HX-20 HCCS Forth
Quick Reference Guide

With Enhancements by Martin Hepperle, May 5th 2022

c	character (ASCII code), cccc: string	n	16 bit signed number
f	flag (false=0, otherwise true)	u	16 bit unsigned number
addr	16-bit address	nd	signed double number
b	8-bit number	ud	unsigned double number

Enhancements and new words are highlighted by a light yellow background.

Word	Stack	Purpose
`'`	(--- pfa)	Output the PFA of a word. Used in the form `' word`.
`-`	(n1 n2 --- n3)	Subtract n2 from n1.
`!`	(n addr ---)	Store the number n at address addr.
`!CSP`	(---)	Save the stack pointer in variable `CSP`.
`#`	(nd1 --- nd2)	Generate one output character from the signed double number nd1 and return remainder nd2.
`#>`	(nd --- addr u)	Finish output conversion and return string address addr and length u.
`#LAG`	(--- addr n)	Editor: return address and length of the text in `PAD` following the cursor position.
`#LEAD`	(--- addr n)	Editor: return address and length of the text in `PAD` in front of the cursor position.
`#LOCATE`	(--- n1 n2)	Editor: Return column n1 and line number n2 of current cursor position in `PAD`.
`#S`	(nd1 --- nd2)	Generate output characters for the signed double number nd1 until the remainder nd2 is zero.
`(`	(---)	Start a comment terminated by). Must be followed by a space character.
`(.")`	(---)	The run time procedure compiled by `<.">`.
`(;CODE)`	(---)	The run time procedure compiled by `;CODE`.
`(+LOOP)`	(---)	The run time procedure compiled by `+LOOP`. Uses the loop counter on the return stack.
`(ABORT)`	(---)	The run time procedure compiled by `ABORT`. Clears the data and return stack.
`(CLK)`	(---)	Used by `TIME@`. Updates buffered date and time.
`(DO)(---)`		The run time procedure compiled by `DO`. Moves the loop counter to the return stack.
`(FIND)`	(addr1 nfa --- pfa b true) (addr1 addr2 --- false)	Search the dictionary starting at nfa for a word with name starting at addr1. If found, the PFA, length byte and true are left. Otherwise false is returned.
`(LOOP)`	(---)	The runtime procedure compiled by `LOOP`. Uses the loop counter on the return stack.
`(NUMBER)`	(---)	The run time procedure compiled by `NUMBER`.
`(RND)`	(---)	Create a new `SEED` value for the `RND` word.
`*`	(n1 n2 --- n3)	Multiply n2 by n1.
`*/`	(n1 n2 n3 --- n4)	Multiply n1 by n2 and divide by n3 using intermediate double numbers.
`*/MOD`	(n1 n2 n3 --- n4 n5)	Multiply n1 by n2 and divide by n3 using doubles. Return remainder n4 and quotient n5.
`,`	(n ---)	Compile the number n into the current definition.
`.`	(n ---)	Output the number n followed by a space character to the terminal.
`."`	(---)	Compile the string cccc and the cfa of a string output routine. Used in the form `." cccc"`.
`.LINE`	(n ---)	Output the line n from the current screen without trailing spaces.
`.R`	(n1 n2 ---)	Output the number n1 right aligned in a field n2 characters wide.
`/`	(n1 n2 --- n3)	Divide n1 by n2 and return the quotient n3.
`/MOD`	(n1 n2 --- n3 n4)	Divide n1 by n2 and return the remainder n3 and the quotient n4.
`:`	(---)	Begin the definition of a word.
`;`	(---)	End the definition of a word.
`;CODE`	(---)	Stop compilation and switch to assembler code.
`;S`	(---)	Stop interpretation, compiled by `;` in a definition.
`?`	(addr ---)	Print the 16-bit value stored at addr.
`?COMP`	(---)	Throw an error message if not compiling.
`?CSP`	(---)	Restore the stack pointer from variable `CSP`.
`?ERROR`	(f n ---)	Throw error message n if flag f is true.
`?EXEC`	(---)	Throw an error message if not executing.
`?LOADING`	(---)	Throw an error message if not loading from mass storage.
`?PAIRS`	(n1 n2 ---)	Throw an error message if n1 is not equal to n2.
`?STACK`	(---)	Throw an error message if the stack pointer is out of bounds.

Word	Stack	Purpose
?TERM	(--- f)	Return not zero if the DEL key is pressed.
@	(addr --- n)	Fetch the number at address addr.
[	(---)	Begin suspended compilation and execute words enclosed in [] brackets.
[COMPILE]	(---)	Compile the following word instead of executing it.
\0	(---)	A dictionary entry with a name '\0'. See also X.
]	(---)	Terminate suspended compilation.
+	(n1 n2 --- n3)	Return the sum of n2 and n1.
+-	(n1 n2 --- n3)	Transfer the sign of n2 to n1.
+!	(n addr ---)	Add n to the number at addr.
+LOOP	(n ---)	Increment the counter in a DO ... +LOOP by n. Loop back to the DO if it is below the limit.
+ORIGIN	(n --- addr)	Leave the address of the nth byte after the boot parameter area ($6000).
<	(n1 n2 --- f)	Return true if n1 is less than n2. Otherwise return false.
<#	(nd1 --- nd2)	Begin output conversion for a sequence of <# # #S SIGN #> words.
<BUILDS	(---)	Used like : <BUILDS ... DOES> ... ; to define named defining words.
<CMOVE	(addr1 addr2 n ---)	Copy n bytes down from addr1 to addr2.
=	(n1 n2 --- f)	Return true if n1 is equal to n2. Otherwise return false.
>	(n1 n2 --- f)	Return true if n1 is greater than n2. Otherwise return false.
-->	(---)	Continue loading on next screen.
-MOVE	(addr n ---)	Editor: copy C/L bytes from addr to line n of current screen.
>R	(n ---)	Moves the number n from the data stack to the return stack.
0	(--- 0)	Constant with the number 0.
0<	(n --- f)	Return true if n is less than zero. Otherwise false is returned,
0=	(n --- f)	Return true if n is equal to zero. Otherwise false is returned,
0BRANCH	(f ---)	Branch by the offset in the next cell if flag f is true. Compiled by IF, UNTIL, WHILE.
1	(--- 1)	Return the number 1.
1+	(n1 --- n2)	Add the number 1 to the top of the stack.
1LINE	(addr --- f)	Editor: Used internally by FIND and TILL
2	(--- 2)	Constant with the number 2.
2!	(d addr ---)	Store the double number d at addr.
2@	(addr --- d)	Get the double number d from addr.
2+	(n1 --- n2)	Add the number 2 to the top of the stack.
2*	(n1 --- n2)	Multiply the number on top of the stack by two.
2/	(n1 --- n2)	Divide the number on top of the stack by two.
2DROP	(d ---)	Drop the double number d from the top of the stack.
2DUP	(d --- d d)	Duplicate the double number on top of the stack.
3	(--- 3)	Constant with the number 3.
ABORT	(---)	Clear the data and return stacks and return control to the terminal.
ABS	(n --- u)	Return the unsigned absolute value of n.
AGAIN	(---)	Construct an infinite loop BEGIN ... AGAIN
ALLOT	(n ---)	Allocate n bytes by moving the current dictionary pointer.
AND	(n1 n2 --- n3)	Perform a bitwise AND of n1 and n2:
ANOTHER	(---)	Editor: Increment screen number and prepare empty screen for editing.
ASCII	(--- n)	Return the code for an ASCII character. Used in the form ASCII c outside of definitions.
[ASCII]	(--- n)	Compile the code for an ASCII character. Used in the form [ASCII] c inside a definition.
B	(---)	Editor: Move cursor back by the length of PAD.
BACK	(addr ---)	Compiles the backward branch offset from HERE to addr into the dictionary (AGAIN, UNTIL).
BASE	(--- addr)	Variable with the current number base.
BCDBIN	(n1 --- n2)	Convert the positive two-digit BCD number n1 to a binary number n2.
BEEP	(n1 n2 ---)	Sounds a beep with the frequency n1 and duration n2. n1 can be in [1…28] or in [29…56].
BEGIN	(---)	Definition of a BEGIN ... UNTIL, BEGIN ... AGAIN, BEGIN ... WHILE ... REPEAT construct.
BL	(--- c)	Constant with the ASCII code for the blank character.
BLANKS	(addr n ---)	Fill memory starting at addr with n blank characters.
BLK	(n --- addr)	Return the address of block number n.
BRANCH	(---)	Branch unconditionally by the offset in the next cell. Compiled by ELSE, AGAIN, REPEAT.
C	(---)	Editor: Insert text at current cursor position and copy it to PAD.
C!	(b addr ---)	Write an 8-bit number b to the given address addr.
C,	(b ---)	Insert the 8-bit number b into the next dictionary position.
C/L	(--- n)	Constant with the number of characters per line (64).
C@	(addr --- b)	Read an 8-bit number b from the given address.
CASS	(---)	Select external cassette recorder as mass storage. Affects LOAD and SAVE.

Word	Stack	Purpose
`CFA`	(pfa --- cfa)	Convert a words parameter field address to its code field address.
`CLS`	(---)	Clear screen and home cursor.
`CMOVE`	(addr1 addr2 u ---)	Copy u bytes from addr1 to addr2.
`COLD`	(---)	Perform a cold start of the Forth system.
`COMPILE`	(---)	Compile the cfa of the next word into the dictionary.
`CONSTANT`	(n ---)	Define a named constant with value: value `CONSTANT` name.
`CONTEXT`	(--- addr)	Variable containing a pointer to the current vocabulary .
`CONV`	(n --- addr n)	Convert n to a 3 digit string and return its address and length.
`COPY`	(----)	Copy the LCD screen to internal printer, independent of the `PRINT` flag.
`COUNT`	(addr1 --- addr2 n)	Return addr2, length n of a string starting at addr1 with a length byte (addr2 = addr1+1).
`CR`	(---)	Output a carriage return
`CREATE`	(---)	Create a header for a word: `CREATE word`, followed by `COMPILE` or `,`.
`CSP`	(--- addr)	This variable contains the current stack pointer.
`CURRENT`	(--- addr)	Variable with address of address of current definition.
`D`	(n ---)	Editor: delete line n of current screen.
`D.`	(nd ---)	Outputs the signed double number nd.
`D.R`	(nd n ---)	Output the signed double number nd in a field of width n.
`D/L`	(n ---)	Editor: display current screen from line n. Pressing 'N' lists the next line, any other key stops.
`D+`	(nd1 nd2 --- nd3)	Adds two double numbers.
`D+-`	(nd1 n --- nd2)	Transfer the sign of n to the number d1
`DABS`	(nd1 --- nd2)	Return the absolute value of the double number d1
`DATE`	(---)	Output the current date to the terminal.
`DAY`	(--- n)	Return the day of the month in BCD format [1-31].
`DCHAR`	(n b1 b2 b3 b4 b5 b6 ---)	Define user character n with bit rows b1 … b6
`DECIMAL`	(---)	Set the variable `BASE` to 10.
`DEFINITIONS`	(cccc ---)	Set the `CURRENT` vocabulary to cccc.
`DELETE`	(n ---)	Editor: Delete n characters backwards from cursor position.
`DEPTH`	(--- n)	Return the number of cells on the computational stack.
`DIGIT`	(c n1 --- n2 true) (c n1 --- false)	Convert the ASCII character code c with base n1 to its numeric value n2 and a success flag.
`DLITERAL`	(nd --- nd)	Compile the literal value nd to be pushed on the stack.
`DMINUS`	(nd1 --- nd2)	Change the sign of double word nd1 (nd2 = –nd1).
`DO`	(n1 n2 ---)	Compile a `DO` `LOOP` or `DO` `+LOOP` from start n2 to end n1 (exclusive).
`DOES>`	(---)	Define the run time part of a word defined with `<BUILDS` ... `DOES>`.
`DP`	(--- addr)	Variable with the address of the next free position in the dictionary.
`DPL`	(--- addr)	Variable with the number of digits to the right of the decimal point of double numbers.
`DRAW`	(c x1 y1 x2 y2 ---)	Draw a line from $(x,y)_1$ to $(x,y)_2$ with color c (1=set, 0=clear).
`DROP`	(n ---)	Drop one number from the stack.
`DUP`	(n --- n n)	Duplicate the top number on the stack.
`-DUP`	(n1 --- n1) (n1 --- n1 n1)	Duplicate n1 if n1 is not equal to zero. Otherwise leave n1. Same as `?DUP` in other Forth implementations.
`E`	(n ---)	Editor: Erase line n.
`ELSE`	(---)	Define the `ELSE` part of an `IF` ... `ELSE` ... `THEN` construct, executed if f before `IF` is false.
`EMIT`	(c ---)	Output the character having ASCII code c.
`EMPTY-BUFFERS`	(---)	Clear the mass storage buffer.
`ENCLOSE`	(addr1 c --- addr1 n1 n2 n3)	Used by `WORD` to tokenize a text string at addr1 by delimiter c. Return n1: offset to first no-delimiter, n2 offset of first delimiter behind n1, n3 offset of first character behind the string.
`END`	(f ---)	Same as `UNTIL`. Branch back to the matching `BEGIN` of a `BEGIN` ... `END` construct if f is false.
`ENTER`	(---)	Editor: Execute the screen currently edited.
`ERASE`	(addr u ---)	Fill memory starting at addr with u bytes of zero.
`ERROR`	(n ---)	Raise error number n. Vectors through `MESSAGE` and `ABORT`.
`EXECUTE`	(cfa ---)	Execute the definition of the code field at addr (a cfa).
`EXPECT`	(addr n ---)	Read up to n characters from keyboard to buffer at addr and append a null byte. CR terminates.
`F`	(---)	Editor: Find text cccc and move cursor behind it and copy it to `PAD`. Usage: `F cccc`.
`FEED`	(n ---)	Advance the paper by n lines.
`FENCE`	(--- addr)	Variable with the lowest address valid for `FORGET`. Used to protect part of the vocabulary.
`FILE`	(f ---)	Read (f=1) or dump (f=0) screen `SCR` from/to mass storage.
`FILL`	(addr u b ---)	Fill memory starting at addr with u bytes of b.
`FIND`	(--- addr)	Editor: `FIND cccc`.
`-FIND`	(addr --- pfa b true)	Find the word cccc. Usage: `-FIND cccc`. If found, return the PFA , name length b and true,

Word	Stack	Purpose
	(addr --- false)	else false.
FIRST	(--- addr)	Constant with the address of the mass storage buffer ($0710).
FLD	(--- addr)	Variable with the field width for number output.
FORGET	(---)	Remove word cccc from dictionary: FORGET cccc.
FORTH	(---)	Set FORTH as the current vocabulary.
H	(n ---)	Editor: hold line n in PAD.
HALT	(---)	If the flag at $8F is set, then display "Press space" and wait for a key press, otherwise return.
HERE	(--- addr)	Return the address of the next free dictionary entry.
HEX	(---)	Set the variable BASE to 16.
HLD	(--- addr)	Variable with the last character of the current output conversion.
HOLD	(c ---)	Insert the ASCII character c into the current output conversion buffer.
HOME	(---)	Move the cursor to the upper left corner of the display window.
HRS	(--- n)	Return the current hour in BCD format [0-23].
I	(--- n)	Return the inner counter of a DO ... LOOP construct.
ID.	(nfa ---)	Output the name of the word with the given NFA.
IF	(f ---)	Execute the first part of a IF ... THEN or IF ... ELSE ... THEN construct if f is true.
IMMEDIATE	(---)	Does not compile but execute the last defined word.
IN	(--- addr)	Variable with input buffer index. Used by WORD, QUERY and LOAD.
INTERPRET	(---)	The outer interpreter.
IPAD	(n ---)	Editor: Insert line from PAD at line n.
J	(--- n)	Return the next outer counter of a nested DO ... LOOP construct.
KEY	(--- c)	Return the ASCII code of the next key pressed. PF-Keys return $FEnn where nn is $F1…$FA.
L	(---)	Editor: List current screen line by line.
LAST	(--- nfa)	Return the NFA of the most recently defined word in the CURRENT dictionary.
LATEST	(--- nfa)	Return the NFA of the last defined word.
LEAVE	(---)	Leave a DO ... LOOP or DO ... +LOOP construct at the final LOOP or +LOOP.
LFA	(pfa --- lfa)	Convert the parameter field address of a word to its link field address.
LIMIT	(--- addr)	Constant with the address of the first byte above the mass storage buffers ($0B30).
LINE	(n --- addr)	Editor: Get start address addr of line n of the current screen. n in [0…15]
LIST	(n ---)	Outputs the contents of screen n. Sets variable SCR to n and BASE to 10.
LIT	(n ---)	Internally used in a definition. Places the given number on the stack.
LITERAL	(n---) / (--- n)	Compiles the given number n into a definition. At runtime n is placed on the stack.
LOAD	(n ---)	Load and interpret screen n from the mass storage. Tape must already positioned accordingly.
LOOP	(---)	Increment the counter in a DO ... LOOP by one. Loop back to the DO if it is below the limit.
M	(n ---)	Editor: Delete n characters back from cursor and redisplay PAD.
M*	(n1 n2 --- nd)	Multiply n1 by n2 and leave the double number nd.
M/	(nd n1 --- n2 n3)	Divide signed double number nd by n1 and leave the rest n2 and the quotient n3.
M/MOD	(ud1 u2 --- u3 ud4)	Divide the unsigned double ud1 by the unsigned u2 and leave the rest u3 and the quotient ud4.
MASK	(--- addr)	A variable with $8F used to mask 8- to 7-bit characters.
MATCH	(addr1 n1 addr2 n2 --- f offset)	Search n1 bytes from addr1 for a match with the string of n2 characters starting at addr2. Returns a match flag and the offset from addr1of the character following the match.
MAX	(n1 n2 --- max)	Return the larger number of n1 and u2.
MCASS	(---)	Select internal microcassette as mass storage. Affects LOAD and SAVE.
MESSAGE	(n ---)	Output line n of screen 4. If WARNING is 0 only the number is output.
MIN	(n1 n2 --- max)	Return the smaller number of n1 and u2.
MINS	(--- n)	Return the minutes in BCD format [0-59].
MINUS	(n1 --- n2)	Change the sign of the number n1 (n2 = –n1).
MOD	(n1 n2 --- n3)	Return the remainder of the division n1/n2 with the sign of n1.
MON	(---)	Jump into the MONITOR. (only for V1.0 ROMs)
MORE	(---)	Save the current screen and prepare ANOTHER.
MTC	(n1 n2 ---)	Move cursor to column n1 in row n2. Home is (0, 0).
MTH	(--- n)	Return the month number in BCD format [1-12].
N	(---)	Editor: find next occurrence of text in PAD.
NFA	(pfa --- nfa)	Convert a words parameter field address to its name field address.
NUMBER	(addr --- d)	Convert a character string at addr to a signed double number. The string starts with a length byte and an optional minus sign.
OFFSET	(--- addr)	A variable with the mass storage block offset in screens, added to SCR by BLK.
OR	(n1 n2 --- n3)	Return the bitwise OR of n1 and n2.
OUT	(--- addr)	A variable with output buffer index. Incremented by EMIT.
OVER	(n1 n2 --- n1 n2 n1)	Copy the second number on the stack to the top.

Word	Stack	Purpose
P	(n ---)	Editor: place the following text at line n. Usage: 3 P : LINE3 3 . CR ;
PAD	(--- addr)	Return the address of the scratchpad. Usually starts with a length byte.
PFA	(nfa --- pfa)	Convert a words name field address to its parameter field address.
PGET	(x y --- n)	Get the state of a pixel on the LCD screen. I set, n=1, else n=0;
PICK	(n --- n)	Copy the number from level n of the stack to the top.
PLOT	(n x y ---)	Set (n=1) or reset (n=0) the pixel.
PRINT	(n ---)	Switch logging to printer on (n=1) or off (n=0).
PRINT	(n ---)	Switch output off (n=0), to LCD (n=1), printer (n=2) or RS232C (n=4).
PROGRAM	(---)	Ask for a screen number, create an empty screen and enter the Editor.
PSET	(n x y ---)	Set (n=1) or reset (n=0) a pixel on the LCD screen.
QUERY	(---)	Enter up to 80 characters of text from the terminal into the TIB.
QUIT	(n ---)	The outer interpreter loop. Takes return stack pointer n.
R	(--- n)	Return a copy of the cell on top of the return stack.
R#	(--- addr)	Editor: a variable with the current editing position [0…1023].
R>	(--- n)	Move the number n from the return stack to the data stack.
REPEAT	(---)	Used in a definition of a BEGIN ... WHILE ... REPEAT construct.
RND	(n1 --- n2)	Return a new pseudo random number.
ROLL	(n ---)	Rotate the top n cells of the stack by pulling the nth cell and pushing it to the top.
ROT	(n1 n2 n3 --- n2 n3 n1)	Rotate the top three cells by pulling the 3rd cell and pushing it to the top.
RP!	(---)	Reset the return stack pointer to its initial startup value.
RP@	(--- addr)	Return the address of the return stack pointer. Points to one byte below the last stack value.
RPAD	(n ---)	Editor: Replace line n with line from PAD.
S	(n ---)	Editor: insert blank line at line n. The previous line n and subsequent lines are pushed down, line 15 drops out.
S->D	(n --- nd)	Convert the signed number n to the signed double number d.
SAVE	(---)	Save the screen currently edited on mass storage.
SCR	(--- addr)	A variable with the number of the last listed screen.
SECONDS	(--- ud)	Leave the unsigned double number ud with the seconds elapsed since midnight.
SECS	(--- n)	Return the minutes of the hour in BCD format [0-59].
SEED	(---)	Return 0x009A.
SEEK	(n ---)	Seek to tape count n. calls ROM_EB8F.
RSPWR	(n ---)	Apply (f=1) or remove (f=0) power from the RS232C port.
RSPGET	(--- b true) (--- false)	If available, return one byte b from the RS232C port buffer and true otherwise return false.
RSPUT	(b ---)	Output the byte b to the RS232C port. All 8 bits are output.
SHIFT	(n1 b --- n2)	Shift n1 by b bits left (–) or right (+). b should be in +/-[0...15].
SIGN	(n nd --- nd)	Store a '-' character in PAD if n is negative. Used in <# #> conversion sequences.
SMUDGE	(---)	Toggle the smudge bit in the header of the most recently created definition.
SP!	(---)	Reset the stack pointer to its initial start value.
SP@	(--- addr)	Return the current stack pointer.
SPACE	(---)	Output a space character.
SPACES	(n ---)	Output n space characters.
STATE	(--- addr)	A variable with the state of compilation. 0=execution, 1=compilation.
SWAP	(n1 n2 --- n2 n1)	Swap the two number on top of the stack.
T	(n ---)	Editor: type line n and copy it to PAD.
TAPCNT	(n f ---) (0 --- n)	If f=1 set the current value of the tape counter to n, otherwise return the current value.
TEXT	(c ---)	Editor: read one line of text up to delimiter c and hold in PAD. Used by C, F, P, X, TILL.
THEN	(---)	Mark the end of IF ... THEN and IF ... ELSE ... THEN constructs.
TIB	(--- addr)	A variable with the address of the terminal input buffer.
TILL	(---)	Editor: Delete from cursor to end of given text and copy it to PAD. Usage: TILL cccc
TIME	(---)	Output the current time to the terminal.
TIME@	(---)	Internally used by DATE, TIME, get time and date.
TOGGLE	(addr b ---)	Complement (XOR) the word at address addr with the bit pattern b.
TOP	(---)	Editor: Move cursor to top left.
-TRAILING	(addr n1 --- addr n2)	Trim the trailing blanks of the string at addr.
TRAM	(---)	Toggle access to protected lower RAM < 0x80.
TRAVERSE	(addr1 n --- addr2)	Move across the name field of a word. If n=1, addr1 is the nfa of the word, if n=-1 addr1 is the last character in the name. Address addr2 is the other end of the name.
TYPE	(addr u ---)	Type u characters starting at address addr to the terminal.
U*	(u1 u2 --- ud)	Multiply the unsigned integer u1 by u2 and return unsigned double ud.

Word	Stack	Purpose
`U.`	(n ---)	Output the number n as an unsigned integer.
`U/`	(ud u1 --- u2 u3)	Divide the unsigned double ud by unsigned u1. Return unsigned remainder u2 and quotient u3
`U<`	(u1 u2 --- f)	Return true if the unsigned number u1 is less than u2.
`UDP`	(--- addr)	Constant with the address 0x11E for user defined characters. Used by `DCHAR`.
`UNTIL`	(f ---)	Repeat the words in a `BEGIN` ... `UNTIL` construct until f is true.
`USER`	(n ---)	Create a user variable cccc at offset n in the user variable area. Usage: `n USER cccc`.
`VARIABLE`	(---)	Define a variable cccc. Used in the form `VARIABLE cccc`.
`VLIST`	(---)	Output a list of all words in the current `VOCABULARY`. Use the DEL key to stop output.
`VOCABULARY`	(---)	Define a vocabulary. Used in the form `VOCABULARY cccc`.
`VOC-LINK`	(--- addr)	Variable with the address of the vocabulary link field
`WARNING`	(--- addr)	Variable with a flag (0, 1) determining the action on non-system errors.
`WHERE`	(n1 n2 ---)	Editor: output offending position “Scr# Line#” in case of an interpretation error.
`WHILE`	(f ---)	Repeat the `WHILE` ... `REPEAT` part of a `BEGIN` ... `WHILE` ... `REPEAT` construct while f is true.
`WIDTH`	(--- addr)	Variable with the length of a Forth name (`WIDTH @ .` outputs `31`).
`WIND`	(n ---)	Increment the tape counter by n and `SEEK` to the new position.
`WORD`	(c --- addr)	Return the address of the input up to the delimiter c or CR, the buffer starts with a length byte.
`X`	(---)	Editor: find and delete first occurrence of text and copy it to `PAD`.
`XFILE`	(---)	used by `FILE` for file save and load
`XOR`	(n1 n2 --- n3)	Return the bitwise eXclusive OR of n1 and n2.
`YR`	(--- n)	Return the current year in BCD format.

Notes

- When the serial interface is powered up by `1 RSPWR` it uses the cassette buffer as a receive buffer. Avoid any cassette operation until `0 RSPWR` is used to power the port off. This can also interfere if output to the RS232C port is activated with `PRINT`.
- The date and time related words `DAY`, `MTH`, `YR` and `HRS`, `MINS`, `SECS` return a value in packed BCD (Binary Coded Decimal) format. Printing a result of 45 minutes in HEX produces the expected number, e.g. `MINS HEX .` will print `45` but `MINS DECIMAL .` would output 69. The values returned by these words are only updated when `TIME@` is executed. Use `BCDBIN` to convert BCD to binary.

Mass Storage Options

The Forth ROM can use the external or the internal cassette for storing and retrieving screens. You can either manually position the tape or use the `WIND` word with positive and negative offsets. Also `TAPCNT` may be useful.

Example of using the microcassette

`MCASS`	select the micro cassette as mass storage device
`-200 WIND`	or CTRL+PF1 (tape ops), PF4 (rewind), PF5 (quit tape ops)
`0 1 TAPCNT`	or SHIFT + PF1 to reset the tape counter
	… edit screen #1
`SAVE`	store screen #1 on tape
`ANOTHER`	open a new empty screen #2 … edit screen #2
`SAVE`	store screen #2 on tape
`-500 WIND`	CTRL+PF1 (tape ops), PF4 (rewind), PF5 (quit tape ops)
`1 LOAD`	seek forward and load and execute screen #1

Tip when using the internal microcassette: if a `3 LOAD` shows `3 Searching_` and then immediately returns to the prompt, the mass storage has probably been set to the default `CASS`. Execute `MCASS` and try again.

Screen Editor

The variable `SCR` holds screen number for editor.

The word `ENTER` executes the currently edited screen.

The screen editor uses `PAD` as a buffer.

`L`	List the current screen and the current line with the string cursor '#'.
Line oriented commands use a number on the stack	
n `P` cccc	Place text cccc into line n.
n `D`	Delete line n. Subsequent lines move up, a blank line is appended.
n `E`	Erase (clear) line n.
n `S`	Insert a blank line at line n. Line n to 14 move down, line 15 is lost.
n `T`	Type line n and copy it to `PAD`.
n `H`	Copy line n to `PAD`.
n `IPAD`	Insert line from `PAD` at line n. Line n to 14 move down, line 15 is lost.
n `RPAD`	Replace line n with line from `PAD`.
n `D/L`	Display line n, each press of 'N' displays the next line
Cursor # oriented commands	
`C` cccc	Insert text at current position and copy it to `PAD`.
`F` cccc	Find text, move cursor behind it and copy it to `PAD`.
`TILL` cccc	Delete from cursor to end of given text and copy it to `PAD`.
`X` cccc	Find and delete first occurrence of text and copy it to `PAD`.
`N`	Find next occurrence of text in `PAD`.
`B`	Move cursor back by the length of `PAD`.
n `DELETE`	Delete n characters back from cursor.
n `M`	Move cursor by n characters to the right (n>0) or to the left (n<0).

Error MSG codes

0	(reserved)
1	Stack Empty
2	Dictionary Full
[3	Has Incorrect Address Mode (Assembler)]
[4	Isn't unique]
5	Parameter Outside Valid Range
6	Screen Number Out of Range
[7	Stack Full]
[8	Can't Open or Extend File]
[9	Read/Write not Completed]
[10	Can't Redefine End-of-Line]
[11	Can't Divide by Zero]
[12	Undefined Execution Vector]
[13	Branch Too Long (Assembler)]
[14	Incorrect CURRENT Vocabulary]
[15	(reserved)]
[16	(reserved)]
17	Compilation Only
18	Execution Only
19	Conditionals not Paired
20	Definition not Finished
21	In Protected Dictionary
22	Use Only When LOADing
23	Off Current Editing Screen
24	Not in CURRENT Vocabulary
[25	System Memory Clash]

[not used]

		country code							
		0	1	2	3	4	5	6	7
character code	35	#	#	#	#	#	#	#	#
	36	$	$	$	$	¤	$	$	¤
	64	@	@	@	É	É	§	à	É
	91	[	[	[	Æ	Ä	Ä	°	Æ
	92	\	\	\	Ø	Ö	Ö	ç	Ø
	93	]	]	]	Å	Å	Ü	§	Å
	94	^	^	^	Ü	Ü	^	^	Ü
	96	`	`	`	é	é	`	`	é
	123	{	{	{	æ	ä	ä	é	æ
	124	|	|	|	ø	ö	ö	ù	ø
	125	}	}	}	å	å	ü	è	å
	126	~	~	~	ü	ü	ß	¨	ü
country		SE	DE	FR	DK	SE	DE	FR	NO
		ASCII			national				

Figure 1: If you do not want to switch keyboards, you can use the corresponding keys for entering characters like '@', '[' or ']'. These character sets are available in the European versions of the HX-20.

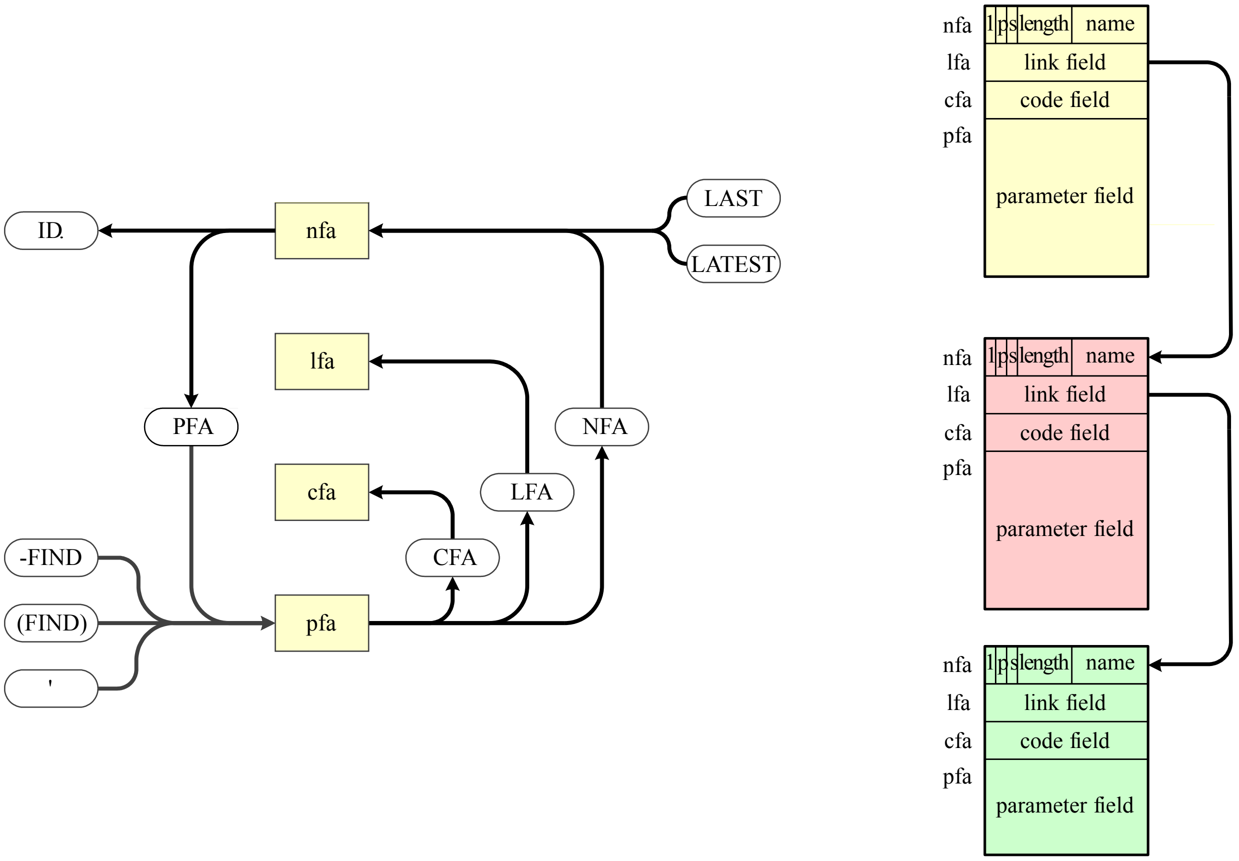


Figure 2: Header addresses and related words as used in figForth. Bits 6 and 5 in the name field length byte denote 'p'recedence (1 = immediate) and 's'mudge; bit 7 is always 1, as is the bit 7 in the last character of name.

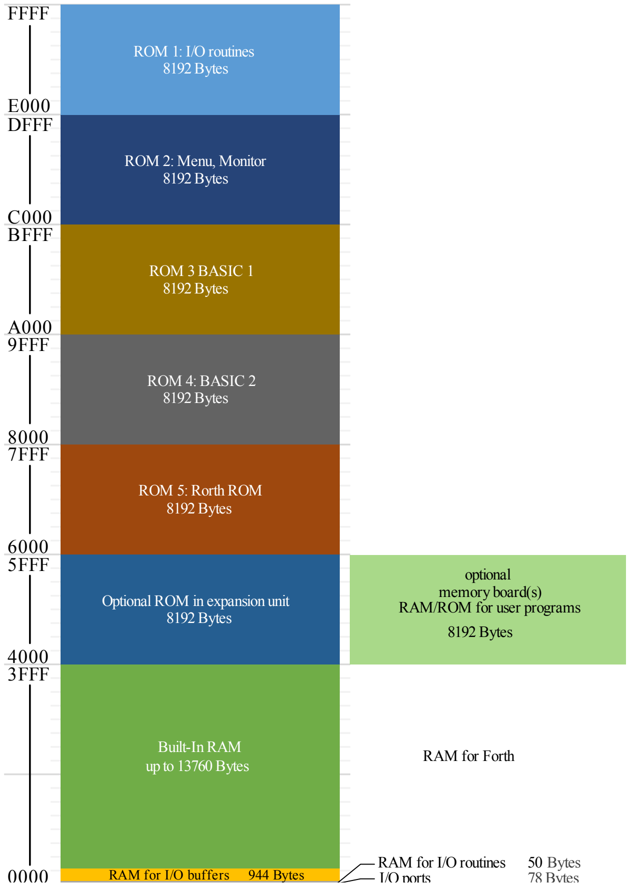
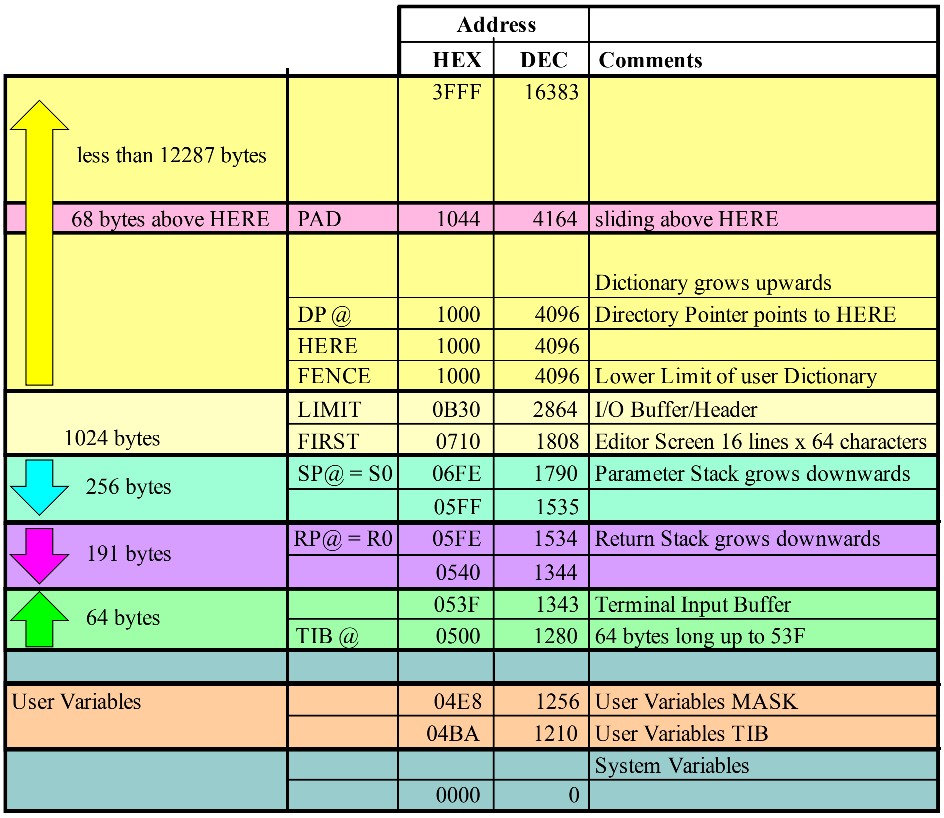


Figure 3: Global memory map of a standard HX-20 system with he Forth-ROM.



		Address		
		HEX	DEC	Comments
↑ less than 12287 bytes		3FFF	16383	
68 bytes above HERE	PAD	1044	4164	sliding above HERE
				Dictionary grows upwards
	DP @	1000	4096	Directory Pointer points to HERE
	HERE	1000	4096	
	FENCE	1000	4096	Lower Limit of user Dictionary
1024 bytes	LIMIT	0B30	2864	I/O Buffer/Header
	FIRST	0710	1808	Editor Screen 16 lines x 64 characters
↓ 256 bytes	SP@ = S0	06FE	1790	Parameter Stack grows downwards
		05FF	1535	
↓ 191 bytes	RP@ = R0	05FE	1534	Return Stack grows downwards
		0540	1344	
↑ 64 bytes		053F	1343	Terminal Input Buffer
	TIB @	0500	1280	64 bytes long up to 53F
User Variables		04E8	1256	User Variables MASK
		04BA	1210	User Variables TIB
				System Variables
		0000	0	

Figure 4: Detailed memory map of the HX-20 figForth.

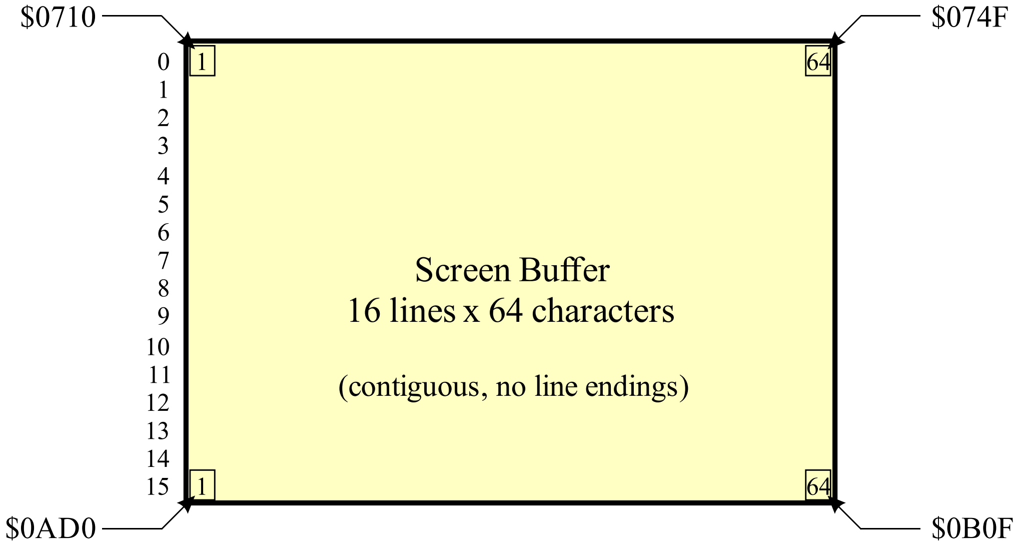


Figure 5: Buffer used for the screen editor and screen I/O of the HX-20 figForth.

Address	User Variables	Offset DEC	Length	Contents HEX	Contents DEC
4E8	MASK	56	2	007F	127
4E6	?	54	2	C752	
4E4	?	52	2	804F	
4E2	?	50	2	0000	0
4E0	HLD	48	2	1040	4160
4DE	R#	46	2	FFE0	65504
4DC	CSP	44	2	06FE	1790
4DA	FLD	42	2	0000	0
4D8	DPL	40	2	FFFF	65535
4D6	BASE	38	2	000A	10
4D4	STATE	36	2	0000	0
4D2	CURRENT	34	2	0D0E	3342
4D0	CONTEXT	32	2	0D0E	3342
4CE	OFFSET	30	2	0000	0
4CC	SCR	28	2	0001	1
4CA	OUT	26	2	0312	0786
4C8	IN	24	2	0003	3
4C6	BLK	22	2	0000	0
4C4	VOC-LINK	20	2	0D0E	3342
4C2	DP	18	2	1000	4096
4C0	FENCE	16	2	1000	4096
4BE	WARNING	14	2	0000	0
4BC	WIDTH	12	2	001F	31
4BA	TIB	10	2	0500	1280
4B8	R0	8	2	05FE	1534
4B6	S0	6	2	06FE	1790

Figure 6: Map of the user variables in the HX-20 figForth, including two system variables R0 and S0.