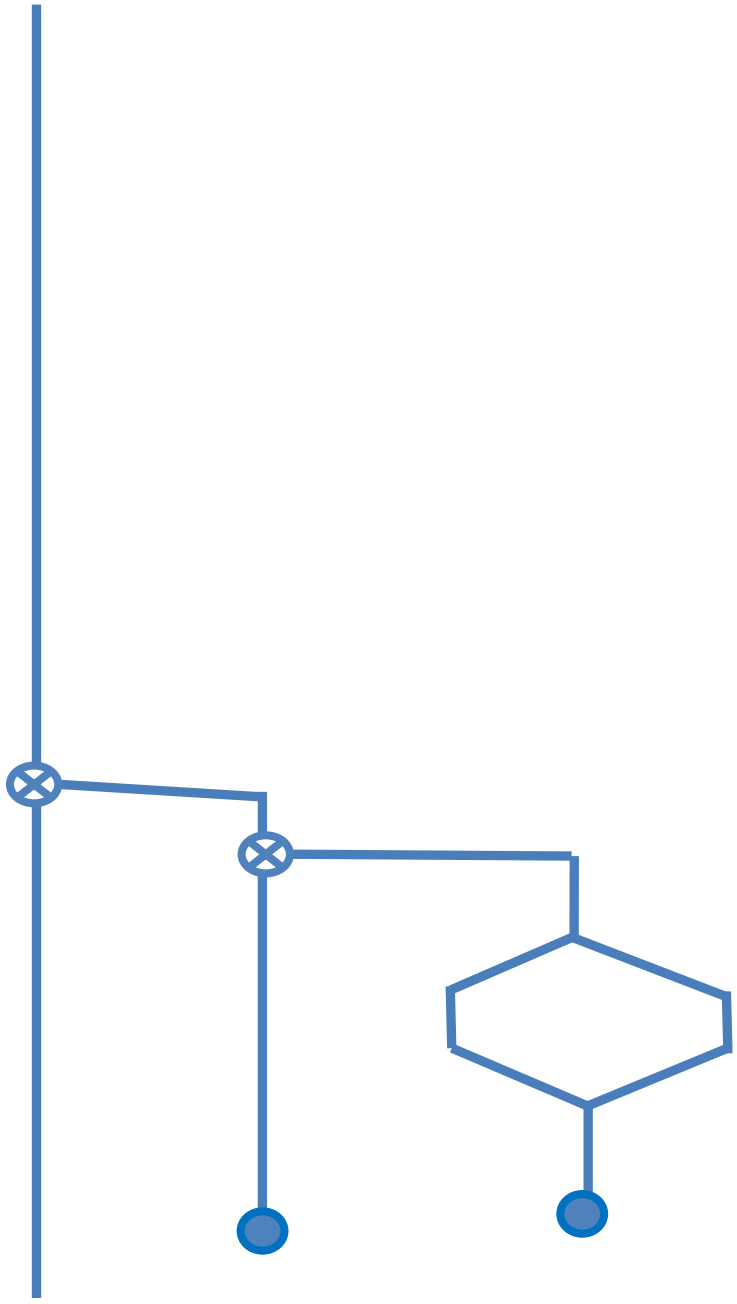


VALUE, TO and +TO

SVFIG

Aug. 26, 2023

Bill Ragsdale

The Basics

A word defined by VALUE yields its stored value as does a CONSTANT.

However, its value may be changed by prefacing with TO.

The value may be incremented by prefacing with +TO.

Today

We will examine the development of VALUE, TO and +TO.

First, the classical method in Forth 79 onward.

A high-level implementation.

And the innovation in Win32Forth by Andrew McKewan.

History on VALUE

Was proposed casually by Chuck at the 1979 Standards Meeting in Catalina upon complaints on the extra step in reading from variables.

We all got excited.

Published by Michael McNeil in 1980 FORML Proceedings and Paul Bartoldi in Forth Dimensions.

Forth 79 Background, I

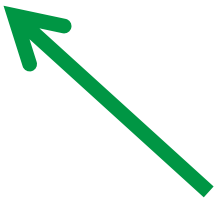
```
0x777 VALUE A-Value

<header-A-Value>  DOVALUE  0x777      \ as compiled

: VALUE
    create ,                           \ header and storage
    ['] DOVALUE >BODY lastcfa  @ ! ;   \ rewrite the cfa

CODE DOVALUE
     from W+cell locate following parameter field
     add TOS space
     load TOS from that address
     return to interpreter ;c
```

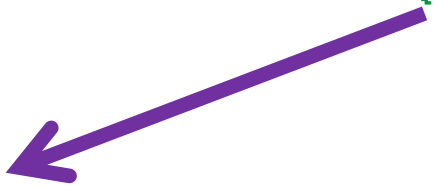
Forth 79 Background, II

```
<compiled code>
       . . .  (TO)  <pfa-A-Value>  . . .

: TO
   '  ['] DOVALUE >BODY over @ =    \ get cfa and validate
   if  state @                      \ has value's compiled address
       if  COMPILE (TO) >body ,     \ compiles the VALUES's pfa
          else  >body  !  Then      \ or store now
     else  abort" Not a value
    then    ;   immediate

CODE (TO)
   get address pointed by [IP]    \ the values's storage address
   copy TOS to address
   increment IP by cell
   delete TOS
   to interpreter c;
```

As Compiled

```
0  VALUE  A-Value

<header-A-Value>   DOVALUE  <storage cell>

: A-Demo   212  to  A-Value  ;

<header-A-Demo>  DOCOLON  LIT 212  (TO) <address>  UNNEST


CODE (TO)
   get address pointed by [IP]  \ values's storage address
   copy TOS to address
   increment IP by cell
   delete TOS
   to interpreter c;
```





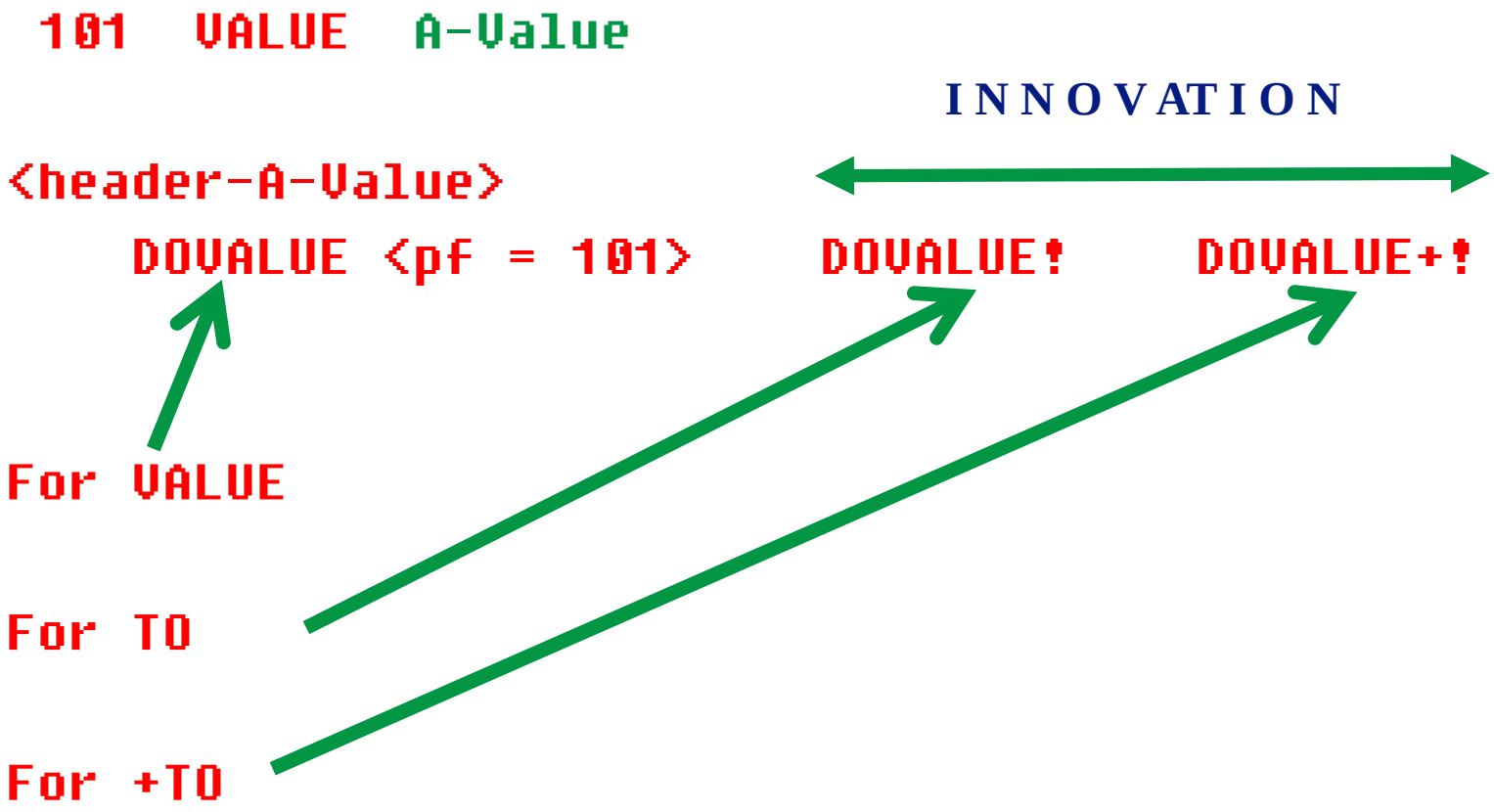
Forth 79 Background, III

```
<interpreting>
   . . 0x1234 TO A-Value  . . .     \ immediately stores


: A-Demo    0x1234  TO  A-Value ;

<header-A-Demo>
  DODOLON  LIT  0x1234  (TO)   <pfa-A-Value>  UNNEST

: B-Demo    0x5678  +TO  A-Value ;

<header-B-Demo>
  DODOLON  LIT  0x5678  (+TO)  <pfa-A-Value>  UNNEST
```

My High-level Version
probably portable

```
: VALUE    create ,   does>  @  ;

0  value (DOVALUE)  \ to locate the cfa for DOVALUE

: (TO)   r> dup cell+ >r  @ ! ;    \ run-time to do TO.

: TO
   '  [']  (DOVALUE) @ over @ =    \ get cfa and validate
   if  state @                      \ has value's compiled address
       if  compile (TO) >body ,     \ compiles the VALUES's pfa
          else  >body  !  then
     else  abort" Not a value
    then    ;   immediate

For +TO  replace ! With +! two places.
```

Win32Forth Innovation



Win32Forth Method

```
<header>
DOVALUE <storage cell> DOVALUE!  DOVALUE+!

DOVALUE Reads from next cell

DOVALUE! Stores into prior cell.

DOVALUE+!  Increments into 2nd prior cell.

 TO compiles DOVALUE!

+TO compiles DOVALUE+!
```

Win32Forth Compiled

```
<header-A-Value>
DOVALUE <storage cell> DOVALUE!  DOVALUE+!

: Demo-1  A-VALUE    ;
<header>  DOCOLON   <add-DOVALUE>  UNNEST

: Demo-2  222 TO  A-Value ;
<header>  DOCOLON  LIT 222  <add-DOVALUE!>  UNNEST

: Demo-3  333 +TO A-Value ;
<header>  DOCOLON  LIT 333  <add-DOVALUE+!> UNNEST
```

Advantages Of Win32Forth

- The value definition is two cells larger, but . . .
- Each value usage is one cell smaller.
- Fast as only one in-line cfa is interpreted.
- Facilitates decompiling as there is a specific cfa for VALUE, TO and +TO.

Conclusions

- VALUES seem to be increasing in usage over VARIABLES.
- Meta-compiling can be quite challenging.
- I’m unsure of the complexity to meta-compile for direct threaded and subroutine threaded versions.

Win32Forth Cosmology

The Complete Forth Textbook