Appendix 430eForth Commands - Commands are Case Sensitive

Stack Comments: Stack inputs and outputs are shown in the form: (input1 input2 ... -- output1 output2 ...)

Stack Abbreviations of Number Types

Flag - Boolean flag, either 0 or -1 **char** - ASCII character or a byte **n** - 16 bit number **addr** - 16 bit **address** **d** - 32 bit number

Stack Manipulation Commands

?DUP	(n -- n n | 0)	Duplicate top of stack if it is not 0.
DUP	(n – n n)	Duplicate top of stack.
DROP	(n --)	Discard top of stack.
SWAP	(n1 n2 -- n2 n1)	Exchange top two stack items.
OVER	(n1 n2 -- n1 n2 n1)	Make copy of second item on stack.
ROT	(n1 n2 n3 -- n2 n3 n1)	Rotate third item to top.
PICK	(n -- n1)	Zero based, duplicate nth item to top. (e.g. 0 PICK is DUP).
>R	(n --)	Move top item to return stack for temporary storage.
R>	(-- n)	Retrieve top item from return stack.
R@	(-- n)	Copy top of return stack onto stack.
2DUP	(d -- d d)	Duplicate double number on top of stack.
2DROP	(d1 d2 --)	Discard two double numbers on top of stack
DEPTH	(-- n)	Count number of items on stack.

Arithmetic Commands

+	(n1 n2 -- n3)	Add n1 and n2.
-	(n1 n2 -- n3)	Subtract n2 from n1 (n1-n2=n3).
*	(n1 n2 -- n3)	Multiply. n3=n1*n2
/	(n1 n2 -- n3)	Division, signed (n3= n1/n2).
2*	(n -- n*2)	Logic left shift.
2/	(n -- n/2)	Logic right shift.
UM+	(n1 n2 -- nd)	Unsigned addition, double precision result.
UM*	(n1 n2 -- nd)	Unsigned multiply, double precision result.
M*	(n n -- d)	Signed multiply. Return double product.
UM/MOD	(nd n1 -- mod quot)	Unsigned division with double precision dividend.
M/MOD	(d n -- mod quot)	Signed floored divide of double by single. Return mod and quotient.
MOD	(n1 n2 -- mod)	Modulus, signed (remainder of n1/n2).
/MOD	(n1 n2 -- mod quot)	Division with both remainder and quotient.
*/MOD	(n1 n2 n3 -- n4 n5)	Multiply and then divide (n1*n2/n3)
*/	(n1 n2 n3 -- n4)	Like */MOD, but with quotient only.
ABS	(n1 -- n2)	If n1 is negative, n2 is its two's complement.
NEGATE	(n1 -- n2)	Two's complement.
MAX	(n1 n2 -- n3)	n3 is the larger of n1 and n2.
MIN	(n1 n2 -- n3)	n3 is the smaller of n1 and n2.
DNEGATE	(d1 -- d2)	Negate double number. Two's complement.
D+	(d1 d2 -- d3)	Add double numbers.

Logic and Comparison Commands

AND	(n1 n2 -- n3)	Logical bit-wise AND.
OR	(n1 n2 -- n3)	Logical bit-wise OR.
XOR	(n1 n2 -- n3)	Logical bit-wise exclusive OR.
NOT	(n1 -- n2)	Bit-wise one's complement.
0<	(n -- flag)	True if n is negative.
U<	(n1 n2 -- flag)	True if n1 less than n2. Unsigned compare.
<	(n1 n2 -- flag)	True if n1 less than n2.
=	(n1 n2 -- flag)	True if n1 equals n2.
>	(n1 n2 -- flag)	True if n1 greater than n2.

RAM Memory Commands

@	(addr -- n)	Replace addr by number at addr.
C@	(addr -- char)	Fetch least-significant byte only.
!	(n addr --)	Store n at addr.
C!	(char addr --)	Store least-significant byte only.
+!	(n addr --)	Add n to number at addr.
COUNT	(addr1 -- addr+1 char)	Move string count from memory onto stack.
ALLOT	(n --)	Add n bytes to the RAM pointer DP.
HERE	(-- addr)	Address of next available RAM memory location.
PAD	(-- addr)	Address of a scratch area of at least 64 bytes.
TIB	(-- addr)	Address of terminal input buffer.
CMOVE	(addr1 addr2 n --)	Move n bytes starting at memory addr1 to addr2.
FILL	(addr n char --)	Fill n bytes of memory at addr with char.

Flash Memory Commands

I!	(n addr --)	Store n at flash memory addr.
IALLOT	(n --)	Add n bytes to the flash memory pointer CP.
WRITE	(addr1 addr2 --)	Write 128 bytes from RAM memory addr1 to flash memory addr2.
ERASE	(addr --)	Erase an 128 byte page in flash memory at addr.

User Variables

'BOOT	(-- addr)	Contains address of application command to boot.
BASE	(-- addr)	Contains radix for number conversion
CP	(-- addr)	Contains first free address in flash memory
DP	(-- addr)	Contains first free address in RAM memory

Terminal Input-Output Commands

EMIT	(char --)	Display char.
KEY	(-- char)	Get an ASCII character from the keyboard.
?KEY	(-- char -1 | 0)	Return ASCII character from the keyboard and a true flag. Return false flag if no character available.
.	(n --)	Display number n with a trailing blank.
U.	(n --)	Display an unsigned integer with a trailing blank.
.R	(n1 n2 --)	Display signed number n1 right justified in n2 character field.
U.R	(n1 n2 --)	Display unsigned number n1 right justified in n2 character field.
?	(addr --)	Display contents at memory addr.
<#	(--)	Start numeric output string conversion.
#	(n1 -- n2)	Convert next digit of number and add to output string
#S	(n --)	Convert all significant digits in n to output string.
HOLD	(char --)	Add char to output string.
SIGN	(n --)	If n is negative, add a minus sign to the output string.
#>	(xd -- addr n)	Terminate numeric string, leaving addr and count for TYPE.
CR	(--)	Display a new line.
SPACE	(--)	Display a space.
SPACES	(n --)	Display n spaces.
ACCEPT	(addr n --)	Accept n characters into buffer at addr.
CHAR	(-- char)	Parse next command and return its first character.
TYPE	(addr n --)	Display a string of n characters starting at address addr.
BL	(-- 32)	Return ASCII Blank character.
DECIMAL	(--)	Set number base to decimal.
HEX	(--)	Set number base to hexadecimal.

Compiler and Interpreter Commands

:<name>	(--)	Begin a colon definition of <name>.
;	(--)	Terminate execution of a colon definition.
CREATE <name>	(--)	Dictionary entry with no parameter field space reserved.
VARIABLE <name>	(--)	Defines a variable. At run-time, <name> leaves its address.
CONSTANT <name>	(n --)	Defines a constant. At run-time, n is left on the stack.
,	(n --)	Compile n to the dictionary in flash memory
[	(--)	Switch from compilation to interpretation.
]	(--)	Switch from interpretation to compilation.
WORD<text>	(char -- addr)	Get the char delimited string <text> from the input stream and leave as a counted string at addr.
TOKEN	(-- addr)	Parse next word from input stream and copy it to name buffer in RAM at addr.
(comment)	(--)	Ignore comment text.
\ comment	(--)	Ignore comment till end of line.
." <text>"	(--)	Compile <text> message. At run-time display text message.
.(<text>)	(--)	Display <text> from the input stream.
$" <text>"	(-- addr)	Compile <text> message. At run-time return its address.
ABORT" <text>"	(flag --)	Compile <test> message. At run-time display message and abort if flag is true. Otherwise, ignore message and continue.
COLD	(--)	Start eForth system.
QUIT	(--)	Return to interpret mode, clear data and return stacks.
QUERY	(--)	Accept input stream to terminal input buffer.
NUMBER?	(addr -- n -1 | addr 0)	Convert a number string to integer. Push a flag on tos.
EXECUTE	(addr --)	Execute command definition at addr.
@EXECUTE	(addr --)	Execute command definition whose execution address is in addr.
EXIT	(--)	Terminate execution of a colon definition.

Compiler Structure Commands

IF	(flag --)	If flag is zero, branches forward to ELSE or THEN.
ELSE	(--)	Branch forward to THEN.
THEN	(--)	Terminate a IF-ELSE-THEN structure.
FOR	(n --)	Setup loop with n as index. Repeat loop n+1 times.
NEXT	(--)	Decrement loop index by 1, branch back to FOR. Terminate FOR-NEXT loop when index is negative.
AFT	(--)	Branch forward to THEN in a loop to skip the first round
BEGIN	(--)	Start an indefinite loop.
AGAIN	(--)	Branch backward to BEGIN.
UNTIL	(flag --)	Branch backward to BEGIN if flag is false. If flag is true, terminate BEGIN-UNTIL loop.
WHILE	(flag --)	If flag is false, branch forward to terminate BEGIN-WHILE-REPEAT loop. If flag is true, continue execution till REPEAT.
REPEAT	(--)	Resolve WHILE clause. Branch backward to BEGIN.

Utility Commands

' <name>	(-- addr)	Look up <name> in the dictionary. Return execution address.
DUMP	(addr --)	Dump 128 bytes of RAM memory starting from addr.
.S	(--)	Dump the parameter stack.
WORDS	(--)	Display all eForth commands
HI	(--)	Default application. Display sign-on message.
APP!	(addr --)	Build a turnkey system to execute an application at addr.

####