TERSE Standard Glossary
9/21/81

This is a description of the TERSE vocabulary. The words are presented in ASCII order. The first line of each entry shows a symbolic description of the action of the word: Symbols indicating which parameters are to be placed on the stack before executing the word, 3 dashes (---) indicating execution, then any parameters left on the stack by the word. In this notation, the top of the stack is to the right. If the place of the word in the input string is not completely obvious, it is shown explicitly. If no dashes are shown the word does not affect the stack. Symbols are used as follows:

b	Block number
c	7-bit ASCII character code
f	Flag: 0=False, non-zero=True. All words which return a flag return 0 or 1.
m n p q r	16-bit integers
nnnn pppp	The name of a word
ssss	A string of characters

Immediately following the name of a word, certain characters may appear within paraentheses. These denote some special action or characteristics:

C	The word may be used only within a colon-definition. A following digit (C0 or C2) indicates the number of memory cells used when the word is compiled if other than one. A following + or - sign indicates that the word either pushes or pops a value on the stack during compilation. (This action is not related to its action during execution.)
E	The word may not normally be compiled within a colon-definition.
P	The word has its immediate bit set; it is executed directly, even when encountered during compile mode.
U	The word applies to a user variable (in a multi-user system each user would have his own copy.)
X	The word is cross-compilable.
G	GAS Terse only

Unless stated otherwise, all references to numbers apply 16-bit integers, with the most significant bit as the sign bit and the negative in two's complement form. Similarly, all arithmetic will be assumed to be 16-bit signed integer arithmetic with error and overflow indication unspecified.

Standard Definitions

!	(X)	m p ---	Store m at address p.
#DRVS	(X)	--- p	Returns ~~the address of a variable containing~~ the number of drives in your system.
'	(P)	' nnnn --- p	Leave address of verb nnnn on stack. A compiler directive, ' is executed when encountered in a colon definition; the address of the following word's code field is found immediately (at compilation) and stored in the dictionary (after the address of LIT) as a literal to be placed on the stack at execution time. Within a colon definition, ' nnnn is identical to: LIT [' nnnn ,]
(	(P)	(ssss)	Ingore a comment that will be delimited by a right parentheses. CAUTION: No imbedded right parentheses, and no leading or trailing space is required.
*	(X)	m n --- p	16-bit signed multiply. p=m*n
+	(X)	m n --- q	16-bit integer addition. q=m+n
+!	(X)	m p ---	Add integer m to value at address p.
+B!	(X)	m p ---	Add the low-order 8 bits of m to the byte at address p.
+BLOCK		m --- b	Return the sum of m plus the number of the block currently being interpreted.
+LOOP	(C,X)	m ---	Add m to the loop index. Exit from the loop is made when the resulant index reaches or passes the limit, if m is greater than zero; or when the index is less than (passes) the limit, if m is less than zero.
,	(X)	m ---	Store m into the next available dictionary word, advancing the dictionary pointer.
,"	(X)	," ssss"	Stores a message delimited by " at the next available dictionary location with the length of the message being the first byte.
-	(X)	m n --- q	16-bit integer subtraction: q=m-n

-!	(X) m p --- Subtract integer m from the value at address p.
-->	(P) (Pronounced "next block") Continue interpretation with the next block (Equivalent to 1 +BLOCK CONTINUED). When used in the last block of a file, a ;S will be executed.
->L	(X) m p --- n Double-precision logical shift right on m for p bits with the value returned as n. See also <-L (Shift left).
-DUP	(X) m --- m (if zero) m --- m m (non-zero) Duplicate the top value on the stack if it is not zero. Used with IF to avoid the need for an ELSE with a DROP.
.	m --- Print the value on the stack as an integer, converted according to the current number base.
."	(P) ." ssss" Transmit a message delimited by a " to the selected output device.
.BLK#	Causes block number of each screen to be printed out as it is loaded.
.HOU	Change output device to HOUSTON INSTRUMENTS printer. This output routine should run any standard printer.
.LIST	Change output device to printer.
.NBLK#	Turns off the .BLK# option.
.NLIST	Change output device to CRT.
.NSCR	Turns off the .SCR option.
.SCR	Causes each screen to be listed out as it is loaded.
/	(X) m n --- q 16-bit signed integer divide, q=m/n. The quotient is truncated and the remainder is lost. (Actually defined as /MOD DROP)
/MOD	(X) m n --- r q 16-bit integer divide, m/n. The quotient is left on top of the stack, the remainder beneath. The remainder has the sign of the quotient, q.

0	(X)	--- n	Puts a 0 on the stack. (0 is a CONSTANT)
0<	(X)	m --- f	True if m is negative.
0<>	(X)	m --- f	True if m is not equal to 0.
0<=	(X)	m --- f	True if m is negative or equal to 0.
0<FRAME	(X)		See <FRAME for a complete description.
0=	(X)	m --- f	True if m is zero.
0>	(X)	m --- f	True if m is positive and non-zero.
0>=	(X)	m --- f	True if m is positive or equal to 0.
0FRAME>	(X)		See FRAME> for a complete description.
1	(X)	--- n	Puts a 1 on the stack. (1 is a CONSTANT)
1+	(X)	m --- q	q=m+1
1+!	(X)	p ---	Add 1 to the contents of the word at location p.
1+B!	(X)	p ---	Add 1 to the contents of the byte at location p.
1-	(X)	m --- q	q=m-1
1-!	(X)	p ---	Subtract 1 from the contents of the word at location p.
1-B!	(X)	p ---	Subtract 1 from the contents of the byte at location p.
1<FRAME	(X)		See <FRAME for a complete description.
1FRAME>	(X)		See FRAME> for a complete description.
1LOCAL@	(X)		See LOCAL for a complete description.

1LOCAL	(X) See LOCAL for a complete description.
1PARAM@	(X) See PARAM for a complete description.
1PARAM	(X) See PARAM for a complete description.
2*	(X) m --- q q=m*2 (Shift left)
2+	(X) m --- q q=m+2 (Increment by 2)
2-	(X) m --- q q=m-2 (Decrement by 2)
2/	(X) m --- q q=m/2 (Shift Right)
2<FRAME	(X) See <FRAME for a complete description.
2FRAME>	(X) See FRAME> for a complete description.
2DROP	(X) m n --- Drop the top two values from the stack (to drop a double precision number for example).
2DUP	(X) m n --- m n m n Duplicate the top two values on the stack.
2LOCAL@	(X) See LOCAL for a complete description.
2LOCAL	(X) See LOCAL for a complete description.
2PARAM@	(X) See PARAM for a complete description.
2PARAM	(X) See PARAM for a complete description.
2SWAP	(X) m n p q --- p q m n Swap two pairs of values (e.g. double-precision numbers).
3<FRAME	(X) See <FRAME for a complete description.
3FRAME>	(X) See FRAME> for a complete description.
3LOCAL@	(X) See LOCAL for a complete description.

3LOCAL	(X)	See LOCAL for a complete description.
3PARAM@	(X)	See PARAM for a complete description.
3PARAM	(X)	See PARAM for a complete description.
4<FRAME	(X)	See <FRAME for a complete description.
4FRAME>	(X)	See FRAME> for a complete description.
4LOCAL@	(X)	See LOCAL for a complete description.
4LOCAL	(X)	See LOCAL for a complete description.
4PARAM@	(X)	See PARAM for a complete description.
4PARAM	(X)	See PARAM for a complete description.
:	(X) : nnnn	Create a dictionary entry defining nnnn as equivalent to the following sequence of TERSE words. Set STATE to compile mode. (Extension: Set the context vocabulary equivalent to the current vocabulary).
;	(C,P,X)	Terminate a colon-definition and set STATE to immediate mode.
;S	(E)	Stop interpretation of a symbolic block.
<	(X) m n --- f	True if m<n
<-L	(X) m n --- p	Logical double-precision shift left on m for n bits with the value returned as p. See also ->L (logical shift right).
<<	(E) m n ---	Use: m n << verbs >> Similar to a DO...LOOP except that these are conditionals that may be employed during interpretation, although they are much slower. In conjuction with the words [and] they may be used within a colon definition to control compilation, although they are not compiled. These words can be nested.
<=	(X) m n --- f	True if m<n or m=n

<>	(X)	m n --- f	True if m is unequal to n
<FORK	(X)	n <FORK pppp1 pppp2...ppppn FORK>	Execute the nth verb following <FORK and remove n from the stack. Execution resumes with the verb following FORK>. All verbs are skipped if n is greater than the number of verbs between <FORK and FORK>. Numeric literals.
<FRAME	(X)	n ---	Create a stack frame with n local values. All previous values that are on the stack can be accessed using the various PARAM verbs (see PARAM). All local values pushed on the stack can also be accessed using the various LOCAL verbs (see LOCAL). The stack frame can be cleared using the various FRAME> verbs (see FRAME>). In addition, the following <FRAME verbs are available: 1<FRAME, 2<FRAME, 3<FRAME, and 4<FRAME. Reword
<STK	(X)		Marks the top of the stack at compilation time. Usually used to detect whether there are an unbalanced number of IF-ELSE-THENs, etc. Should be used in conjuntion with STK>. See also <STKD and <STKH.
<STKD	(X)		Executes the <STK and the DECIMAL verbs.
<STKH	(X)		Executes the <STK and the HEX verbs.
=	(X)	m n --- f	True if m=n
>	(X)	m n --- f	True if m>n
>=	(X)	m n --- f	True if m>n or m=n
>>	(E)		Terminate a conditional interpretation sequence begun by <<.
>R	(C,X)	m ---	Push m onto top of the return stack. (See R>)
?		p ---	Print the value contained at address p according to the current base.
?BELL	(X)		Transmits a <?><BELL><CR><LF> to the selected output device.
@	(X)	p --- q	Return the word at location p.

A"	(X) A" ssss" --- q Makes a string similarly to ." but does not type it. Instead A" returns the string address q. The string may be typed by COUNT TYPE or STYPE.
A=	(X) m A= nnnn See ARRAY for a complete description.
ABORT	(X) m --- Enter the Abort sequence, reset the stack and the return stack. Then print the message at m, execute ?BELL, and return control to the terminal. This can be used in conjunction with A" .
ABS	(X) m --- q Leave the absolute value of a number.
AND	(X) m n --- q Bitwise logical AND of m and n.
ARRAY	(X) m ARRAY nnnn Define an array named nnnn and allocate m uninitialized words of RAM). The sequence i nnnn will return the address of the i-th word on the stack. The index should be in the range 0 <= i <= m-1, but no check is made for values exceeding this range.
ASM	(P) Switch the context pointer so that dictionary searches will begin at the Assembler Vocabulary. A CODE define automatically switches the CONTEXT to ASM.
B!	(X) m p --- Store the least significant 8 bits of m at byte-address p.
B,	(X) m --- Store the low 8 bits of m into the next available dictionary byte, advancing the dictionary pointer.
B:	m --- q q=m+308. Used for calculating block numbers of drive B.
B@	(X) p --- q Return the 8-bit value q found at byte-address p.
BA=	(X) m BA= nnnn See BARRAY for a complete description.
BARRAY	(X) m BARRAY nnnn Define an array named nnnn and allocate m uninitialized bytes of RAM). The sequence i nnnn will leave the address of the i-th byte on the stack. The index should be in the range 0 <= i <= m-1, but no check is made for values exceeding this range.

BASE (U) --- p
Returns the address of a variable containing the current number conversion base.

BASE?
Prints the current base minus 1 on the output device.

BEGIN (CO+,P,X) BEGIN ... WHILE ... REPEAT
or BEGIN ... END
Mark the start of a sequence of words to be executed repetitively. If ... WHILE ... REPEAT is used the loop will be repeated as long as the stack encountered by WHILE is TRUE (REPEAT merely effects an unconditional jump back to BEGIN); when WHILE sees a FALSE value (0) on the stack it causes an immediate exit out of the loop. In case the sequence can be written such that the test for completion is at the end ... END can be used conveniently to end the loop on a TRUE value or to go back to BEGIN on FALSE. Both WHILE and END drop the value they test.

BELL
Sends a BELL (07H) to the terminal.

BIT! (X) f m n ---
Stores bit value f (0 or 1) into bit m at address n.

BIT@ (X) m n --- ~~f~~ p non-zero, if set)
Returns the value of bit m (0 or 1) at address n.

BIT-CALC (X) m n --- m n
Returns a bit mask, m, and and the address, n, for bit m at address n. Used by BIT@ and BIT!.

BKSP (X)
Transmits a Backspace (08H) to the selected output device.

BLK/DISK (X) --- p
Returns the number of blocks per disk.

BLK/SIDE (X,G) --- p
Returns the number of blocks on one side of a disk.

BLK (U) --- p
A variable containing the number of the block being listed or edited. This variable is not used in the file system.

BLKMOVE (X) p q n ---
Move the n blocks starting at block p into the n blocks ~~starting at block p into the n blocks starting~~ at block q. Overlapping areas may be used. This works for physical blocks only.

BLKSHIFT m n q---
Shift blocks m thru n by the amount q. The block will be located at m+q thru n+q. This works for physical blocks only.

BLOCK b --- p

Leave the first address of Block b. If the block is not already in memory, it is transferred from disk into whichever core buffer has been least recently accessed. If the block occupying that buffer has been updated, it is FLUSHed (See FLUSH) before Block b is read into the buffer.

BMOVE (X) p q n ---

Move the n bytes starting at byte-address p into the n byte-cells starting at byte-address q. The contents of p is moved first.

BONE (X) p ---

Set the word at location p to 1.

BPTR --- n

A variable containing a pointer to the most recently used disk block buffer. Disk block buffers are headed by a link to the next block and the block number followed by the data. A link of 0 indicates the end of the chain.

BR= (X) m BR= nnnn

See BRAMALLOT for a complete dexcription.

BRAMALLOT (X) m BRAMALLOT nnnn

Allocate m contiguous bytes of RAM such that when nnnn is executed, the address of the beginning of the RAM is returned.

BTABLE (X) BTABLE nnnn

Define the beginning of a table of bytes. The values to be entered into the table must follow the definitions of the table. The sequence i nnnn will leave the address of the i-th byte on the stack. The index should be 0 <= i < number-of-table-entries. No check is made on the range of i.

BUFFER b --- p

Obtain a core buffer for Block b, leaving the first buffer cell address. The block is not read from disk, and is automatically marked as updated.

BUFFER1 (X) --- p

Returns the address of the first disk buffer.

BUFFER2 (X) --- p

Returns the address of the second disk buffer.

BV= (X) BV= nnnn

Executes a 0 BVARIABLE.

BVARIABLE (X) m BVARIABLE nnnn

Create a word nnn which when executed will push the address of an 8 bit variable (initialized to the low 8 bits of m) onto the stack.

BYE

Exit to ICE or GAS monitor.

```
BZERO    (X)           p ---
         Set the byte at location p to 0.

C=       (X)              m C= nnnn
         See CONSTANT for a complete description.

CASE     (X)  (C2+,P)   m n --- (m)
                        m n CASE <action for m=n> ELSE <drop> THEN
         If m equals n,  drop  both  m  and  n  and  execute the words
         directly following  CASE  until  the  next  ELSE  or  THEN  ;
         otherwise, drop n  but leave  m and  execute the  words after
         ELSE (or THEN if  no ELSE is used).  The  selection of one of
         many cases can be done by:
                    m n1 CASE <action for m=n1> ELSE
                      n2 CASE <action for m=n2> ELSE
                      n3 CASE <action for m=n2> ELSE
                         <otherwise action> THEN THEN THEN
         (m will still be on the stack in the otherwise section).

CCALC               m --- q
         Converts a link address  m  to  the  code  address  q of that
         routine.

CODE     (X)  CODE nnnn
         Create a dictionary entry defining  nnnn as equivalent to the
         following sequence of  assembler  code.  (Extension:  set the
         context vocabulary to Assembler.) REWORD (EX: CODE ... NEXT)

COM      (X)        m --- q
         Complement each bit of m (Leave one's complement).

CONSTANT (X)        m CONSTANT nnnn
         Create a word which  when executed  pushes m  onto the stack.
         ~~Since the "constant" value m maybe modified by the sequence q
         ' nnnn 3 + ! it is oftentimes advantageous to define a
         variable as a constant, particulary if it is accessed more
         than it is modified.~~

CONTEXT  (U)        --- p
         Return the address of a variable  containing a pointer to the
         vocabulary in  which dictionary  searches are  to begin.  See
         CURRENT.

CONTINUED (E)       b ---
         Continue  interpretation  at  block  b.  (The  preferred
         implementation in multi-buffer systems is such that the block
         buffer currently being accessed  will be used  for storage of
         block b, leaving other buffers unaffected.)

COPY                m n ---
         Copies block m  to block  n.  This works  for physical blocks
         only.

COUNT    (X)  p --- m n
         Leave byte-address  m and  byte-count n  of a  message string
         beginning at word-address  p.  It is presumed  that the first
         byte at p contains the byte-count and that the actual message
         starts with the second byte  in location p.  Typically, COUNT
```

			is followed by WRITE or TYPE
CR			Transmit a <CR><LF> to the selected output device.
CURRENT	(U)		A variable containing a pointer to the vocabulary into which new words are to be entered. CURRENT @ @ leaves the link address of the next entry to be defined.
CYL	(X)	--- m	Returns the address of a system variable containing the currently selected track/cylinder.
CYL/DISK	(X)	--- m	Returns the number of tracks/cylinders per disk.
DATA	(X)	DATA nnnn	Define the beginning of a set of data. The values to be used must be following the definition of the data (, and B, are used to store the values). When nnnn is executed it will leave the address of the first byte of data on the top of the stack.
DEBUG			Switch in the DEBUG vocabulary.
DECIMAL			Set the numeric conversion base to decimal mode.
DED			Executes the DECIMAL and EDIT verbs.
DEFINITIONS			Set CURRENT equal to CONTEXT. See CURRENT, CONTEXT and VOCABULARY.
DIR		m n ---	Lists the first line of each block that starts with "(" from block n to block m-1.
DISKCOPY			Copys all blocks from disk drive A to drive B.
DLIST			Lists the context vocabulary verbs with their link field and code field addresses. Repeated pressings of the space bar continue the listing.
DLIT	(C)	DLIT l h	Automatically compiled before each double precision literal encountered in a colon definition. Execution of DLIT causes the contents of the next 2 instruction words to be pushed onto the stack. High value is on top.

DO	(C,X) m n ---	Begin a loop, to be terminated by LOOP or +LOOP. The loop index begins at n, and may be modified at the end of the loop by any positive or negative value. The loop is terminated when an increment index reaches or exceeds m, or when a decremented index becomes less than m. Within nested loops, the word I always returns the index of the innermost loop that is being executed, while J returns the index of the next outer loop, and K returns the index of the second outer loop.
DP	--- q	A variable containing a pointer to the next available dictionary location.
DP+!	n ---	Add the signed value n to the dictionary pointer (DP). As DP may be an internal register rather than a VARIABLE, it is accessible only through HERE and DP+!
DRIVE	(X) --- p	Returns the address of a system variable containing the currently selected drive.
DROP	(X) m ---	Drop the top value from the stack.
DUMP	m ---	Lists 64 bytes in hex format starting at address m on the output device. Repeated pressings of the space key causes the next 64 bytes to be listed. Press any other key to exit.
DUP	(X) m --- m m	Duplicate the top value on the stack.
E-C		Marks all block-buffers as empty. Updated blocks are not flushed. Contents of buffers are undefined.
EDIT	(P)	Brings in the EDIT vocabulary, thereby making its verbs accessible. (ie. CONTEXT is set to EDIT).
ELSE	(C2,P,X)	Precede the false part of an IF...ELSE...THEN conditional. It may be ommitted if the false part is empty.
END	(C2-,P,X) f ---	Mark the end of a BEGIN..END loop. If f is true the loop is terminated. If f is false, control returns to the first word after the corresponding BEGIN.
EX		See FLUSH for complete details.
EXEC	q ---	Depending on the STATE variable either q is stored in the dictionary or address q is loaded into HL and a PCHL is done.

FCALC FCALC nnnn --- p
Returns the code address of verb nnnn.

FILECOPY m n ---
Copies blocks n thru m inclusive from drive A to drive B. This works for physical blocks only.

FILES
Informs the system that you are working with files and not with physical blocks. See NOFILES.

FIX FIX nnnn
Allows you to redefine verb nnnn in the dictionary and have all higher level verbs use the new definition.

FLD --- p
A variable containing the field length reserved for a number during output conversion.

FLOAD FLOAD nnnn
LOADs file nnnn into memory.

FLUSH
Write all blocks that have benn flagged as "updated" to disk. Return when output is done. (See UPDATE)

FORGET FORGET nnnn
Delete all verbs after and including nnnn. Frees up memory and deletes verbs from dictionary. Should not be used for verbs in the basic system.

FORK> (X)
See <FORK

FRAME> (X) n---
Removes the current stack frame that was set up by <FRAME. This includes popping all local values and n previous values. See <FRAME for a complete description.

FSLD FSLD nnnn
Load the system from a file named nnnn.

FSYSAVE FSYSAVE nnnn
Save the entire dictionary as binary data into the file named nnnn. To reload the system, see FSLD.

GET --- p
Return the address of a variable containing the address of the character input routine.

GETC --- n
Inputs an ASCII character n from the selected input device.

H. m ---
Convert and output in hexadecimal mode, unsigned, and preceded by a blank. BASE is unchanged. Format specifications are observed.

HABORT	(X)	m ---	Just like ABORT except the string at HERE is displayed.
HELP	(E)	---	List the dictionary. This starts with the CONTEXT vocabulary.
HERE	(U)	--- p	Return the address of the next available dictionary location.
HEX			Switch the numeric conversion base to hexadecimal.
HEXLIST		m b ---	List the ASCII contents and hexadecimal contents of block b starting at byte m on the selected output device. This works for physical blocks only.
HEXSHOW		b ---	Lists ASCII contents and hexadecimal contents of block b on the selected output device. Repeated pressings of the space bar on the control terminal will list the next 16 bytes of the block. A <CR> will begin listing at the next physical block, and pressing any other key will terminate the sequence. This works for physical blocks only.
I	(C,X)	--- m	Returns the index of an intermost DO-loop.
I+	(X)	m --- q	Adds m to the index of the intermost DO-loop. q=m+I
ICE			Enter the ICEbox monitor. Can be used to debug machine code routines.
IF	(C2+,P,X)	f IF <true part> ELSE <false part> THEN f IF <true part> THEN	IF s the first word of a conditional. If f is true, the words following IF are executed and the words following ELSE are not executed. The ELSE part of the conditional is optional. If f is false, words between IF and ELSE, of between IF and THEN when no ELSE is used, are skipped. IF-ELSE-THEN conditionals may be nested.
IFEND	(E)		Terminate a conditional interpretation sequence begun by IFTRUE.
IFTRUE	(E)	f IFTRUE...OTHERWISE...IFEND ---	Unlike IF..ELSE..THEN, these conditionals may be employed during interpretation. In conjuction with the words [and] they may be used within a colon definition to control compilation, although they are not to be compiled. These words cannot be nested.
IMMED			Mark the most recently made dictionary entry such that when encountered at compile time it will be executed rather than

			compiled.
INP	(X)	m --- n	Inputs from port m returning value n.
J	(C,X)	--- m	Within a nested DO-loop, return the index of the next outer loop.
J+	(X)	m --- q	Adds m to DO-loop index J. q=m+J
K	(C,X)	--- m	Within a nested DO-loop, return the index of the second outer loop.
K+	(X)	m --- q	Adds m to DO-loop index K. q=m+K
LAST		--- p	A variable containing the compilation address of the most recently created dictionary entry.
LEAVE	(C,X)		Force termination of a DO-loop at the next opportunity by setting the loop limit equal to the current value of the index. The index itself remains unchanged, and execution proceeds normally until LOOP or +LOOP is encountered.
LINE		m --- p	Leave the word address of the begininning of line m for the block whose number is contained at BLK. (For editing purposes a block is divided into 16 lines, numbered 0-15, of 64 characters.)
LINELOAD		m b ---	Begin interpreting at line m of Block b. (0 <= m <= 15) This works for both physical blocks and files (where m is the relative block number if you are working with files).
LIST		b ---	List ASCII symbolic contents of block b on the selected output device. This works for physical blocks only.
LIT	(C)	LIT m	Automatically compiled before each literal encountered in a colon definition. Execution of LIT causes the contents of the next dictionary cell to be pushed onto the stack.
LITERAL		n m ---	Store n in the dictionary (as 2 words:LIT n). Does nothing if STATE is set to compile mode. If DPREC=0 then m is dropped else 3 words are compiled: DLIT n m. ?
LITES	(G)	m ---	Sets the beginning value for the LEDs on the GAS.

LOAD b ---
Begin interpreting at block b. The block must terminate its own interpretation with ;S , --> or CONTINUED.

LOCAL (X) n---q
Returns the address, q, of the nth local value on the stack created by the current stack frame (see <FRAME). When n=1, the address of the first local value pushed on the stack is returned. In addition, the following LOCAL verbs are available: 1LOCAL, 2LOCAL, 3LOCAL, and 4LOCAL. To get the actual local values instead of the stack addresses, the following LOCAL verbs are available: 1LOCAL@, 2LOCAL@, 3LOCAL@, and 4LOCAL@. To use local variables, create a stack frame using <FRAME, then execute an m nLOCAL !.

LOOP (C,X)
Increment the DO-loop index by one, terminating the loop if the new index is equal to or greater than the limit.

MAP
Maps out the EDIT and ASM vocabularies and maps in the Screen RAM. Must be executed before using screen ram.

MAX (X) m n --- p
Leave the greater of the two numbers.

MEMAP@ (G) --- m
Returns the current value of the memory map.

MEMAP! (G) m ---
Sets the current memory map to m, but does not do any mapping.

MIN (X) m n --- p
Leave the lesser of the two numbers.

MINUS (X) m --- -m
Negate a number by taking its two's complement.

MOD (X) m n --- r
Leave the remainder of m/n, with the same sign as m.

MOVE (X) p q n ---
Move the contents of n memory cells beginning at address p into n cells beginning at address q. The contents of p is moved first; overlapping of data can occur.

NAND (X) m n --- q
Logical AND followed by COMplement.

NEXT (X)
End of Code code; terminate a ~~code~~ CODE definition. Capital letters

NEWVOCAB
Create a new vocabulary nnnn and append it to the other vocabularies.

NOFILES	Informs the operating system that you will be referring to physical blocks and not to files.
NOR	(X) m n --- q Logical OR followed by COMplement.
NOT	(X) m --- f Equivalent to 0=
OK	Prints a <CR> and "OK" on the output device.
ONE	(X) p --- Sets the word at location p to 1.
OR	(X) m n --- q Bitwise logical inclusive OR of m and n.
OTHERWISE (E)	An interpreter-level conditional word. See IFTRUE.
OUTP	(X) m n --- Outputs byte-value m to output port n. The high byte of n goes out on the upper address lines for sub-port numbers.
OVER	(X) m n --- m n m Push the second stack value.
PAGE	Clears the terminal screen or performs an action suitable to the output device currently active.
PARAM@	(X) See PARAM for a complete description.
PARAM	(X) n---q Returns the address, q, of the nth value on the stack that was there before the <FRAME verb was executed (see <FRAME). When n=1, the address of the top value that was on the stack will be returned. In addition, the following PARAM verbs are available: 1PARAM, 2PARAM, 3PARAM, and 4PARAM. To get the actual parameter values instead of the stack addresses, the following verbs are available: PARAM@, 1PARAM@, 2PARAM@, 3PARAM@, and 4PARAM@.
PDSTAT	(X) --- m Returns the Disk Status Port number.
PDDATA	(X) --- m Returns the Disk Data Port number.
PDCMD	(X) --- m Returns the Disk Command Port number.
PDTRK	(X) --- m Returns the Disk Track Select Port number.

PDSECT (X) --- m
Returns the Disk Sector Select Port number.

PDSEL (X) --- m
Returns the Disk Select Port number.

PICK (X) n --- q
Return the nth value on the stack~~, not counting n itself~~ (2 PICK is equivalent to OVER). 1 PICK = DUP

PIMODE (X) --- m
ICE port number for interrupts.

PLDATA (X) --- m
Returns the List Data Port number (Printer Data Port).

PLSTAT (X) --- m
Returns the List Status Port number (Printer Status Port).

POLLC --- n
Inputs an ASCII charactern from the selected input device. n will be zero if a character is not ready.

PRINTOUT m n ---
Lists ASCII contents of blocks n upto but not including m on selected output device. Only blocks starting with "(" are listed. The listing is prefaced by a DIR listing.

PROT
Turns on write-protection circuits in the ICEbox. Makes it impossible to write to locations below 4000H.

PTDATA (X) --- m
Returns the Terminal Data Port number.

PTSTAT (X) --- m
Returns the Terminal Status Port number.

PUT
A variable containing the address of the put character output routine.

PUTC n ---
Outputs ASCII character n to the selected output device.

R= m R= nnnn
See RAMALLOT for a complete description.

R> (C,X) --- n
Pop the value from the return stack and push it onto the user stack. See >R.

RAMALLOT m RAMALLOT nnnn
Allocate m contiguous words of RAM such that when nnnn is executed the beginning address of the RAM will be pushed on the stack.

RAMLEN		--- n	Returns a value which is the amount of variable space used since the last RAMMARK verb was executed.
RAMMARK			Remembers the next available variable location (VARHERE). Ususally used in conjunction with RAMLEN to determine the amount of variable space used.
REPEAT	(C2-,P,X)		Effect an unconditional jump back to the beginning of a BEGIN..WHILE..REPEAT loop. See BEGIN.
REPLACE		REPLACE nnnn1 nnnn2	Makes all uses of the old verb named nnnn1 the same as the verb nnnn2.
ROT	(X)	m n p --- n p m	Rotate the top three values on the stack, bringing the deepest to the top.
RP@	(X)	--- p	Return the address of the top of the return stack.
S!	(X)	p m ---	Executes the SWAP and ! verbs (ie. Stores m at address p).
SB!	(X)	p m ---	Executes the SWAP and B! verbs (ie. Stores the least significant 8 bits of m at byte-address p).
SCR		--- q	A variable whose value is the current block used for the input string being interpreted.
SHOW		b ---	List ASCII symbolic contents of block b on the selected output device. Repeated pressings of the space bar on the control terminal will list the next block in sequence. Pressing any other key will terminate the sequence.
SEC/TRK	(X)	--- m	Returns the number of sectors on a track.
SECTOR	(X)	--- m	Returns the address of a system variable containing the currently selected sector.
SIDE/DISK	(X,G)	--- m	Returns the number of sides on the disk (ie. 1 for single-sided disks).
SIDE	(X,G)	--- m	Returns the address of a system variable containing the currently selected side of the disk.
SKIP	(C,X)		Skips the next word within a colon definition. Used with FIND

		and NUMBER.
SP@	(X) --- p	Return the address of the top of the stack. (e.g. 1 2 SP@ @ . . . would type 2 2 1)
SPACE		Output a space character to the selected output device.
SPACES	n ---	Output n spaces to the selected output device. No action for n < 1.
SPACES?	p --- m n	Leaves starting address m and character count n of a message string beginning at address p. n is the length of the message after all trailing spaces have been subtracted starting at address p+63.
STATE	--- q	A variable whose value is set to compile mode or immediate mode.
STK>	(X)	Checks the top of the stack to see if it was marked by the <STK, <STKD, or <STKH verbs; if not, a STACK PARITY ERROR is displayed. See <STK, <STKD, or <STKH for a complete description.
STYPE	q ---	Equivalent to COUNT TYPE.
SWAB	(X) m --- n	Exchange the high and low order bytes of value m.
SWAP	(X) m n --- n m	Exchange the top two stack values.
SYSAVE	m ---	Save the entire dictionary as binary data starting at block m. To restore the dictionary and boot the system, type: m LOAD. This works for physical blocks only.
SYSCOPY	---	Copies blocks 1 thru 44 from disc drive A to drive B.
TABLE	(X) TABLE nnnn	Ex: TABLE XX 4, 5, 6, Define the beginning of a table of words. The values to be entered into the table must follow the definitions of the table. The sequence i nnnn will leave the address of the i-th word on the stack. The index should be 0 <= i < number-of-table-entries. No check is made on the range of i.
TERSE		Brings in the TERSE vocabulary, thereby making all TERSE verbs accessible. (ie. Sets CONTEX,GT to TERSE)

THEN ♦	(CO-,P,X)	Terminate an IF..ELSE..THEN conditional sequence.
TRACK	(X) --- p	Returns the address of a variable containing the track number currently being accessed.
TYPE	m n ---	Send a string of n characters starting at byte address m to terminal.
U!	(X) m n ---	Stores value m into write protected location n and re-protects.
U<	(X) m n --- f	True if unsigned m<n.
U<=	(X) m n ---f	True if unsigned m<n or m=n.
U>	(X) m n --- f	True if unsigned m>n.
U>=	(X) m n --- f	True if unsigned m>n or m=n.
UB!	(X) m n ---	Stores byte value m into write protected location n and re-protects.
UNMAP		Maps in the EDIT, DEBUG and ASM vocabularies. The mapping is done automatically by verbs that need those vocabularies, and maps out the Screen RAM (4000H - 7FFFH).
UNPROT		Makes it possible to write to locations below 4000h in colon definitions.
UNX	(X)	Executes the UNMAP verb and disables interrupts.
UPDATE		Flag the most recently referenced block as updated. The block will subsequently be transferred automatically to disk should its buffer be required for storage of a different block. See FLUSH.
V=	(X) m V= nnnn	Executes a 0 VARIABLE.
VARHERE	(X) --- p	Returns the address of the next available variable location.
VARIABLE	(X) m VARIABLE nnnn	Create a word nnnn which when executed will push the address of a 16 bit variable (initialized to m) onto the stack.

VOCABULARY VOCABULARY nnnn
Create a new vocabulary named nnnn that will append to the current vocabulary. Execution of nnnn will cause it to become the context vocabulary. See CURRENT, CONTEX,GT and DEFINITIONS.

VPTR --- q
A variable similar to DP that points to the next available variable location. Currently starts at E000h and progresses toward SP@. VPTR may be set by the user to a more useful location (i.e. C000h in commercial mode).

WHERE
Output information about the status of TERSE after an error abort. Indicate at least the last word compiled and the last block accessed.

WHILE (X) (C2+,P) f WHILE ---
Test the value on the stack and if FALSE exit out of a BEGIN..WHILE..REPEAT loop. See BEGIN.

XC? (X) --- f
False if Cross-Compiling; otherwise, True.

XOR (X) m n --- q
Bitwise logical exclusive OR of m and n.

XDI (X)
Disables interrupts.

ZERO (X) p ---
Set the word at location p to 0.

[COMPILE] : nnnn ... [COMPILE] pppp ...;
Forces the compilation of the immediate mode verb pppp.

[(P)
Stop compilation. The words following the left bracket in a colon definition are executed, not compiled.

[[(E)
Use: [[..... f]]
Similar to a BEGIN...END except that these are conditionals that may be employed during interpretation, although they are much slower. In conjunction with the verbs [and], they may be used within a colon definition to control compilation, although they are not compiled. These words may be nested.

] (P)
Start compilation. Following words are compiled into the dictionary.

]] (E) f ---
Terminates a conditional interpretation sequence begun by [[.

{
Puts the value of CONTEXT on the return stack and sets the context vocabulary to TERSE. Used with }.

}

Restores the context vocabulary to what it was before {. See {

---------------------end of TERSE Glossary------------------------------