ENGLISH AS A SECOND LANGUAGE FOR FORTH PROGRAMMERS

There is a difference between spelling a word and saying a word. In normal communication we do not ess-pee-ee-ell-ell words. Likewise we do not normally pronounce punctuation. (Period.) Sometimes it is necessary to spell for complete understanding but comprehension is generally easier when natural language is spoken.

A language may also have special signs & symbols which are normally "said", e.g., "&" and "#" are normally pronounced "and" & "number". Spelling often has very little relationship to the pronunciation, e.g., "lb" is "pound" & "cwt" is "hundredweight".

Forth is

"a language of natural English words and signs using machine oriented syntax for command and control of machines".

The so-called natural pronunciation of Forth words given in the Forth-79 and Forth-83 standard documents are mostly spellings. Experience has shown that Forth programs are easier to teach and understand when natural English words are used for the special signs.

An obvious example of this is "#". The standard documents give "sharp" as its natural pronunciation. This is patently wrong. The musical sharp sign is similar to it but different. The semantics of all occurrences of "#" in standard Forth words are connected with numbers. "#" should be said "number" and spelled "number-sign".

The standard specification of "@" is "value at [address]". It make more sense to say "value" than "fetch" when we read it. Likewise "!" reads better as "set", which is what the function is called in other high-level languages. E.g., the body of the definition of "DEFINITIONS"

```
CURRENT @ CONTEXT !
```

reads naturally as

```
current value context set
```

This says What it does, not How it does it. This agrees with the Forth-83 standard document, which says that "descriptive" names are to be preferred to "procedural" names (section 4).

Reflection leads to "add" as the meaning of "+!". A fragment of code

```
0 #LINE !   1 #PAGE +!
```

reads naturally as

    zero number-line set   one number-page add

The so-called natural pronunciation of the apostrophe "'" is given as "tick" in the Forth-83 standard document, ignoring descriptive and procedural names. A better word for this is "address". This also works well in compounds: from the Perry Line-Editor read

    'START   'LINE   'CURSOR   'FIND

as

    address-start address-line address-cursor address-find

"compiled-address" is a better name for "[']". In general say "compiled-name" for "[name]". An exception is "[COMPILE]" which will be considered below.

The function of the dot or period "." in Forth is "print". The Forth word spelled "dot-quote" is used to print a message; the Forth word spelled "ABORT-quote" is used to abort with a message. They should be said "print-message" and "abort-message", which are perfect descriptions.

Forth has three common conventions for names within parentheses.

"(name)" is the default value of a vectored word. E.g.,

    (CR)   (KEY)

"(name)" is used by "name". E.g.,

    (.)

"(name)" is compiled by "name". E.g.,

    (.")   (ABORT")   (LOOP)

Notice that the third case is a subset of the first.

All these uses can be covered by "primitive".

(CR)	primitive CR
(KEY)	primitive key
(.)	primitive print
(.")	primitive print message
(ABORT")	primitive abort message
(LOOP)	primitive loop

"," is used to lay down values in the dictionary, and we say "lay down" for this function.

Just as Forth is said to have been discovered, not invented, so the foregoing words were discovered in the order given. Having come so far, we would like to go the rest of the way.

"[" is used to initiate interpretation, but "INTERPRET" is a word in the controlled word-set. We pronounce it "evaluate".

A stumbling stone in learning Forth is the difference between "]", "COMPILE", and "[COMPILE]".

If we think of "]" as "construct" we have a way to distinquish "]" from the other two. "COMPILE" will "compile" a defined word into the dictionary; "]" will do whatever is necessary to "construct" the dictionary, including defined words, literal values, logical structures, and anything else.

We now face the problem of "[COMPILE]". The use of brackets is out of the ordinary. This does not mean "compiled compile". I propose that following the English examples of "lb" and "cwt" that it be pronounced "compiled in-line". I also propose that "[IN-LINE]" be adopted as a synonym of "[COMPILE]". When teaching (advanced) Forth use "[IN-LINE]" to explain the function initially; later "[COMPILE]" can be given as a synonym.

Some examples to show how this hangs together.

```
: ASCII  ( -- c ) BL WORD COUNT 1- ABORT" ?"
   STATE @ IF [COMPILE] LITERAL THEN ; IMMEDIATE
```

"Define ASCII BL word count one-minus abort-message question-mark state value if compiled-inline literal then. Immediate."

Note that punctuation was not pronounced. Punctuation can be spelled when necessary for comprehension.

From the preceding paragraph we see how "(" should be pronounced, i.e., "note". Likewise ".(" is "print-note".

From an integer-ascii conversion definition:

```
... [ ASCII A 10 - ] LITERAL ...
```

"... evaluate ascii A 10 minus construct literal ..."

The prefix "C" is used in Forth to show byte-related operations. Just as "cwt" is pronounced "hundredweight" so "C@", "C!", and "C," have the pronunciation "byte-value", "byte-set", and "byte-laydown".

We can say "define" for ":" as we did above. The ";" can be silent punctuation, or we can use the vernacular "already" until some-one has a better suggestion.

Here is a summary of the proposed pronunciations. Can you think of any others?

#	number
@	value
!	set
+!	add
'	address
[']	compiled address
.	print
."	print message
ABORT"	abort message
(name)	primitive name
,	lay down
[	evaluate
]	construct
[COMPILE]	compiled in-line
(	note
.(	print note
C@	byte value
C!	byte set
C,	byte lay down
:	define
;	already

FINIS