 4/1 with the decimal equivalent 65) corresponding to the decimal value 10 are used. This sequence then continues up to and including the digit with the decimal value 71 which is represented by the ASCII character "~" (position 7/14 with a decimal equivalent 126). A negative number may be represented by preceding the digits with a single leading minus sign, the character "-".

number types

All number types consist of some number of bits. These bits are either arbitrary or are weighted.

Signed and unsigned numbers use weighted bits. Weighted bits within a number have a value of a power of two beginning with the rightmost (least-significant) bit having the value of two to the zero power. This weighting continues to the leftmost bit increasing the power by one for each bit. For an unsigned number this weighting pattern includes the leftmost bit; thus, for an unsigned 16-bit number the weight of the leftmost bit is 32,768. For a signed number this weighting pattern includes the leftmost bit but the weight of the leftmost bit is negated; thus, for a signed 16-bit number the weight of the leftmost bit is -32,768. This weighting pattern for signed numbers is called two's complement notation.

Unspecified weighted numbers are either unsigned numbers or signed numbers; program context determines whether the number is signed or unsigned. See: "11.7 Stack Parameters"

pictured numeric output

The use of numeric output definitions which convert numerical values into text strings. These definitions are used in a sequence which resembles a symbolic 'picture' of the desired text format. Conversion proceeds from least-significant digit to most-significant digit, and converted characters are stored from higher memory addresses to lower.

program

A complete specification of execution to achieve a specific function (application task) expressed in FORTH source code form.

receive

The process of obtaining one character from the current input device. The selection of the current input device is system dependent.

recursion

The process of self-reference, either directly or indirectly.

return

The means of indicating the end of text by striking a key on an input device. The key used is system dependent. This key is typically called "RETURN", "CARRIAGE RETURN", or "ENTER".

screen

Textual data arranged for editing. By convention, a screen consists of 16 lines (numbered 0 through 15) of 64 characters each. Screens usually contain program source text, but may be used to view mass storage data. The first byte of a screen occupies the first byte of a mass storage block, which is the beginning point for text interpretation during a load.

5. DEFINITIONS OF TERMS

search order

A specification of the order in which selected vocabularies in the dictionary are searched. Execution of a vocabulary makes it the first vocabulary in the search order. The dictionary is searched whenever a word is to be located by its name. This order applies to all dictionary searches unless otherwise noted. The search order begins with the last vocabulary executed and ends with FORTH , unless altered in a system dependent manner.

source definition

Text consisting of word names suitable for compilation or execution by the text interpreter. Such text is usually arranged in screens and maintained on a mass storage device.

stack, data

A last in, first out list consisting of 16-bit binary values. This stack is primarily used to hold intermediate values during execution of word definitions. Stack values may represent numbers, characters, addresses, boolean values, etc.

When the name 'stack' is used alone, it implies the data stack.

stack, return

A last in, first out list which contains the addresses of word definitions whose execution has not been completed by the address interpreter. As a word definition passes control to another definition, the return point is placed on the return stack.

The return stack may cautiously be used for other values.

string, counted

A sequence of consecutive 8-bit bytes located in memory by their low memory address. The byte at this address contains a count {0..255} of the number of bytes following which are part of the string. The count does not include the count byte itself. Counted strings usually contain ASCII characters.

string, text

A sequence of consecutive 8-bit bytes located in memory by their low memory address and length in bytes. Strings usually, but not exclusively, contain ASCII characters. When the term 'string' is used alone or in conjunction with other words it refers to text strings.

structure, control

A group of FORTH words which when executed alter the execution sequence. The group starts and terminates with compiler words. Examples of control structures: DO ... LOOP DO ... +LOOP BEGIN ... WHILE ... REPEAT BEGIN ... UNTIL IF ... THEN IF ... ELSE ... THEN See: "9.9 Control Structures"

transportability

This term indicates that equivalent execution results when a program is executed on other than the system on which it was created. See: "equivalent execution"

true

A non-zero value represents the true state of a flag. Any non-zero value will be accepted by a standard word as 'true'; all standard words return a 16-bit value with all bits set to one when returning a 'true' flag.

user area

An area in memory which contains the storage for user variable.

variable, user

A variable whose data storage area is usually located in the user area. Some system variables are maintained in the user area so that the words may be re-entrant to different users.

vocabulary

An ordered list of word definitions. Vocabularies are an advantage in separating different word definitions that may have the same name. More than one definition with the same name can exist in one vocabulary. The latter is called a redefinition. The most recently created redefinition will be found when the vocabulary is searched.

vocabulary, compilation

The vocabulary into which new word definitions are appended.

word

A sequence of characters terminated by one blank or the end of the input stream. Leading blanks are ignored. Words are usually obtained via the input stream.

word definition

A named FORTH execution procedure compiled into the dictionary. Its execution may be defined in terms of machine code, as a sequence of compilation address, or other compiled words.

word name

The name of a word definition. Word names are limited to 31 characters and may not contain an ASCII space. If two definitions have different word names in the same vocabulary they must be uniquely findable when this vocabulary is searched. See: "vocabulary" "9.5.3 EXPECT"

word set
A named group of FORTH word definitions in the Standard.

word set, assembler extension
Additional words which facilitate programming in the native machine language of the computer which are by nature system dependent.

word set, double number extension
Additional words which facilitate manipulation of 32-bit numbers.

word set, required
The minimum words needed to compile and execute Standard Programs.

word set, system extension
Additional words which facilitate the access to internal system characteristics.

word, standard
A named FORTH procedure definition, in the Required word set or any extension word sets, formally reviewed and accepted by the Standards Team.

6. REFERENCES

The following document is considered to be a portion of this Standard:

American National Standard Code for Information Interchange, X3.4-1977 (ASCII), American National Standards Institute, 1430 Broadway, New York, NY 10018, USA.

The following documents are noted as pertinent to the FORTH-83 Standard, but are not part of this Standard.

FORTH-77, FORTH Users Group, FST-780314

FORTH-78, FORTH International Standards Team

FORTH-79, FORTH Standards Team

FORTH-83 STANDARD, Appendices, FORTH Standards Team

Webster's Collegiate Dictionary shall be used to resolve conflicts in spelling and English word usage.

7.   REQUIREMENTS

7.1 Documentation Requirements

7.1.1   Each Standard System shall be accompanied by a statement of:

1. System dictionary space used in bytes;

2. Application dictionary space available in bytes;

3. Data space in bytes;

4. Return stack space in bytes;

5. Mass storage block ranges used by the system;

6. Mass storage block ranges available to applications;

7. Operator's terminal facilities available;

8. System action taken upon each of the general or specified error conditions as identified in this standard.

7.1.2   Each standard program shall be accompanied by a statement of the minimum requirements for:

1. Dictionary space in bytes;

2. Data stack space in bytes;

3. Return stack space in bytes;

4. Mass storage block ranges;

5. Operator's terminal facilities

7.2 Testing Requirements

The following host computer configuration is specified as a minimum environment for testing against this Standard. Applications may require different capacities.

1. 2000 bytes of memory for application dictionary;

2. Data stack of 64 bytes;

7. REQUIREMENTS

3. Return stack of 48 bytes;

4. Mass storage capacity of 32 blocks, numbered 0 through 31;

5. One ASCII input/output device acting as an operator's terminal.

8.   COMPLIANCE AND LABELING

The FORTH Standards Team hereby specifies the requirements for labeling of systems and applications so that the conditions for program portability may be established.

A Standard System may use the specified labeling if it complies with the terms of this Standard and meets the particular Word Set definitions.

A Standard Program (application) may use the specified labeling if it utilizes the specified Standard System according to this Standard and executes equivalently on any such system.

In a system or application, a standard word may not be redefined to perform a different function within the vocabulary FORTH.

FORTH Standard

A system may be labeled:

FORTH-83 Standard

if it includes all of the Required Word Set in either source or object form and complies with the text of this Standard. After executing "FORTH-83" the dictionary must contain all of the Required Word Set in the vocabulary FORTH, as specified in this Standard.

Standard Sub-set

A system may be labeled:

FORTH-83 Standard Sub-set

if it includes a portion of the Required Word Set and complies with the remaining text of this Standard. However, no Required Word may be present with a non-standard definition.

Standard with Extensions

A system may be labeled:

FORTH-83 Standard with <name> Standard Extension(s)

if it comprises a FORTH-83 Standard System and one or more Standard Extension Word Set(s). For example, a designation would be in the form:

8. COMPLIANCE AND LABELING

FORTH-83 Standard with Double-Number Standard Extension

Standard Program

A FORTH source program which executes equivalently on any Standard System may be labeled:

FORTH-83 Standard Program

See: "equivalent execution" "7. REQUIREMENTS"

Standard Program with Environmental Dependencies

A program which is standard in all ways except for specific environmentally dependent words may be labeled:

FORTH-83 Standard Program with Environmental Dependencies

if the following additional requirements are met:

1) Environmental dependencies (including hardware dependencies) shall be factored into an isolated set of application word definitions.

2) Each environmentally dependent word definition must be fully documented, including all dependencies in a manner at least as detailed as the standard words.

9. USAGE

9.1 Words Names and Word Definitions

A Standard Program may reference only the definitions of the Required Word Set and Standard Extensions and definitions which are subsequently defined in terms of these words. Furthermore, A Standard Program must use the standard words as required by any conventions of this Standard. Equivalent execution must result from Standard Programs.

The implementation of a Standard System may use words and techniques outside the scope of the Standard, provided that no program running on that system is required to use words outside the Standard for normal operation.

If a Standard System or Standard Program redefines Standard definitions within the FORTH vocabulary, these definitions must comply with the Standard.

9.2 Addressable Memory

The FORTH system may share the dictionary space with the user's application. The native addressing protocol of the host computer is beyond the scope of this Standard.

Therefore, in a Standard Program, the user may only operate on data which was stored by the application. No exceptions!

A Standard Program may address:

1. parameter fields of words created with CREATE , VARIABLE , and user defined words which execute CREATE ;

2. dictionary space ALLOTted;

3. data in a valid mass storage block buffer.
   See: "9.7 Multiprogramming Impact";

4. data area of user variables;

5. text input buffer and PAD up to the amount specified as the minimum for each area.

A Standard Program may NOT address:

1. directly into the data or return stacks;

2. into a definition's name field, link field, or code field;

3. into a definition's parameter field if not stored by the application.

9.3 Return Stack

A Standard Program may cautiously use the return stack with the following restrictions:

The return stack may not be accessed inside a do-loop for values placed on the return stack before the loop was entered. Further, neither I nor J may be used to obtain the index of a loop if values are placed and remain on the return stack within the loop. When the do-loop is executed all values placed on the return stack within that loop must be removed before LOOP , +LOOP , or LEAVE is executed. Similarly, all values placed on the return stack within a colon definition must be removed before the colon definition is terminated at ; or before EXIT is executed.

9.4 Compilation

The system uses the return stack and the dictionary in a system dependent manner during the compilation of colon definitions. Some words use the data stack in a system dependent manner during compilation. See: "sys (11.7)"

9.5 Terminal Input and Output

9.5.1 KEY

A Standard System must receive all valid ASCII characters. Each KEY receives one ASCII character, with more-significant bits environmentally dependent and might be zero. KEY must receive as many bits as are obtainable. A Standard Program without environmental dependencies may only use the least significant 7-bit ASCII character received by KEY . For example: KEY 127 AND

9.5.2 EXPECT

Control characters may be processed to allow system dependent editing of the characters prior to receipt. Therefore, a Standard Program may not anticipated that control characters can be received.

9.5.3 EMIT

Because of the potential non-transportable action by terminal devices of control characters, the use of ASCII control characters is an environmental dependency. Each EMIT deals with only one ASCII character. The ASCII character occupies the least-significant 7 bits; the more-significant bits may be environmentally dependent. Using the more-significant bits when other than zero is an environmentally dependent usage. EMIT must display as many bits as can be sent.

9.5.4 TYPE

Because of the potential non-transportable action by terminal devices of control characters, the use of ASCII control characters is an environmental dependency.

9.6 Transporting Programs Between Standard Systems

Further usage requirements are expected to be added for transporting programs between Standard Systems.

9.7 Multiprogramming Impact

In a multiprogrammed system, Device Layer words and those words which implicitly reference the Device Layer words may relinquish control of the processor to other tasks. Although there is insufficient experience to specify a standard for multiprogramming, historical usage dictates that a programmer be aware of the potential impact with regard to resources shared between tasks. The only shared resources specified within the Standard are block buffers. Therefore the address of a block buffer returned by BLOCK or BUFFER becomes invalid during and after the execution of any word marked by the attribute M in the glossary or any words executing them. A block buffer is valid only if its address is valid. See: "11.4 Attributes"

9.8 Numbers

Interpreted or compiled numbers are in the range {-32,768..65,535}. See: "number conversion"

9.9 Control Structures

Control structures are compiled inside colon definitions. Control structures can be nested but cannot overlap. For additional limitations see DO .

10. ERROR CONDITIONS

10.1 Possible Actions on an Error

When an error condition occurs, a Standard System may take one or more of the following actions:

1. ignore and continue;

2. display a message;

3. execute a particular word;

4. set interpret state and interpret a block;

5. set interpret state and begin interpretation;

6. other system dependent actions.

See: "7.1 Documentation Requirements"

10.2 General Error Conditions

The following error conditions apply in many situations. These error conditions are listed below, but may occur at various times and with various words.

1. input stream exhausted before encountering a required <name> or delimiting character;

2. insufficient stack space or insufficient number of stack entries during text interpretation or compilation;

3. a word not found and not a valid number, during text interpretation or compilation;

4. compilation of incorrectly nested control structures;

5. execution of words restricted to compilation only, when not in the compile state and while not compiling a colon definition;

6. FORGETting within the system to a point that removes a word required for correct execution;

7. insufficient space remaining in the dictionary;

8. a stack parameter out of range, e.g., a negative number when a +n was specified in the glossary;

10. ERROR CONDITIONS

9. correct mass storage read or write was not possible.

11. GLOSSARY NOTATION

11.1 Order

The glossary definitions are listed in ASCII alphabetical order.

11.2 Capitalization

Word names are capitalized throughout this Standard.

11.3 Stack Notation

The stack parameters input to and output from a definition are described using the notation:

```
     before -- after
```

before	stack parameters before execution
after	stack parameters after execution

In this notation, the top of the stack is to the right. Words may also be shown in context when appropriate.

Unless otherwise noted, all stack notation describes exectution time. If it applies at compile time, the line is followed by: (compiling) .

11.4 Attributes

Capitalized symbols indicate attributes of the defined words:

C The word may only be used during compilation of a colon definition.

I Indicates that the word is IMMEDIATE and will execute during compilation, unless special action is taken.

M This word has a potential multiprogramming impact.
See: "9.7 Multiprogramming Impact"

U A user variable.

11.5 Serial Numbers

When a substantive alteration to a word's definition is made or when a new word is added, the serial number will be the last two digits of the year of the Standard in which such change was made (i.e., "83"). When such change is made within a Working Draft, the number will be suffixed with the character identifying the draft (i.e., "83A").

11.6 Pronunciation

The natural language pronunciation of word names is given in double quotes (") where it differs from English pronunciation.

11.7 Stack Parameters

Unless otherwise stated, all references to numbers apply to 16-bit signed integers. The implied range of values is shown as {from..to}. The contents of an address is shown by double braces, particularly for the contents of variables, i.e., BASE {{2..72}}.

The following are the stack parameter abbreviations and types of numbers used throughout the glossary. These abbreviations may be suffixed with a digit to differentiate multiple parameters of the same type.

11. GLOSSARY NOTATION

Stack Abbrv.	Number Type	Range in Decimal	Minimum Field
flag	boolean	0=false, else=true	16
true	boolean	-1 (as a result)	16
false	boolean	0	0
b	bit	{0..1}	1
char	character	{0..127}	7
8b	8 arbitrary bits (byte)	not applicable	8
16b	16 arbitrary bits	not applicable	16
n	number (weighted bits)	{-32,768..32,767}	16
+n	positive number	{0..32,767}	16
u	unsigned number	{0..65,535}	16
w	unspecified weighted number (n or u)	{-32,768..65,535}	16
addr	address (same as u)	{0..65,535}	16
32b	32 arbitrary bits	not applicable	32
d	double number	{-2,147,483,648.. 2,147,483,647}	32
+d	positive double number	{0..2,147,483,647}	32
ud	unsigned double number	{0..4,294,967,265}	32
wd	unspecified weighted double number (d or ud)	{-2,147,483,648.. 4,294,967,295}	32
sys	0, 1, or more system dependent stack entries	not applicable	na

Any other symbol refers to an arbitrary signed 16-bit integer in the range {-32,768..32,767}, unless otherwise noted.

Because of the use of two's complement arithmetic, the signed 16-bit number (n) -1 has the same bit representation as the unsigned number (u) 65,535. Both of these numbers are within the set of unspecified weighted numbers (w). See: "arithmetic, two's complement" "number" "number types" "stack, data"

11.8 Input Text

<name>

An arbitrary FORTH word accepted from the input stream. This notation refers to text from the input stream, not to values on the data stack. See: "10.2 General Error Conditions"

ccc

A sequence of arbitrary characters accepted from the input stream until the first occurrence of the specified delimiting character. The delimiter is accepted from the input stream, but is not one of the characters ccc and is therefore not otherwise processed. This notation refers to text from the input stream, not to values on the data stack. Unless noted otherwise, the number of characters accepted may be from 0 to 255. See: "10.2 General Error Conditions"

11.9 References to other words and definitions

Glossary definitions may refer to other glossary definitions or to definitions of terms. Such references are made using the expression "See:". These references provide additional information which apply as if the information is a portion of the glossary entry using "See:".

12. REQUIRED WORD SET

12.1 The Required Word Set Layers

The words of the Required Word Set are grouped to show like characteristics. No implementation requirements should be inferred from this grouping.

Nucleus layer

```
     !  *  */  */MOD  +  +!  -  /  /MOD  0<  0=  0>  1+  1-  2+
     2-  2/  <  =  >  >R  ?DUP  @  ABS  AND  C!  C@  CMOVE
     CMOVE>  COUNT  D+  D<  DEPTH  DNEGATE  DROP  DUP  EXECUTE
     EXIT  FILL  I  J  MAX  MIN  MOD  NEGATE  NOT  OR  OVER  PICK
     R>  R@  ROLL  ROT  SWAP  U<  UM*  UM/MOD  XOR
```

Device layer

```
     BLOCK  BUFFER  CR  EMIT  EXPECT  FLUSH  KEY  SAVE-BUFFERS
     SPACE  SPACES  TYPE  UPDATE
```

Interpreter layer

```
     #  #>  #S  #TIB  '  (  -TRAILING  .  .(  <#  >BODY  >IN
     ABORT  BASE  BLK  CONVERT  DECIMAL  DEFINITIONS  FIND
     FORGET  FORTH  FORTH-83  HERE  HOLD  LOAD  PAD  QUIT  SIGN
     SPAN  TIB  U.  WORD
```

Compiler layer

```
     +LOOP  ,  ."  :  ;  ABORT" ALLOT  BEGIN  COMPILE  CONSTANT
     CREATE  DO  DOES>  ELSE  IF  IMMEDIATE  LEAVE  LITERAL  LOOP
     REPEAT  STATE  THEN  UNTIL  VARIABLE  VOCABULARY  WHILE  [
     [']  [COMPILE]  ]
```

12.2 The Required Word Set Glossary

! 16b addr -- 79 "store"
16b is stored at addr.

# +d1 -- +d2 79 "sharp"
The remainder of +d1 divided by the value of BASE is converted to an ASCII character and appended to the output string toward lower memory addresses. +d2 is the quotient and is maintained for further processing. Typically used between <# and #> .

#> 32b -- addr +n 79 "sharp-greater"
Pictured numeric output conversion is ended dropping 32b. addr is the address of the resulting output string. +n is the number of characters in the output string. addr and +n together are suitable for TYPE .

#S +d -- 0 0 29 "sharp-s"
+d is converted appending each resultant character into the pictured numeric output string until the quotient (see: #) is zero. A single zero is added to the output string if the number was initially zero. Typically used between <# and #> .

#TIB -- addr U,83 "number-t-i-b"
The address of a variable containing the number of bytes in the text input buffer. #TIB is accessed by WORD when BLK is zero. {{0..capacity of TIB}} See: "input stream"

' -- addr M,83 "tick"
Used in the form:
' <name>
addr is the compilation address of <name>. An error condition exists if <name> is not found in the currently active search order.

(-- I,M,83 "paren"
-- (compiling)
Used in the form:
(ccc)
The characters ccc, delimited by) (closing parenthesis), are considered comments. Comments are not otherwise processed. The blank following (is not part of ccc. (may be freely used while interpreting or compiling. The number of characters in ccc may be zero to the number of characters remaining in the input stream up to the closing parenthesis.

* w1 w2 -- w3 79 "times"
w3 is the least-significant 16 bits of the arithmetic product of w1 times w2.

*/ n1 n2 n3 -- n4 83 "times-divide"

n1 is first multiplied by n2 producing an intermediate 32-bit result. n4 is the floor of the quotient of the intermediate 32-bit result divided by the divisor n3. The product of n1 times n2 is maintained as an intermediate 32-bit result for greater precision than the otherwise equivalent sequence: n1 n2 * n3 / . An error condition results if the divisor is zero or if the quotient falls outside of the range {-32,768..32,767}. See: "division, floored"

*/MOD n1 n2 n3 -- n4 n5 83 "times-divide-mod"

n1 is first multiplied by n2 producing an intermediate 32-bit result. n4 is the remainder and n5 is the floor of the quotient of the intermediate 32-bit result divided by the divisor n3. A 32-bit intermediate product is used as for */ . n4 has the same sign as n3 or is zero. An error condition results if the divisor is zero or if the quotient falls outside of the range {-32,768..32,767}. See: "division, floored"

+ w1 w2 -- w3 79 "plus"

w3 is the arithmetic sum of w1 plus w2.

+! w1 addr -- 79 "plus-store"

w1 is added to the w value at addr using the convention for + . This sum replaces the original value at addr.

+LOOP n -- C,I,83 "plus-loop"
sys -- (compiling)

n is added to the loop index. If the new index was incremented across the boundary between limit-1 and limit then the loop is terminated and loop control parameters are discarded. When the loop is not terminated, execution continues to just after the corresponding DO . sys is balanced with its corresponding DO . See: DO

, 16b -- 79 "comma"

ALLOT space for 16b then store 16b at HERE 2- .

- w1 w2 -- w3 79 "minus"

w3 is the result of subtracting w2 from w1.

-TRAILING addr +n1 -- addr +n2 79 "dash-trailing"

The character count +n1 of a text string beginning at addr is adjusted to exclude trailing spaces. If +n1 is zero, then +n2 is also zero. If the entire string consists of spaces, then +n2 is zero.

. n -- M,79 "dot"

The absolute value of n is displayed in a free field format with a leading minus sign if n is negative.

```
."          --                          C,I,83    "dot-quote"
            --   (compiling)
     Used in the form:
            ." ccc"
     Later execution will display the characters ccc up to but
     not including the delimiting " (close-quote).  The blank
     following ." is not part of ccc.

.(          --                          I,M,83    "dos-paren"
            --                          (compiling)
     Used in the form:
            .( ccc)
     The characters ccc up to but not including the delimiting )
     (closing parenthesis) are displayed.  The blank following .(
     is not part of ccc.

/           n1 n2 -- n3                 83           "divide"
     n3 is the floor of the quotient of n1 divided by the divisor
     n2.  An error condition results if the divisor is zero or if
     the quotient falls outside of the range {-32,768..32,767}.
     See:  "division, floored"

/MOD        n1 n2 -- n3 n4              83       "divide-mod"
     n3 is the remainder and n4 the floor of the quotient of n1
     divided by the divisor n2.  n3 has the same sign as n2 or is
     zero.  An error condition results if the divisor is zero or
     if the quotient falls outside of the range
     {-32,768..32,767}.  See:  "division, floored"

0<          n -- flag                   83        "zero-less"
     flag is true if n is less than zero (negative).

0=          w -- flag                   83      "zero-equals"
     flag is true if w is zero.

0>          n -- flag                   83     "zero-greater"
     flag is true if n is greater than zero.

1+          w1 -- w2                    79         "one-plus"
     w2 is the result of adding one to w1 according to the
     operations of + .

1-          w1 -- w2                    79        "one-minus"
     w2 is the result of subtracting one from w1 according to the
     operation of - .

2+          w1 -- w2                    79         "two-plus"
     w2 is the result of adding two to w1 according to the
     operation of + .

2-          w1 -- w2                    79        "two-minus"
     w2 is the result of subtracting two from w1 according to the
     operation of - .
```

2/ n1 -- n2 83 "two-divide"

n2 is the result of arithmetically shifting n1 right one bit. The sign is included in the shift and remains unchanged.

: -- sys M,79 "colon"

A defining word executed in the form:

: <name> ... ;

Create a word definition for <name> in the compilation vocabulary and set compilation state. The search order is changed so that the first vocabulary in the search order is changed so that the first vocabulary in the search order is replaced by the compilation vocabulary. The compilation vocabulary is unchanged. The text from the input stream is subsequently compiled. <name> is called a "colon definition". The newly created word definition for <name> cannot be found in the dictionary until the corresponding ; or ;CODE is successfully processed.

An error condition exists if a word is not found and cannot be converted to a number or if, during compilation from mass storage, the input stream is exhausted before encountering ; or ;CODE . sys is balanced with its corresponding ; . See: "compilation" "9.4 Compilation"

; -- C,I,79 "semi-colon"
sys -- (compiling)

Stops compilation of a colon definition, allows the <name> of this colon definition to be found in the dictionary, sets interpret state and compiles EXIT (or a system dependent word which performs an equivalent function). sys is balanced with its corresponding : . See: EXIT : "stack, return" "9.4 Compilation"

< n1 n2 -- flag 83 "less-than"

flag is true if n1 is less than n2.
-32678 32767 < must return true.
-32768 0 < must return true.

<# -- 79 "less-sharp"

Initialize pictured numeric output conversion. The words:

#> #S <# HOLD SIGN

can be used to specify the conversion of a double number into an ASCII text string stored in right-to-left order.

= w1 w2 -- flag 83 "equals"

flag is true if w1 is equal to w2.

> n1 n2 -- flag 83 "greater-than"

flag is true if n1 is greater than n2.
-32768 32767 > must return false.
-32768 0 > must return false.

```
>BODY        addr1 -- addr2                83          "to-body"
     addr2 is the parameter field address corresponding to the
     compilation address addr1.  See:  "9.2 Addressable Memory"

>IN          -- addr                       U,79          "to-in"
     The address of a variable which contains the present
     character offset within the input stream {{0..the number of
     characters in the input stream}}.  See:  WORD

>R           16b --                        C,79           "to-r"
     Transfers 16b to the return stack.  See  "9.3 Return Stack"

?DUP         16b -- 16b 16b                79    "question-dupe"
     or      0 -- 0
     Duplicate 16b if it is non-zero.

@            addr -- 16b                   79            "fetch"
     16b is the value at addr.

ABORT                                      79
     Clears the data stack and performs the function of QUIT .
     No message is displayed.

ABORT"       flag --                       C,I,83  "abort-quote"
             --   (compiling)
     Used in the form:
             flag ABORT" ccc"
     When later executed, if flag is true the characters ccc,
     delimited by " (close-quote), are displayed and then a
     system dependent error abort sequence, including the
     function of ABORT , is performed.  If flag is false, the
     flag is dropped and execution continues.  The blank
     following ABORT" is not part of ccc.

ABS          n -- u                        79         "absolute"
     u is the absolute value of n.  If n is -32,768 then u is the
     same value.  See:  "arithmetic, two's complement"

ALLOT        w --                          79
     Allocates w bytes in the dictionary.  The address of the
     next available dictionary entry is updated accordingly.

AND          16b1 16b2 -- 16b3             79
     16b3 is the bit-by-bit logical 'and' of 16b1 with 16b2.

BASE         -- addr                       U,83
     The address of a variable containing the current numeric
     conversion radix.  {{2..72}}
```

BEGIN -- C,I,79
-- sys (compiling)

Used in the form:

```
        BEGIN ... flag UNTIL
or
        BEGIN ... flag WHILE ... REPEAT
```

BEGIN marks the start of a word sequence for repetitive execution. A BEGIN-UNTIL loop will be repeated until flag is true. A BEGIN-WHILE-REPEAT will be repeated until flag is false. The words after UNTIL or REPEAT will be executed when either loop is finished. sys is balanced with its corresponding UNTIL or WHILE . See: "9.9 Control Structures"

BLK -- addr U,79 "b-l-k"

The address of a variable containing the number of the mass storage block being interpreted as the input stream. If the value of BLK is zero the input stream is taken from the text input buffer. {{0..the number of blocks available -1}} See: TIB "input stream"

BLOCK u -- addr M,83

addr is the address of the assigned buffer of the first byte of block u. If the block occupying that buffer is not block u and has been UPDATEed it is transferred to mass storage before assigning the buffer. If block u is not already in memory, it is transferred from mass storage into an assigned block buffer. A block may not be assigned to more than one buffer. If u is not an available block number, an error condition exists. Only data within the last buffer referenced by BLOCK or BUFFER is valid. The contents of a block buffer must not be changed unless the change may be transferred to mass storage.

BUFFER u -- addr M,83

Assigns a block buffer to block u. addr is the address of the first byte of the block within its buffer. This function is fully specified by the definition for BLOCK except that if the block is not already in memory it might not be transferred from mass storage. The contents of the block buffer assigned to block u by BUFFER are unspecified.

C! 16b addr -- 79 "c-store"

The least-significant 8 bits of 16b are stored into the byte at addr.

C@ addr -- 8b 79 "c-fetch"

8b is the contents of the byte at addr.

CMOVE addr1 addr2 u -- 83 "c-move"

Move u bytes beginning at address addr1 to addr2. The byte at addr1 is moved first, proceeding toward high memory. If u is zero nothing is moved.

CMOVE> addr1 addr2 u -- 83 "c-move-up"

Move the u bytes at address addr1 to addr2. The move begins by moving the byte at (addr1 plus u minus 1) to (addr2 plus u minus 1) and proceeds to successively lower addresses for u bytes. If u is zero nothing is moved. (Useful for sliding a string towards higher addresses).

COMPILE -- C,83

Typically used in the form:

: <name> ... COMPILE <namex> ... ;

When <name> is executed, the compilation address compiled for <namex> is compiled and not executed. <name> is typically immediate and <namex> is typically not immediate. See: "compilation"

CONSTANT 16b -- M,83

A defining word executed in the form:

16b CONSTANT <name>

Creates a dictionary entry for <name> so that when <name> is later executed, 16b will be left on the stack.

CONVERT +d1 addr1 -- +d2 addr2 79

+d2 is the result of converting the characters within the text beginning at addr1+2 into digits, using the value of BASE , and accumulating each into +d1 after multiplying +d1 by the value of BASE . Conversion continues until an unconvertible character is encounter. addr2 is the location of the first unconvertible character.

COUNT addr1 -- addr2 +n 79

addr2 is addr1+1 and +n is the length of the counted string at addr1. The byte at addr1 contains the byte count +n. Range of +n is {0.255} See: "string, counted"

CR -- M,79 "c-r"

Displays a carriage-return and line-feed or equivalent operation.

CREATE -- M,79

A defining word executed in the form:

CREATE <name>

Creates a dictionary entry for <name>. After <name> is created, the next available dictionary location is the first byte of <name>'s parameter field. When <name> is subsequently executed, the address of the first byte of <name>'s parameter field is left on the stack. CREATE does not allocate space in <name>'s parameter field.

D+ wd1 wd2 -- wd3 79 "d-plus"

wd3 is the arithmetic sum of wd1 plus wd2.

D< d1 d2 -- flag 83 "d-less-than"

flag is true if d1 is less than d2 according to the operation of < except extended to 32 bits.

```
DECIMAL      --                            79
     Set the input-output numeric conversion base to ten.

DEFINITIONS  --                            79
     The compilation vocabulary is changed to be the same as the
     first vocabulary in the search order.  See:  "vocabulary,
     compilation"

DEPTH        -- +n                         79
     +n is the number of 16-bit values contained in the data
     stack before +n was placed on the stack.

DNEGATE      d1 -- d2                      79          "d-negate"
     d2 is the two's complement of d1.

DO           w1 w2 --                      C,I,83
             -- sys   (compiling)
     Used in the form:
             DO ... LOOP
     or
             DO ... +LOOP
     Begins a loop which terminates based on control parameters.
     The loop index begins at w2, and terminates based on the
     limit w1.  See LOOP and +LOOP for details on how the loop is
     terminated.  The loop is always executed at least once.  For
     example: w DUP DO ... LOOP executes 65,536 times.  sys is
     balanced with its corresponding LOOP or +LOOP .  See:  "9.9
     Control Structures"

     An error condition exists if insufficient space is available
     for at least three nesting levels.

DOES>        -- addr                       C,I,83          "does"
             --   (compiling)
     Defines the execution-time action of a word created by a
     high-level defining word.  Used in the form:
             : <namex> ... <create> ... DOES> ... ;
     and then
             <namex> <name>
     where <create> is CREATE or any user defined word which
     executes CREATE .

     Marks the termination of the defining part of the defining
     word <namex> and then begins the definition of the
     execution-time action for words that will later be defined
     by <namex>.  When <name> is later executed, the address of
     <name>'s parameter field is placed on the stack and then the
     sequence of words between DOES> and ; are executed.

DROP         16b --                        79
     16b is removed from the stack.

DUP          16b -- 16b 16b                79              "dupe"
     Duplicate 16b.
```

ELSE   --   C,I,79
sys1 -- sys2   (compiling)

Used in the form:
flag IF ... ELSE ... THEN

ELSE executes after the true part following IF . ELSE forces execution to continue at just after THEN . sys1 is balanced with its corresponding IF . sys2 is balanced with its corresponding THEN . See: IF THEN

EMIT   16b --   M,83

The least-significant 7-bit ASCII character is displayed. SEE: "9.5.3 EMIT"

EXECUTE   addr --   79

The word definition indicated by addr is executed. An error condition exists if addr is not a compilation address

EXIT   --   C,79

Compiled within a colon definition such that when executed, that colon definition returns control to the definition that passed control to it by returning control to the return point on the top of the return stack. An error condition exists if the top of the return stack does not contain a valid return point. May not be used within a do-loop. See: ; "stack, return" "9.3 Return Stack"

EXPECT   addr +n --   M,83

Receive characters and store each into memory. The transfer begins at addr proceeding towards higher addresses one byte per character until either a "return" is received or until +n characters have been transferred. No more than +n characters will be stored. The "return" is not stored into memory. No characters are received or transferred if +n is zero. All characters actually received and stored into memory will be displayed, with the "return" displaying as a space. See: SPAN "9.5.2 EXPECT"

FILL   addr u 8b --   83

u bytes of memory beginning at addr are set to 8b. No action is taken if u is zero.

FIND   addr1 -- addr2 n   83

addr1 is the address of a counted string. The string contains a word name to be located in the currently active search order. If the word is not found, addr2 is the string address addr1, and n is zero. If the word is found, addr2 is the compilation address and n is set to one of two non-zero values. If the word found has the immediate attribute, n is set to one. If the word is non-immediate, n is set to minus one (true).

FLUSH   --   M,83

Performs the function of SAVE-BUFFERS then unassigns all block buffers. (This may be useful for mounting or changing mass storage media).

FORGET   --   M,83

Used in the form:

```
        FORGET <name>
```

If <name> is found in the compilation vocabulary, delete <name> from the dictionary and all words added to the dictionary after <name> regardless of their vocabulary. Failure to find <name> is an error condition. An error condition also exists if the compilation vocabulary is deleted. See: "10.2 General Error Conditions"

FORTH   --   83

The name of the primary vocabulary. Execution replaces the first vocabulary in the search order with FORTH . FORTH is initially the compilation vocabulary and the first vocabulary in the search order. New definitions become part of the FORTH vocabulary until a different compilation vocabulary is established. See: VOCABULARY

FORTH-83   --   83

Assures that a FORTH-83 Standard System is available, otherwise an error condition exists.

HERE   -- addr   79

The address of the next available dictionary location.

HOLD   char --   79

char is inserted into a pictured numeric output string. Typically used between <# and #>.

I   -- w   C,79

w is a copy of the loop index. May only be used in the form:

```
        DO ... I ... LOOP
```

or

```
        DO ... I ... +LOOP
```

IF   flag --   C,I,79
            -- sys   (compiling)

Used in the form:

    flag IF ... ELSE ... THEN

or

    flag IF ... THEN

If flag is true, the words following IF are executed and the words following ELSE until just after THEN are skipped. The ELSE part is optional.

If flag is false, the words from IF through ELSE , or from IF through THEN (when no ELSE is used), are skipped. sys is balanced with its corresponding ELSE or THEN . See: "9.9 Control Structures"

IMMEDIATE   --   79

Marks the most recently created dictionary entry as a word which will be executed when encountered during compilation rather than compiled.

J   -- w   C,79

w is a copy of the index of the next outer loop. May only be used within a nested DO-LOOP or DO-+LOOP in the form, for example:

    DO ... DO ... J ... LOOP ... +LOOP

KEY   -- 16b   M,83

The least-significant 7 bits of 16b is the next ASCII character received. All valid ASCII characters can be received. Control characters are not processed by the system for any editing purpose. Characters received by KEY will not be displayed. See: "9.5.1 KEY"

LEAVE   --   C,I,83
            --   (compiling)

Transfers execution to just beyond the next LOOP or +LOOP . The loop is terminated and loop control parameters are discarded. May only be used in the form:

    DO ... LEAVE ... LOOP

or

    DO ... LEAVE ... +LOOP

LEAVE may appear within other control structures which are nested within the do-loop structure. More than one LEAVE may appear within a do-loop. See: "9.3 Return Stack"

LITERAL   -- 16b   C,I,79
            16b --   (compiling)

Typically used in the form:

    [16b] LITERAL

Compiles a system dependent operation so that when later executed, 16b will be left on the stack.

LOAD u -- M,79

The contents of >IN and BLK , which locate the current input stream, are saved. The input stream is then redirected to the beginning of screen u by setting >IN to zero and BLK to u. The screen is then interpreted. If interpretation from screen u is not terminated explicitly it will be terminated when the input stream is exhausted and then the contents of >IN and BLK will be restored. An error condition exists if u is zero. See: >IN BLK BLOCK

LOOP -- C,I,83
sys -- (compiling)

Increments the DO-LOOP index by one. If the new index was incremented across the boundary between limit-1 and limit the loop is terminated and loop control parameters are discarded. When the loop is not terminated, execution continues to just after the corresponding DO . sys is balanced with its corresponding DO . See: DO

MAX n1 n2 -- n3 79 "max"

n3 is the greater of n1 and n2 according to the operation of > .

MIN n1 n2 -- n3 79 "min"

n3 is the lesser of n1 and n2 according to the operation of < .

MOD n1 n2 -- n3 83

n3 is the remainder after dividing n1 by the divisor n2. n3 has the same sign as n2 or is zero. An error condition results if the divisor is zero or if the quotient falls outside of the range {-32,768..32,767}. See: "division, floored"

NEGATE n1 -- n2 79

n2 is the two's complement of n1, i.e, the difference of zero less n1.

NOT 16b1 -- 16b2 83

16b2 is the one's complement of 16b1.

OR 16b1 16b2 -- 16b3 79

16b3 is the bit-by-bit inclusive-or of 16b1 with 16b2.

OVER 16b1 16b2 -- 16b1 16b2 16b3 79

16b3 is a copy of 16b1.

PAD -- addr 83

The lower address of a scratch area used to hold data for intermediate processing. The address or contents of PAD may change and the data lost if the address of the next available dictionary location is changed. The minimum capacity of PAD is 84 characters.

PICK +n -- 16b 83

16b is a copy of the +nth stack value, not counting +n itself. {0..the number of elements on stack-1}

```
        0 PICK is equivalent to DUP
        1 PICK is equivalent to OVER
```

QUIT -- 79

Clears the return stack, sets interpret state, accepts new input from the current input device, and begins text interpretation. No message is displayed.

R> -- 16b C,79 "r-from"

16b is removed from the return stack and transferred to the data stack. See: "9.3 Return Stack"

R@ -- 16b C,79 "r-fetch"

16b is a copy of the top of the return stack.

REPEAT -- C,I,79
sys -- (compiling)

Used in the form:

```
        BEGIN ... flag WHILE ... REPEAT
```

At execution time, REPEAT continues execution to just after the corresponding BEGIN . sys is balanced with its corresponding WHILE . See: BEGIN

ROLL +n -- 83

The +nth stack value, not counting +n itself is first removed and then transferred to the top of the stack, moving the remaining values into the vacated position. {0..the number of elements on the stack-1}

```
        2 ROLL is equivalent to ROT
        0 ROLL is a null operation
```

ROT 16b1 16b2 16b3 -- 16b2 16b3 16b1 79 "rote"

The top three stack entries are rotated, bringing the deepest to the top.

SAVE-BUFFERS -- M,79 "save-buffers"

The contents of all block buffers marked as UPDATEed are written to their corresponding mass storage blocks. All buffers are marked as no longer being modified, but may remain assigned.

SIGN n -- 83

If n is negative, an ASCII "-" (minus sign) is appended to the pictured numeric output string. Typically used between <# and #> .

SPACE -- M,79

Displays an ASCII space.

SPACES +n -- M,79

Displays +n ASCII spaces. Nothing is displayed if +n is zero.

SPAN -- addr U,83

The address of a variable containing the count of characters actually received and stored by the last execution of EXPECT . See: EXPECT

STATE -- addr U,79

The address of a variable containing the compilation state. A non-zero content indicates compilation is occurring, but the value itself is system dependent. A Standard Program may not modify this variable.

SWAP 16b1 16b2 -- 16b2 16b1 79

The top two stack entries are exchanged.

THEN -- C,I,79
sys -- (compiling)

Used in the form:

```
        flag IF ... ELSE ... THEN
or
        flag IF ... THEN
```

THEN is the point where execution continues after ELSE , or IF when no ELSE is present. sys is balanced with its corresponding IF or ELSE . See: IF ELSE

TIB -- addr 83 "t-i-b"

The address of the text input buffer. This buffer is used to hold characters when the input stream is coming from the current input device. The minimum capacity of TIB is 80 characters.

TYPE addr +n -- M,79

+n characters are displayed from memory beginning with the character at addr and continuing through consecutive addresses. Nothing is displayed if +n is zero. See: "9.5.4 TYPE"

U. u -- M,79 "u-dot"

u is displayed as an unsigned number in a free-field format.

U< u1 u2 -- flag 83 "u-less-than"

flag is true if u1 is less than u2.

UM* u1 u2 -- ud 83 "u-m-times"

ud is the unsigned product of u1 times u2. All values and arithmetic are unsigned.

UM/MOD ud u1 -- u2 u3 83 "u-m-divide-mod"

u2 is the remainder and u3 is the floor of the quotient after dividing ud by the divisor u1. All values and arithmetic are unsigned. An error condition results if the divisor is zero or if the quotient lies outside the range {0..65,535}. See: "floor, arithmetic"

UNTIL flag -- C,I,79

sys -- (compiling)

Used in the form:

BEGIN ... flag UNTIL

Marks the end of a BEGIN-UNTIL loop which will terminate based on flag. If flag is true, the loop is terminated. If flag is false, execution continues to just after the corresponding BEGIN . sys is balanced with its corresponding BEGIN . See: BEGIN

UPDATE -- 79

The currently valid block buffer is marked as modified. Blocks marked as modified will subsequently be automatically transferred to mass storage should its memory buffer be needed for storage of a different block or upon execution of FLUSH or SAVE-BUFFERS .

VARIABLE -- M,79

A defining word executed in the form:

VARIABLE <name>

A dictionary entry for <name> is created and two bytes are ALLOTted in its parameter field. This parameter field is to be used for contents of the variable. The application is responsible for initializing the contents of the variable which it creates. When <name> is later executed, the address of its parameter field is placed on the stack.

VOCABULARY -- M,83

A defining word executed in the form:

VOCABULARY <name>

A dictionary entry for <name> is created which specifies a new ordered list of word definitions. Subsequent execution of <name> replaces the first vocabulary in the search order with <name>. When <name> becomes the compilation vocabulary new definitions will be appended to <name>'s list. See: DEFINITIONS "search order"

WHILE   flag --   C,I,79
               sys1 -- sys2   (compiling)

Used in the form:

BEGIN ... flag WHILE ... REPEAT

Selects conditional execution based on flag. When flag is true, execution continues to just after the WHILE through to the REPEAT which then continues execution back to just after the BEGIN . When flag is false, execution continues to just after the REPEAT , exiting the control structure. sys1 is balanced with its corresponding BEGIN . sys2 is balanced with its corresponding REPEAT . See: BEGIN

WORD   char -- addr   M,83

Generates a counted string by non-destructively accepting characters from the input stream until the delimiting character char is encountered or the input stream is exhausted. Leading delimiters are ignored. The entire character string is stored in memory beginning at addr as a sequence of bytes. The string is followed by a blank which is not included in the count. The first byte of the string is the number of characters {0..255}. If the string is longer than 255 characters, the count is unspecified. If the input stream is already exhausted as WORD is called, then a zero length character string will result.

If the delimiter is not found the value of >IN is the size of the input stream. If the delimiter is found >IN is adjusted to indicate the offset to the character following the delimiter. #TIB is unmodified.

The counted string returned by WORD may reside in the "free" dictionary area at HERE or above. Note that the text interpreter may also use this area. See: "input stream"

XOR   16b1 16b2 -- 16b3   79   "x-or"

16b3 is the bit-by-bit exclusive-or of 16b1 with 16b2.

[--   I,79   "left-bracket"
               --   (compiling)

Sets interpret state. The text from the input stream is subsequently interpreted. For typical usage see LITERAL . See:]

[']   -- addr   C,I,M,83   "bracket-
               --   (compiling)   tick"

Used in the form:

['] <name>

Compiles the compilation address addr of <name> as a literal. When the colon definition is later executed addr is left on the stack. An error condition exists if <name> is not found in the currently active search order. See: LITERAL

```
[COMPILE]    --                           C,I,M,79    "bracket-
             --   (compiling)                           compile"
```

Used in the form:

```
[COMPILE] <name>
```

Forces compilation of the following word <name>. This allows compilation of an immediate word when it would otherwise have been executed.

```
]            --                           79    "right-bracket"
```

Sets compilation state. The text from the input stream is subsequently compiled. For typical usage see LITERAL .
See: [

13. DOUBLE NUMBER EXTENSION WORD SET

13.1 The Double Number Extension Word Set Layers

Nucleus layer

```
     2!  2@  2DROP  2DUP  2OVER  2ROT  2SWAP  D+  D-  D0=  D2/
     D<  D=  DABS  DMAX  DMIN  DNEGATE  DU<
```

Device layer

     none

Interpreter layer

```
     D.  D.R
```

Compiler layer

```
     2CONSTANT  2VARIABLE
```

13. DOUBLE NUMBER EXTENSION WORD SET

13.2 The Double Number Extension Word Set Glossary

2!   32b addr --   79   "two-store"
32b is stored at addr. See: "number"

2@   addr -- 32b   79   "two-fetch"
32b is the value at addr. See: "number"

2CONSTANT   32b --   M,83   "two-constant"
A defining word executed in the form:
32b 2CONSTANT <name>
Creates a dictionary entry for <name> so that when <name> is later executed, 32b will be left on the stack.

2DROP   32b --   79   "two-drop"
32b is removed from the stack.

2DUP   32b -- 32b 32b   79   "two-dupe"
Duplicate 32b.

2OVER   32b1 32b2 -- 32b1 32b2 32b3   79   "two-over"
32b3 is a copy of 32b1.

2ROT   32b1 32b2 32b3 -- 32b2 32b3 32b1   79   "two-rote"
The top three double numbers on the stack are rotated, bringing the third double number number to the top of the stack.

2SWAP   32b1 32b2 -- 32b2 32b1   79   "two-swap"
The top two double numbers are exchanged.

2VARIABLE   --   M,79   "two-variable"
A defining word executed in the form:
2VARIABLE <name>
A dictionary entry for <name> is created and four bytes are ALLOTted in its parameter field. This parameter field is to be used for contents of the variable. The application is responsible for initializing the contents of the variable which it creates. When <name> is later executed, the address of its parameter field is placed on the stack. See: VARIABLE

D+   wd1 wd2 -- wd3   79
See the complete definition in the Required Word Set.

D-   wd1 wd2 -- wd3   79   "d-minus"
wd3 is the result of subtracting wd2 from wd1.

D.   d --   M,79   "d-dot"
The absolute value of d is displayed in a free field format. A leading negative sign is displayed if d is negative.

13. DOUBLE NUMBER EXTENSION WORD SET

D.R d +n -- M,83 "d-dot-r"

d is converted using the value of BASE and then displayed right aligned in a field +n characters wide. A leading minus sign is displayed if d is negative. If the number of characters required to display d is greater than +n, an error condition exists. See: "number conversion"

D0= wd -- flag 83 "d-zero-equals"

flag is true if wd is zero.

D2/ d1 -- d2 83 "d-two-divide"

d2 is the result of d1 arithmetically shifted right one bit. The sign is included in the shift and remains unchanged.

D< d1 d2 -- flag 83

See the complete definition in the Required Word Set.

D= wd1 wd2 -- flag 83 "d-equal"

flag is true if wd1 equals wd2.

DABS d -- ud 79 "d-absolute"

ud is the absolute value of d. If d is -2,147,483,648 then ud is the same value. See: "arithmetic, two's complement"

DMAX d1 d2 -- d3 79 "d-max"

d3 is the greater of d1 and d2.

DMIN d1 d2 -- d3 79 "d-min"

d3 is the lesser of d1 and d2.

DNEGATE d1 -- d2 79

See the complete definition in the Required Word Set.

DU< ud1 ud2 -- flag 83 "d-u-less"

flag is true if ud1 is less than ud2. Both numbers are unsigned.

14. ASSEMBLER EXTENSION WORD SET

14.1 The Assembler Extension Word Set Layers

Nucleus layer

none

Device layer

none

Interpreter layer

ASSEMBLER

Compiler layer

;CODE CODE END-CODE

14.2 Assembler Extension Word Set Usage

Because of the system dependent nature of machine language programming, a Standard Program cannot use CODE or ;CODE .

14.3 The Assembler Extension Word Set Glossary

```
;CODE        --                           C,I,79   "semi-colon-
             sys1 -- sys2                 (compiling)     code"
     Used in the form:
             : <namex> ... <create> ... ;CODE ... END-CODE
     Stops compilation, terminates the defining word <namex> and
     executes ASSEMBLER.  When <namex> is executed in the form:
             <namex> <name>
     to define the new <name>, the execution address of <name>
     will contain the address of the code sequence following the
     ;CODE in <namex>.  Execution of any <name> will cause this
     machine code sequence to be executed.  sys1 is balanced with
     its corresponding : .  sys2 is balanced with its
     corresponding END-CODE .  See:  CODE  DOES>

ASSEMBLER    --                           83
     Execution replaces the first vocabulary in the search order
     with the ASSEMBLER vocabulary.  See:  VOCABULARY

CODE         -- sys                       M,83
     A defining word executed in the form:
             CODE <name> ... END-CODE
     Creates a dictionary entry for <name> to be defined by a
     following sequence of assembly language words.  Words thus
     defined are called code definitions.  This newly created
     word definition for <name> cannot be found in the dictionary
     until the corresponding END-CODE is successfully processed
     (see:  END-CODE ).  Executes ASSEMBLER .  sys is balanced
     with its corresponding END-CODE .

END-CODE     sys --                       79         "end-code"
     Terminates a code definition and allows the <name> of the
     corresponding code definition to be found in the dictionary.
     sys is balanced with its corresponding CODE or ;CODE .  See:
     CODE
```

15. THE SYSTEM EXTENSION WORD SET

15.1 The System Extension Word Set Layers

Nucleus layer

BRANCH ?BRANCH

Device layer

none

Interpreter layer

CONTEXT CURRENT

Compiler layer

<MARK <RESOLVE >MARK >RESOLVE

15.2 System Extension Word Set Usage

After BRANCH or ?BRANCH is compiled, >MARK or <RESOLVE is executed. The addr left by >MARK is passed to >RESOLVE . The addr left by <MARK is passed to <RESOLVE . For example:

```
     : IF     COMPILE ?BRANCH  >MARK  ; IMMEDIATE
     : THEN   >RESOLVE  ; IMMEDIATE
```

15.3 The System Extension Word Set Glossary

<MARK -- addr C,83 "backward-mark"

Used at the destination of a backward branch. addr is typically only used by <RESOLVE to compile a branch address.

<RESOLVE addr -- C,83"backward-resolve"

Used at the source of a backward branch after either BRANCH or ?BRANCH . Compiles a branch address using addr as the destination address.

>MARK -- addr C,83 "forward-mark"

Used at the source of a forward branch. Typically used after either BRANCH or ?BRANCH . Compiles space in the dictionary for a branch address which will later be resolved by >RESOLVE .

>RESOLVE addr -- C,83"forward-resolve"

Used at the destination of a forward branch. Calculates the branch address (to the current location in the dictionary) using addr and places this branch address into the space left by >MARK .

?BRANCH flag -- C,83"question-branch"

When used in the form: COMPILE ?BRANCH a conditional branch operation is compiled. See BRANCH for further details. When executed, if flag is false the branch is performed as with BRANCH . When flag is true execution continues at the compilation address immediately following the branch address.

BRANCH -- C,83

When used in the form: COMPILE BRANCH an unconditional branch operation is compiled. A branch address must be compiled immediately following this compilation address. The branch address is typically generated by following BRANCH with <RESOLVE or >MARK .

CONTEXT -- addr U,79

The address of a variable which determines the dictionary search order.

CURRENT -- addr U,79

The address of a variable specifying the vocabulary in which new word definitions are appended.

16. CONTROLLED REFERENCE WORDS

The Controlled Reference Words are word definitions which, although not required, cannot be present with a non-standard definition in the vocabulary FORTH of a Standard System. These words have present usage and/or are candidates for future standardization.

-->   --   I,M,79   "next-block"
             --   (compilation)

Continue interpretation on the next sequential block. May be used within a colon definition that crosses a block boundary.

.R   n +n --   M,83   "dot-r"

n is converted using BASE and then displayed right aligned in a field +n characters wide. A leading minus sign is displayed if n is negative. If the number of characters required to display n is greater than +n, an error condition exists. See: "number conversion"

2*   w1 -- w2   83   "two-times"

w2 is the result of shifting w1 left one bit. A zero is shifted into the vacated bit position.

BL   -- 32   79   "b-l"

Leave the ASCII character value for space (decimal 32).

BLANK   addr u --   83

u bytes of memory beginning at addr are set to the ASCII character value for space. No action is taken if u is zero.

C,   16b --   83   "c-comma"

ALLOT one byte then store the least-significant 8 bits of 16b at HERE 1- .

DUMP   addr u --   M,79

List the contents of u addresses starting at addr. Each line of values may be preceded by the address of the first value.

EDITOR   --   83

Execution replaces the first vocabulary in the search order with the EDITOR vocabulary. See: VOCABULARY

EMPTY-BUFFERS --   M,79 "empty-buffers"

Unassign all block buffers. UPDATEed blocks are not written to mass storage. See: BLOCK

16. CONTROLLED REFERENCE WORDS

END   flag --   C,I,79
             sys --   (compiling)

A synonym for UNTIL .

ERASE   addr u --   79

u bytes of memory beginning at addr are set to zero. No action is taken if u is zero.

HEX   --   29

Set the numeric input-output conversion base to sixteen.

INTERPRET   --   M,83

Begin text interpretation at the character indexed by the contents of >IN relative to the block number contained in BLK , continuing until the input stream is exhausted. If BLK contains zero, interpret characters from the text input buffer. See: "input stream"

K   -- w   C,83

w is a copy of the index of the second outer loop. May only be used within a nested DO-LOOP or DO-+LOOP in the form, for example:

```
        DO ... DO ... DO ... K ... LOOP ... +LOOP ... LOOP
```

LIST   u --   M,79

The contents of screen u are displayed. SCR is set to u. See: BLOCK

OCTAL   --   83

Set the numeric input-output conversion base to eight.

OFFSET   -- addr   U,83

The address of a variable that contains the offset added to the block number on the stack by BLOCK or BUFFER to determine the actual physical block number.

QUERY   --   M,83

Characters are received and transferred into the memory area addressed by TIB . The transfer terminates when either a "return" is received or the number of characters transferred reaches the size of the area addressed by TIB . The values of >IN and BLK are set to zero and the value of #TIB is set to the value of SPAN . WORD may be used to accept text from this buffer. See: EXPECT "input stream"

RECURSE   --   C,I,83
             --   (compiling)

Compile the compilation address of the definition being compiled to cause the definition to later be executed recursively.

SCR   -- addr   U,79   "s-c-r"

The address of a variable containing the number of the screen most recently LISTed.

SP@ -- addr 79 "s-p-fetch"
addr is the address of the top of the stack just before SP@ was executed.

THRU u1 u2 -- M,83
Load consecutively the blocks from u1 through u2.

U.R u +n -- M,83 "u-dot-r"
u is converted using the value of BASE and then displayed as an unsigned number right aligned in a field +n characters wide. If the number of characters required to display u is greater than +n, an error condition exists. See: "number conversion"

APPENDIX A. STANDARDS TEAM MEMBERSHIP

A.1 Standard Team Membership: Members

The following is a list in alphabetical order of the people who are FORTH Standards Team Members. These names are provided to indicate the texture and make-up of the team itself. Where appropriate, the official capacity of individuals is also indicated.

Paul Bartholdi, Sauverny, Switzerland	
Robert Berkey, Palo Alto, California USA	Treasurer
David Boulton, Redwood City, California USA	
John Bumgarner, Morgan Hill, California USA	
Don Colburn, Rockville, Maryland USA	
James T. Currie, Jr., Blacksburg, Virginia USA	
Thomas B. Dowling, Lowell, Massachusetts USA	
William S. Emery, Malibu, California USA	
Lawrence P. Forsley, Rochester, New York USA	
Kim R. Harris, Palo Alto, California USA	Referee
John S. James, Los Gatos, California USA	
Guy M. Kelly, La Jolla, California USA	Chair
Thea Martin, Rochester, New York USA	
Michael McNeil, Scotts Valley, California USA	
Robert E. Patten, Modesto, California USA	
Michael Perry, Berkeley, California USA	
David C. Petty, Cambridge, Massachusetts USA	
William F. Ragsdale, Hayward, California USA	
Elizabeth D. Rather, Hermosa Beach, California USA	
Dean Sanderson, Hermosa Beach, California USA	Referee
Klaus Schleisiek, Hamburg, W-Germany	
George W. Shaw II, Hayward, California USA	Referee
Robert L. Smith, Palo Alto, California USA	Secretary
Michael K. Starling, Elkview, West Virginia USA	
John K. Stevenson, Portland, Oregon USA	
Glenn S. Tenney, San Mateo, California USA	Referee

A. STANDARDS TEAM MEMBERSHIP

A.2 FORTH Standards Team Sponsors

The following is a list in alphabetical order of individuals and organizations who have contributed funds and other assistance to aid the word of the FST and deserve recognition for their involvement. FST sponsors have no duties or responsibilities in the FST, but they receive copies of proposals and comments considered at a formal meeting, and drafts and adopted standards prepared as a result of that meeting.

Creative Solutions Inc., 4801 Randolph Rd., Rockville, MD 20852 USA

Fantasia Systems Inc., 1059 Alameda de las Pulgas, Belmont, CA 94002 USA

FORTH, Inc., 2309 Pacific Coast Highway, Hermosa Beach, CA 90254 USA

FORTH Interest Group Inc., P.O. Box 1105, San Carlos, CA 94070 USA

Forthright Enterprises, P.O. Box 50911, Palo Alto, CA 94020 USA

Glen Haydon Enterprises, Box 439 Rt. 2, La Honda, CA 94020 USA

John K. Gotwals, W. Lafayette, IN USA

John D. Hall, Oakland, CA USA

Hartronix, Inc., 1201 N. Stadem, Tempe, AZ 85281 USA

Hewlett-Packard Corvallis Div., 1000 NE Circle Blvd., Corvallis, OR 97330 USA

Information Unlimited Software, Inc., 2401 Marinship, Sausalito, CA 94965 USA

Henry H. Laxen, 1259 Cornell Avenue, Berkeley, CA 94705 USA

Laxen & Harris, Inc.

George B. Lyons, 280 Henderson Street, Jersey Cit, NJ 07302 USA

C. Kevin McCabe, Chicago, IL USA

MicroMotion, 12077 Wilshire Blvd #506, Los Angeles, CA 90025 USA

Bruce R. Montague, Monterey, CA USA

Mountain View Press, P.O. Box 4659, Mountain View, CA 94040 USA

A. STANDARDS TEAM MEMBERSHIP

Michael A. Perry, Berkeley, CA USA

Robert Berkey Services, 2334 Dumbarton Ave., Palo Alto, CA 94303
USA

Royal Greenwich Observatory, Herstmonsioux Castle, Eastbourne,
England

Shaw Laboratories, Ltd., 24301 Southland Drive #216, Hayward, CA
94545 USA

Sygnetron Protection Systems, Inc., 2103 Greenspring, Timonium,
MD 21093 USA

Telelogic Inc., 196 Broadway, Cambridge, MA 02139 USA

UNISOFT, P.O. Box 2644, New Carrollton, MD 20784 USA

APPENDIX B. UNCONTROLLED REFERENCE WORDS

The Uncontrolled Reference Word Set contains glossary definitions which are included for public reference of words that have past or present usage and/or are candidates for future standardization. No recommendation is made that these words be included in a system.

No restrictions are placed on the definition or usage of uncontrolled words. However, use of these names for procedures differing from the given definitions is discouraged.

```
!BITS        16b1 addr 16b2 --                     "store-bits"
     Store the value of 16b1 masked by 16b2 into the equivalent
     masked part of the contents of addr, without affecting bits
     outside the mask.

**           n1 n2 -- n3                                "power"
     n3 is the value of n1 to the power n2.

+BLOCK       w -- u                                "plus-block"
     u is the sum of w plus the number of the block being
     interpreted.

-'           -- addr false                          "dash-tick"
             -- true
     Used in the form:
             -' <name>
     Leave the parameter field of <name> beneath zero (false) if
     <name> can be found in the search order; leave only true if
     not found.

-MATCH       addr1 +n1 addr2 +n2 -- addr3 flag      "dash-match"
     Attempt to find the +n2-length text string beginning at
     addr2 somewhere in the +n1-length text string beginning at
     addr1.  Return the last+1 address addr3 of the match point
     and a flag which is zero if a match exists.

-TEXT        addr1 +n1 addr2 -- n2                  "dash-text"
     Compare two strings over the length +n1 beginning at addr1
     and addr2.  Return zero if the strings are equal.  If
     unequal, return n2, the difference between the last
     characters compared:  addr1(i) - addr2(i).
```

B. UNCONTROLLED REFERENCE WORDS

/LOOP   +n --   C,I   "up-loop"
sys -- (compiling)

A do-loop terminating word. The loop index is incremented by the positive value +n. If the unsigned magnitude of the resultant index is greater than the limit, then the loop is terminated, otherwise execution returns to the corresponding DO . The comparison is unsigned magnitude. sys is balanced with its corresponding DO . See: DO

1+!   addr --   "one-plus-store"

Add one to the 16-bit contents at addr.

1-!   addr --   "one-minus-store"

Subtract one from the 16-bit contents at addr.

;:   -- addr   C,I"semi-colon-colon"

Used to specify a new defining word:

: <namex> <name>

When <namex> is executed, it creates an entry for the new word <name>. Later execution of <name> will execute the sequence of words between ;: and ; , with the address of the first (if any) parameters associated with <name> on the stack.

;S   --   Interpret only"semi-s"

Stop interpretation of a block.

<>   w1 w2 -- flag   "not-equal"

flag is true if w1 is not equal to w2.

<BUILDS   --   "builds"

Used in conjunction with DOES> in defining words, in the form:

: <namex> ... <BUILDS ... DOES> ... ;

and then:

<namex> <name>

When <namex> executes, <BUILDS creates a dictionary entry for the new <name>. The sequence of words between <BUILDS and DOES> established a parameter field for <name>. When <name> is later executed, the sequence of words following DOES> will be executed, with the parameter field address of <name> on the data stack.

<CMOVE   addr1 addr2 u --   "reverse-c-move"

A synonym for CMOVE> .

><   16b1 -- 16b2   "byte-swap"

Swap the high and low bytes within 16b1.

>MOVE<   addr1 addr2 u --   "byte-swap-move"

Move u bytes beginning at addr1 to the memory beginning at addr2. During this move, the order of each byte pair is reversed.

B. UNCONTROLLED REFERENCE WORDS

@BITS addr 16b1 -- 16b2 "fetch-bits"
Return the 16-bits at addr masked by 16b1.

AGAIN -- C,I
sys -- (compiling)
Effect an unconditional jump back to the start of a BEGIN-AGAIN loop. sys is balanced with its corresponding BEGIN . See: BEGIN

ASCII -- char I,M "as-key"
-- (compiling)
Used in the form:

```
        ASCII ccc
```

where the delimiter of ccc is a space. char is the ASCII character value of the first character in ccc. If interpreting, char is left on the stack. If compiling, compile char as a literal so that when the colon definition is later executed, char is left on the stack.

ASHIFT 16b1 n -- 16b2 "a-shift"
Shift the value 16b1 arithmetically n bits left if n is positive, shifting zeros into the least significant bit positions. If n is negative, 16b1 is shifted right; the sign is included in the shift and remains unchanged.

B/BUF -- 1024 "bytes-per-buffer"
A constant leaving 1024, the number of bytes per block buffer.

BELL --
Activate a terminal bell or noise-maker as appropriate to the device in use.

CHAIN -- M
Used in the form:

```
        CHAIN <name>
```

Connect the CURRENT vocabulary to all definitions that might be entered into the vocabulary <name> in the future. The CURRENT vocabulary may not be FORTH or ASSEMBLER . Any given vocabulary may only be chained once, but may be the object of any number of chainings. For example, every user-defined vocabulary may include the sequence:

```
        CHAIN FORTH
```

CONTINUED u -- M
Continue interpretation at block u.

CUR -- addr
A variable pointing to the physical record number before which the tape is currently positioned. REWIND sets CUR=1.

DBLOCK ud -- addr M "d-block"
Identical to BLOCK but with a 32-bit block unsigned number.

B. UNCONTROLLED REFERENCE WORDS

DPL   -- addr   U   "d-p-l"
A variable containing the number of places after the fractional point for input conversion.

FLD   -- addr   U   "f-l-d"
A variable pointing to the field length reserved for a number during output conversion.

H.   u --   M   "h-dot"
Output u as a hexadecimal integer with one trailing blank. The current base is unchanged.

I'   -- w   C   "i-prime"
Used within a colon definition executed only from within a do-loop to return the corresponding loop index.

IFEND   Interpret only"if-end"
Terminate a conditional interpretation sequence begun by IFTRUE .

IFTRUE   flag --   Interpret only "if-true"
Begin an:

        IFTRUE ... OTHERWISE ... IFEND

conditional sequence. These conditional words operated like:

        IF ... ELSE ... THEN

except that they cannot be nested, and are to be used only during interpretation. In conjunction with the words [and] the words [and] they may be used within a colon definition to control compilation, although they are not to be compiled.

INDEX   u1 u2 --   M
Print the first line of each screen over the range {u1..u2}. This displays the first line of each screen of source text, which conventionally contains a title.

LAST   -- addr   U
A variable containing the address of the beginning of the last dictionary entry made, which may not yet be a complete or valid entry.

LINE   +n -- addr   M
addr is the address of the beginning of line +n for the screen whose number is contained in SCR . The range of +n is {0..15}.

LINELOAD   +n u --   "line-load"
Begin interpretation at line +n of screen u.

B. UNCONTROLLED REFERENCE WORDS

LOADS u -- M

A defining word executed in the form:

u LOADS <name>

When <name> is subsequently executed, block u will be loaded.

MAP0 -- addr "map-zero"

A variable pointing to the first location in the tape map.

MASK n -- 16b

16b is a mask of n most-significant bits if n is positive, or n least-significant bits if n is negative.

MOVE addr1 addr2 u --

The u bytes at address addr1 are moved to address addr2. The data are moved such that the u bytes remaining at address addr2 are the same data as was originally at address addr1. If u is zero nothing is moved.

MS +n -- M "m-s"

Delay for approximately +n milliseconds.

NAND 16b1 16b2 -- 16b3

16b3 is the one's complement of the logical AND of 16b1 with 16b2.

NOR 16b1 16b2 -- 16b3

16b3 is the one's complement of the logical OR of 16b1 with 16b2.

NUMBER addr -- d

Convert the count and character string at addr, to a signed 32-bit integer, using the value of BASE . If numeric conversion is not possible, an error condition exists. The string may contain a preceding minus sign.

O. u -- M "o-dot"

Print u in octal format with one trailing blank. The value in BASE is unaffected.

OTHERWISE -- Interpret only

An interpreter-level conditional word. See: IFTRUE

PAGE -- M

Clear the terminal screen or perform a form-feed action suitable to the output device currently active.

READ-MAP -- M "read-map"

Read to the next file mark on tape constructing a correspondence table in memory (the map) relating physical block position to logical block number. The tape should normally be rewound to its load point before executing READ-MAP .

B. UNCONTROLLED REFERENCE WORDS

REMEMBER   --   M

A defining word executed in the form:

        REMEMBER <name>

Defines a word which, when executed, will cause <name> and all subsequently defined words to be deleted from the dictionary. <name> may be compiled into and executed from a colon definition. The sequence

        DISCARD REMEMBER DISCARD

provides a standardized preface to any group of transient word definitions.

REWIND   --   M

Rewind the tape to its load point, setting CUR equal to one.

ROTATE   16b1 n -- 16b2

Rotate 16b1 left n bits if n is positive, right n bits if n is negative. Bits shifted out of one end of the cell are shifted back in at the opposite end.

S0   -- addr   U   "s-zero"

A variable containing the address of the bottom of the stack.

SET   16b addr --   M

A defining word executed in the form:

        16b addr SET <name>

Defines a word <name> which, when executed, will cause the value 16b to be stored at addr.

SHIFT   16b1 n -- 16b2

Logical shift 16b1 left n bits if n is positive, right n bits if n is negative. Zeros are shifted into vacated bit positions.

TEXT   char --   M

Accept characters from the input stream, as for WORD , into PAD , blank-filling the remainder of PAD to 84 characters.

USER   +n --   M

A defining word executed in the form:

        +n USER <name>

which creates a user variable <name>. +n is the offset within the user area where the value for <name> is stored. Execution of <name> leaves its absolute user area storage address.

WORDS   --   M

List the word names in the first vocabulary of the currently active search order.

B. UNCONTROLLED REFERENCE WORDS

```
\LOOP        +n --                        C,I       "down-loop"
             sys --   (compiling)
```

A do-loop terminating word. The loop index is decremented by the positive value +n. If the unsigned magnitude of the resultant index is less than or equal to the limit, then the loop is terminated, otherwise execution returns to the corresponding DO . The comparison is unsigned. sys is balanced with its corresponding DO . See: DO

APPENDIX C. EXPERIMENTAL PROPOSALS

Since FORTH is an extensible language and subject to evolution, the Standard contains a section describing experimental proposals. FORTH users are encouraged to study, implement, and try these proposals to aid in the analysis of and the decision for or against future adoption into the Standard. Readers are cautioned that these proposals contain opinions and conclusions of the authors of the proposals and that these proposals may contain non-standard source code.

C. EXPERIMENTAL PROPOSALS

SEARCH ORDER SPECIFICATION AND CONTROL

WILLIAM F. RAGSDALE

1 INTRODUCTION

The method of selecting the order in which the dictionary is searched has grown from unchained vocabularies to the present use of chained vocabularies. Many techniques are in use for specification of the sequence in which multiple vocabularies may be searched. In order to offer generality and yet get precision in specification, this proposal is offered.

2 DESCRIPTION

The following functions are required:

1. Two search orders exist. CONTEXT is the group of vocabularies searched during interpretation of text from the input stream. CURRENT is the single vocabulary into which new definitions are compiled, and from which FORGET operates.

2. Empty CONTEXT to a minimum number of system words. These are just the words to further specify the search order.

3. Add individual vocabularies into CONTEXT. The most recently added is searched first.

4. Specify which single vocabulary will become CURRENT.

The following optional functions aid the user:

1. Display the word names of the first vocabulary in the CONTEXT search order.

2. Display the vocabulary names comprising CURRENT and CONTEXT search orders.

C. EXPERIMENTAL PROPOSALS

3 ADVANTAGES

Use over the past year has demonstrated that the proposed methods may emulate the vocabulary selection of all other systems. The order is explicit by execution, may be interpreted and compiled, and is obvious from the declaration. The search order is specified at run-time rather than the time a new vocabulary is created.

4 DISADVANTAGES

By migrating to a common structure, vendors give up one point at which they may claim their product is better than others. Another drawback is that the number of CONTEXT vocabularies is fixed; older methods had an indefinite 'tree' structure. In practice, the branching of such a structure was very rarely greater than four.

Forth words operate in a context sensitive environment, as word names may be redefined and have different definitions in different vocabularies. This proposal compounds the problem. By displaying the search order names, the user at least can readily verify the search order.

5 IMPACT

The text of the Forth 83 Standard has been carefully chosen for consistency and generality. However, no specification on how the search order is developed by the user is given. This omission is unavoidable, due to the diversity of contemporary practice. This proposal is intended to complete the Forth 83 requirements in a fashion that exceeds all other methods.

Previously standardized words continue in their use: VOCABULARY, FORTH, DEFINITIONS, and FORGET. However, this proposal assumes that vocabulary names are not IMMEDIATE .

6 DEFINITIONS

Search order:
The sequence in which vocabularies are selected when locating a word by name in the dictionary. Consists of one transient and up to three resident vocabularies.

Transient order:
Execution of any vocabulary makes it the first vocabulary searched, replacing the previously selected transient vocabulary.

C. EXPERIMENTAL PROPOSALS

Resident order:

After searching the transient order, up to three additional vocabularies may be searched. The application program controls this selection.

7 GLOSSARY

ONLY -- ONLY

Select just the ONLY vocabulary as both the transient vocabulary and resident vocabulary in the search order.

FORTH -- ONLY

The name of the primary vocabulary. Execution makes FORTH the transient vocabulary, the first in the search order, and thus replaces the previous transient vocabulary.

ALSO -- ONLY

The transient vocabulary becomes the first vocabulary in the resident portion of the search order. Up to the last two resident vocabularies will also be reserved, in order, forming the resident search order.

ORDER -- ONLY

Display the vocabulary names forming the search order in their present search order sequence. Then show the vocabulary into which new definitions will be placed.

WORDS -- ONLY

Display the word names in the transient vocabulary, starting with the most recent definition.

FORGET -- ONLY

Used in the form:

FORGET <name>

Delete from the dictionary <name> and all words added to the dictionary after <name> regardless of the vocabulary. Failure to find <name> is an error condition. An error condition also exists upon implicitly forgetting a vocabulary (due to its definition after <name>).

DEFINITIONS -- ONLY

Select the transient vocabulary as the current vocabulary into which subsequent definitions will be added.

SEAL -- ONLY

Delete all occurances of ONLY from the search order. The effect is that only specified application vocabularies will be searched.

C. EXPERIMENTAL PROPOSALS

8 TYPICAL SOURCE CODE

```
( ALSO  ONLY                                     82jun12 WFR )
( note the systems -FIND searches 1 to 5 vocabs in CONTEXT   )
VOCABULARY ONLY    ONLY DEFINITIONS
: ALSO                       ( slide transient into resident )
      CONTEXT DUP 2+ 6 CMOVE>  ;

   HERE 2+ ]          ( alter run time from usual vocabulary )
       DOES>  CONTEXT 8 ERASE DUP CONTEXT !    CONTEXT 8 + !
              ALSO  EXIT [
       ' ONLY CFA !        ( Patch into ONLY; make NULL word )
CREATE X  ' EXIT >BODY X !     41088 ' X NFA ! IMMEDIATE
: FORTH      FORTH ;
: DEFINITIONS DEFINITIONS ;         : FORGET  FORGET ;
: VOCABULARY  VOCABULARY  ;         : ONLY      ONLY ;
: WORDS       WORDS  ;

```

```
( ORDER                                         82jun12 WFR )
: ORDER     ( show the search order )
   10 SPACES  CONTEXT 10 OVER + SWAP
      DO I @ ?DUP 0= ?LEAVE  ID. 2 +LOOP
 10 SPACES  CURRENT @ ID.  ;

ONLY FORTH ALSO   DEFINITIONS









```

9 EXAMPLES OF USE

ONLY	reduce search order to minimum
FORTH	search FORTH then ONLY
ALSO EDITOR	search EDITOR, FORTH then ONLY
DEFINITIONS	new definitions will be added into the EDITOR

The same sequence would be compiled:

```
: SETUP  ONLY FORTH  ALSO EDITOR DEFINITIONS ;
```

10 REFERENCES

W. F. Ragsdale, The 'ONLY' Concept for Vocabularies, Proceedings of the 1982 FORML Conference, pub. Forth Interest Group.

C. EXPERIMENTAL PROPOSALS

W. F. Ragsdale, fig-FORTH Installation Manual, Forth Interest Group.

C. EXPERIMENTAL PROPOSALS

DEFINITION FIELD ADDRESS CONVERSION OPERATORS

by

Kim R. Harris

A. INTRODUCTION

The standard provides a transportable way to obtain the compilation address of a definition in the dictionary of a FORTH system (cf., FIND and '). It also provides an operator to convert a compilation address to its corresponding parameter field address. However, the standard does not provide a transportable way to convert either of these addresses to the other fields of a definition. Since various FORTH implementations have different dictionary structures, a standard set of conversion operators would increase transportability and readability.

A set of words is proposed which allows the conversion of any definitions field address to any other.

B. GLOSSARY

In the following words, the compilation address is either the source or the destination, so it is not indicated in the names.

>BODY addr1 -- addr2 "to-body"
addr2 is the parameter field address corresponding to the compilation address addr1.

>NAME addr1 -- addr2 "to-name"
addr2 is the name field address corresponding to the compilation address addr1.

>LINK addr1 -- addr2 "to-link"
addr2 is the link field address corresponding to the compilation address addr1.

BODY> addr1 -- addr2 "from-body"
addr2 is the compilation address corresponding to the parameter field address addr1.

NAME> addr1 -- addr2 "from-name"
addr2 is the compilation address corresponding to the name field address addr1.

C. EXPERIMENTAL PROPOSALS

LINK> addr1 -- addr2 "from-link"
addr2 is the compilation address corresponding to the link field address addr1.

The previous set of words is complete, but may be inefficient for going between two fields when one is not the compilation address. For greater efficiency, additional operators may be defined which name both the source and destination fields.

N>LINK addr1 -- addr2 "name-to-link"
addr2 is the link field address corresponding to the name field address addr1.

L>NAME addr1 -- addr2 "link-to-name"
addr2 is the name field address corresponding to the link field address addr1.

C. DISCUSSION

The previous words provide a complete, consistent, and efficient set of definition field address conversion operations. They can be implemented in a FORTH system which uses any combination of the following options for its dictionary structure:

Link fields first or second.
Fixed or variable length name fields.
Additional fields in the definitions structure.

Heads contiguous or separated from bodies.

Indirect, direct, subroutine, or token threaded code.

The words are compatible with this standard; their inclusion would not require other changes to be made to the standard.

Disadvantages to including them in the standard include:

They add 6 to 8 more words to the standard.

A standard program may not use all of them since it is not allowed to access the name or link fields. However, this does not disqualify them from being in the standard.

If a definition's head is not in the dictionary, an error condition would exist. In this case, what action should the words take in an implemented system?

The author of this experimental proposal recommends that FORTH system implementors try them and that they be included in the System Word Set of the next FORTH standard.

C. EXPERIMENTAL PROPOSALS

D. SOURCE CODE EXAMPLE

High level source code is shown below for a very simple dictionary structure. This code assumes a FORTH system which uses indirect threaded code, heads contiguous to bodies, and a definition structure of the following format:

```
     Name field, 4 bytes long, fixed length.
     Link field, 2 bytes long.
     Code field, 2 bytes long.
     Parameter field, variable length.
```

```
: >BODY   ( acf -- apf )  2+  ;
: BODY>   ( apf -- acf )  2-  ;
: >LINK   ( acf -- alf )  2-  ;
: LINK>   ( alf -- acf )  2-  ;
: >NAME   ( acf -- anf )  6 - ;
: NAME>   ( anf -- alf )  6 + ;
: N>LINK  ( anf -- alf )  4 + ;
: L>NAME  ( alf -- anf )  4 - ;
```

E. EXAMPLES OF USE

No examples are given because their use should be obvious.

D. CHARTER

APPENDIX D.

CHARTER

# of the

FORTH STANDARDS TEAM

1. Purpose and Goals

1.1 Purpose

1.1.1 This Charter establishes and guides a voluntary membership professional organization, the FORTH Standards Team (hereafter referred to as the "FST") and provides a method for its operation.

1.2 Goals

1.2.1 The goal of the FST is the creation, maintenance, and proliferation of a standard (hereafter referred to as the "Standard") for the FORTH computer programming system and for application programs executed by a Standard system. The Standard shall specify requirements and constraints which such computer software must satisfy.

1.2.2 The team shall also develop a method of identification and labeling of FORTH implementations and programs which conform to the Standard.

1.3 Organization

1.3.1 The FST is a voluntary membership organization with no formal status as a legal entity. It operates by consensus of the professional and commercial FORTH community and conducts business by the professional discourse and agreement of its members. It is intended that this Charter be a guide to the operation of the FST subject to reasonable minor digression, rather than being a rigid document under which vested rights are granted.

D. CHARTER

2. METHODS

2.1 Formal Meetings

2.1.1 The FST shall hold periodic formal meetings for discussion and decisions concerning a current or future Standard.

2.1.2 There is not specified frequency for formal meetings. Each meeting shall be at such time and place as was decided at the prior meeting. If a meeting cannot be held as decided, the Chairperson may designate another time and place.

2.1.3 The Chairperson shall send a written notice at least sixty (60) days in advance of each formal meeting to each voting member. A longer notification period is recommended. It is anticipated that the continuing close coordination of the participants, the decision at the prior formal meeting, and publication of a meeting notice in FORTH Dimensions and other trade journals will provide sufficient notice to the FORTH community.

2.1.4 At a formal FST meeting, there shall be general sessions consisting of all attendees. General sessions are for matters that are ready for discussion and decision. All votes concerning the Standard, Charter, or FST procedures must take place during a general session.

2.1.5 Also at formal meetings, subteams will be established to examine groups of proposals and to prepare recommendations for a general session. All meeting attendees may participate in the work and voting of a subteam. Each subteam should elect from its members a coordinator to conduct its meetings and a reporter to record and report its recommendations.

2.1.6 The Chairperson may publish and distribute an agenda at or in advance of a formal meeting. As a guideline, each day of a formal meeting begins with a general session, followed by concurrent subteam meetings followed by another general session.

2.1.7 In view of the voluntary nature of the FST, at least one third of the membership is required to hold a formal meeting. Two thirds of the number of voting members present at the start of each day's first general session shall set the quorum for the remainder of that day.

2.1.8 Between formal meetings, the Chairperson may appoint such informal working groups as is appropriate. Each group may be given a goal and scope to direct its activities. Its conclusions or recommendations must be given to the Chairperson in written form.

2.2 Proposals and Comments

2.2.1 Prior to each formal meeting, the Chairperson may solicit submission of comments and proposals for changes, additions, or deletions to the then-current Standard, the draft Standard or this Charter. A cutoff date may be specified for the submission of such proposals.

2.2.2 A considerable amount of information must accompany each proposal to help FST members analyze the proposal. Therefore, submission of proposals and comments shall be according to the format and instructions shown in the "Proposal/Comment Form" included as an Appendix to this Standard. Any proposal not in the appropriate form or received after the cutoff date may not be considered unless the Chairperson deems it to be of sufficient significance.

2.2.3 Unsolicited proposals and comments by volunteers are acknowledged as valuable. Any individual or group may submit proposals and/or comments concerning the Standard or this Charter. These should be sent to the official address of the FST. Properly formatted proposals and comments are preferred. The author or a representative should plan to attend the next formal meeting to emphasize, support, and possibly modify the proposals.

2.2.4 Since the quantity of proposals and comments may exceed the number for which there is time to be voted upon, submission of a proposal does not automatically mean that it will be voted upon at the next formal FST meeting. The Chairperson or some members appointed by the Chairperson or elected by the voting members may screen and organize the received proposals and comments for voting upon at the next formal meeting.

2.2.5 To allow reflection and examination, proposals and comments shall be distributed to FST voting members and sponsors in advance of a formal meeting. Proposals and comments not distributed in advance, including proposals made during a formal meeting, may be considered at the discretion of the Chairperson.

D. CHARTER

2.3 Draft Standard

After a formal meeting, the referees and officers of the FST shall prepare a draft Standard for review by the then-current FST voting members. The referees and officers shall consolidate proposals accepted by vote during the meeting, resolve any ambiguities or problems, and incorporate these changes with the text of the previous Standard or draft Standard.

2.4 Standard

2.4.1 The referees and officers may, by near unanimous decision (not more than one no vote), declare the draft Standard, as mentioned in the previous paragraph, as being the proposed Standard.

2.4.2 A proposed Standard shall be distributed to all FST voting members for a mail ballot. This ballot shall be based solely on the text of the proposed Standard as distributed.

2.4.3 Each ballot returned shall be signed by the voting member submitting it. An affirmative vote of at least two thirds of the voting members shall adopt the document. Such adoption makes the draft Standard the current, official FST Standard which supersedes all prior Standards.

2.5 Charter

2.5.1 At a formal FST meeting, the charter may be amended by a simple majority of voting members present provided that at least one third of all voting members are present; such amendments become effective at the end of the current formal meeting.

2.5.2 At other than a formal FST meeting, the charter may be amended by a simple majority of all voting members, such vote to be taken by signed mail ballots.

D. CHARTER

3. MEMBERSHIP

3.1 General

Membership in the FST is a privilege, not a right. An invitation for voting membership may be extended to those who the FST feels can contribute to the goals of the Standard and the FST. There are several classes of participation in the efforts of the FST. Membership in each class has no specified term but continues from the time when membership is initiated to the conclusion of the next formal meeting.

3.2 Voting Members

3.2.1 Voting members are individuals who are elected into such membership at the concluding session of a formal FST meeting. Any voting member who resigns between formal meetings shall not be replaced until the membership elections at the conclusion of the next formal meeting. A newly elected voting member gains voting rights only after all voting members have been elected. A significant professional FORTH background is required of voting members.

3.2.2 Each voting member present at a formal meeting shall indicate in writing his or her desire to continue as a voting member. Only these voting members can vote in a general session of a formal meeting on any matters affecting the Standard or the Charter and on the election of all voting members.

3.2.3 Voting members are elected by a simple majority of those voting members present. The number of voting members shall be limited to thirty (30). Individuals eligible to be elected are selected from each of the following ordered categories in order, until the number of voting members reaches the limit.

3.2.3.1 Category 1: current voting member who have actively participated in at least two days of a formal meeting. Voting members are expected to actively participate in subteam meetings and all general sessions.

3.2.3.2 Category 2: current voting members who are not eligible by Category 1, but who have requested in writing that his or her voting membership be maintained.

3.2.3.3 Category 3: eligible candidates. Eligible candidates will be presented to the voting members then elected as follows:

3.2.3.3.1 If the number of eligible candidates does not exceed the number of openings for voting membership, each candidate is voted upon and accepted by a simple majority.

3.2.3.3.2 If the number of eligible candidates does exceed the number of openings for voting membership, candidates will be voted upon by ballot whereby each voting member may vote for up to the number of openings remaining. Those candidates receiving the most votes will be elected until there are no more openings for voting membership.

3.3 Candidates

3.3.1 Candidates are individuals who desire to actively participate in and support the FST by becoming voting members.

3.3.2 To be eligible, each Candidate must: declare in writing to the secretary at the first general session of a formal FST meeting that he or she is a Candidate, actively participate in subteam meetings and all general sessions at a formal FST meeting, and have a significant professional background in FORTH. The Chairperson may request information or ask questions of any candidate to determine his or her technical knowledge and experience. Candidates are expected to submit proposals, participate in the discussions of the formal meeting, and contribute to the work and voting of subteams.

3.4 Observers

3.4.1 Observers are individuals who attend a formal meeting but are neither voting members nor candidates. At the discretion of the Chairperson, they may contribute to the discussion at general sessions and to the work of subteams. The number of observers allowed at a formal meeting may be limited by the Chairperson.

3.5 FST Sponsors

3.5.1 FST sponsors are individuals or organizations who contribute funds and other assistance to aid the work of the FST. FST sponsors have no duties or responsibilities in the FST, but they will receive copies of proposals and comments considered at a formal meeting, and drafts and adopted standards prepared as a result of that meeting.

3.5.3 FST sponsorship exists from the end of one formal meeting to the end of the next formal meeting.

3.5.3 Qualification of FST sponsors may be determined by a simple majority vote at a formal FST meeting. If no such qualification exist, the Chairperson may specify qualifications, including the amount of financial contributions, which will remain in effect until the next formal FST meeting.

4. OFFICERS

4.1 General

There shall be four types of elected officers of the FST: the Chairperson, the Secretary, the Treasurer, and one or more Referees. Each officer shall be elected at a formal meeting of the FST and serve until the next formal meeting.

4.2 Vacancies

If any office other than the Chairperson becomes vacant between formal meetings, the Chairperson may appoint a replacement. If the office of the Chairperson becomes vacant between formal meetings, a new Chairperson shall be elected by an informal majority vote of the remaining officers. At any formal meeting, any officer, including the Chairperson, may be replaced by a simple majority vote of the voting members present at that meeting.

4.3 Chairperson

4.3.1 The Chairperson is responsible for governing the general business of the FST. He or she is responsible for implementing the FST's Charter and any other requirements specified by the Standard.

4.3.2 The Chairperson's term of office shall be from the conclusion of the formal meeting at which he or she is elected to the conclusion of the next formal meeting. The election of a Chairperson is held at the concluding general session of a formal meeting after the election of voting members; hence, newly elected voting members may vote for the Chairperson. Only voting members are eligible to be elected Chairperson.

4.3.3 The Chairperson shall conduct each formal meeting. In general, the meetings will follow the current Robert's Rules of Order; however, the Chairperson may determine the specific rules for a formal meeting.

4.3.4 Any matter needing a decision between formal meetings not specified by this Charter shall be decided by the Chairperson.

4.3.5 The Chairperson has duties and responsibilities specified elsewhere in this Charter.

4.4 Secretary

4.4.1 The Secretary is responsible for recording the activities and results of the FST.

4.4.2 The Secretary is elected at the first general session of a formal meeting and serves until a Secretary is elected at the beginning of the next formal meeting.

4.4.3 The Secretary has many responsibilities.

4.4.3.1 The Secretary is responsible for collecting, maintaining, and archiving the official copies of the Standard, the Charter, all other FST documents, correspondence, and lists of the FST members of each class.

4.4.3.2 During a formal meeting, the Secretary is responsible for:

(a) Keeping the minutes of the general sessions, including all votes taken. For votes affecting the Standard or Charter, he or she shall: record the number of voting members present, determine if a quorum is present, determine the number of affirmative votes required for the vote to pass, the number of voting members voting in the affirmative and negative, and the result of the vote.

(b) Recording and verifying the attendance and membership class of each attendee.

(c) Recording the recommendations of subteams.

4.4.3.3 The Secretary is also responsible for collecting, archiving, and distributing proposals before a formal meeting. He or she is also responsible for incorporating proposals accepted during a formal meeting into the Standard or Charter. Other officers aid the Secretary in these duties.

4.5 Treasurer

4.5.1 The Treasurer is responsible for managing the financial business of the FST. He or she is responsible for maintaining accurate and current financial records and for accepting and dispersing funds for official FST activities.

4.5.2 The Treasurer's term of office shall be from the conclusion of the formal meeting at which he or she is elected to the conclusion of the next formal meeting. The election of a Treasurer is held just after the election of the Chairperson. Only voting members are eligible to be elected Treasurer.

4.6 Referees

4.6.1 At the conclusion of a formal meeting there may be additional technical work required to prepare a draft Standard or Charter. This work shall be performed by the officers of the FST, including a group of Referees. They should be individuals who have superior knowledge and experience in the implementation and use of FORTH.

4.6.2 At least three and no more than five Referees shall be elected by a majority of the voting members present at the concluding general sessions of a formal meeting. This takes place after the election of voting members. A Referee's term is from election at the end of one formal meeting until the end of the next formal meeting. Only voting members are eligible to be elected as Referees.

4.6.3 The Referees shall adopt methods and rules as they deem appropriate to complete their work; they may be informal. However, any matter committed to the Referees for resolution must achieve near unanimous agreement (not more than one no vote). Lacking that, the matter shall be omitted from further action pending further consideration at the next formal meeting.

5. EXPERIMENTAL PROPOSALS

5.1 General

5.1.1 Since FORTH is an extensible language and subject to evolution, the Standard may contain a section describing experimental proposal to aid in the analysis of and the decision for or against future adoption into the Standard. After the results of experimentation are known, each proposal will be considered, at a future formal meeting, for inclusion into the Standard.

5.1.2 An experimental proposal may be individual FORTH words, sets of related words, or specifications for part of the Standard. Experimental proposals may be derived from ordinary proposals or other contributions.

D. CHARTER

5.2 Required Information

Each experimental proposal must contain the following minimum information:

5.2.1 A description of the proposal including an overview of its functions and its interactions with existing FORTH words.

5.2.2 A glossary entry of each word in the form and notation of the Standard.

5.2.3 A statement by the author(s) indicating why the proposal meets inclusion into the Standard. Both advantages and disadvantages should be discussed.

5.3 Suggested Information

It is suggested that each experimental proposal also include:

5.3.1 A source definition for each word in the proposal. High level definitions using Standard words are preferred, but new primitive words may be defined in an assembly language of one commonly-known processor. Sufficient documentation should be provided so that implementation on other processors is direct.

5.3.2 An example showing usage of the new words.

6. VOTING

6.1 General

Only voting members have the right to vote on proposals affecting the Standard, a draft Standard, or this Charter.

6.2 Advisory Votes

At the discretion of the Chairperson, advisory votes may be requested at a formal meeting. At the discretion of the Chairperson, all attendees may participate in an advisory vote.

6.3 Method

Any vote at a formal meeting may be by show of hands or, at the discretion of the Chairperson, by an informal secret paper ballot or a roll call.

D. CHARTER

6.4 Number

A vote to adopt a proposal into the draft Standard or to change the Standard, except for the Experimental Proposals section of the Standard requires a two-thirds affirmative vote of the voting members present at a general session of a formal meeting, provided that the number of votes cast are at least two thirds of that morning's quorum count. To adopt an experimental proposal into the Experimental Proposals section of the draft Standard or to change this Charter, an affirmative vote of a simple majority is required. Accepting any other procedural matter at a formal meeting requires only a simple majority affirmative vote.

6.5 Proxies

All votes must be cast by the particular voting member eligible to vote. No proxy voting is allowed.

APPENDIX E. PROPOSAL/COMMENT FORM

The following pages are the proposal and/or comment submittal form. The form includes instructions which should be explanatory. Copies of submitted proposals and comments will be made available to FORTH Standards Team members and to team sponsors.

```
             FST Proposal and Comment Submittal Form
------------------------------------------------------------------
FST USER  Title:                              Proposal Number:
 ONLY --> Related Proposals:                  Disposition:
==================================================================
Keyword(s):                                        Category:
                                     ( ) Proposal  or  ( ) Comment
FORTH Word(s):                       Section #(s):
------------------------------------------------------------------
Abstract:






------------------------------------------------------------------
Proposal and Discussion:

























-----------------------------------------------------------------
Submitted by:                                       Date:
                                                    Page    of
==================================================================
FORTH Standards Team; PO Box 4545; Mountain View, CA 94040 820801
```

Proposal and Comment Submittal Form Instructions

Please use the supplied forms for your entire proposal. The continuation form is only to be used if absolutely necessary; try to get your proposal to fit on the first sheet. If it helps, use a reducing copy machine to get more material onto the first sheet. If you must use multiple sheets, put the main idea onto the first sheet and less important material onto continuation sheets. Remember that material on continuation sheets may be overlooked.

The proposal forms have been produced on a computer system so that you may produce your proposals using your own computer system. If you print your proposal and form on your computer system, all of the information shown on the form(s) MUST be printed and in the same location.

The following are the instructions for each of the areas of the form:

1. Please think of the most appropriate keyword or keywords describing your proposal.

2. Select the best of the following categories of proposals:
   0 Nucleus Layer other than #1 (i.e., + AND)
   1 Memory Operations (i.e., @ CMOVE)
   2 Dictionary (i.e., ' FORGET)
   3 String Operations (i.e., WORD COUNT)
   4 Interpreter Layer other than #2 or #3 (i.e., ABORT .)
   5 Compiler Layer (i.e., : DO)
   6 Device Layer (i.e., BLOCK TYPE)
   7 Experimental (i.e., 32-bit stack entries)
   8 Other Technical (i.e., mono-addressing)
   9 Charter

3. Mark whether this is a PROPOSAL or a COMMENT.

4. Indicate which FORTH word or words are relevant.

5. Indicate which section or sections of the Standard are relevant.

6. The abstract must be kept short. The title, keywords, category, and abstract may be used in a database for organization and display on a terminal during a Standards Team meeting.

7. Detail your proposal and provide supporting discussion.

8. Indicate the name of the submitter or the names of the submitters.

9. Finally, date the submittal and number each page.

```
      FST Proposal and Comment Submittal Continuation Form
-----------------------------------------------------------------
FST USE ONLY -->                             Proposal Number:
=================================================================
```

```
-----------------------------------------------------------------
Submitted by:                                Date:
                                                   Page   of
=================================================================
FORTH Standards Team; PO Box 4545; Mountain View, CA 94040 820801
```