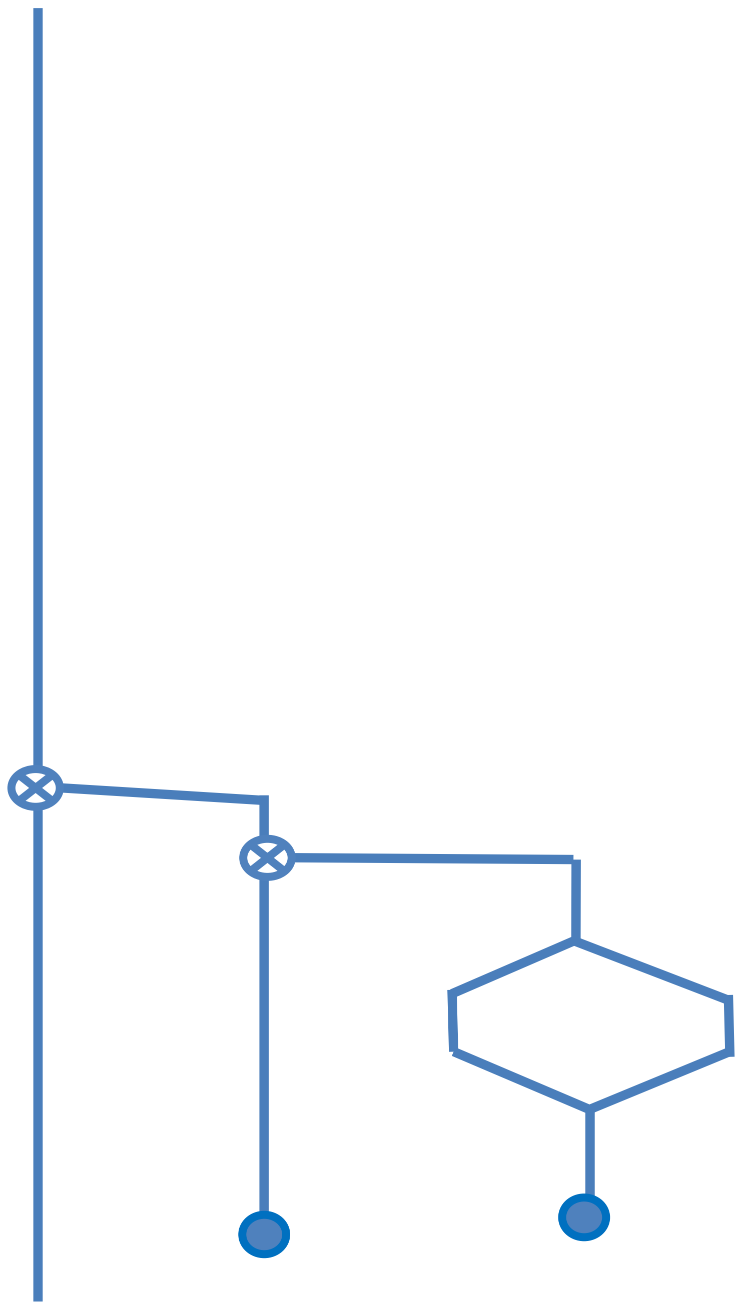


CREATE DOES>

SVFIG

JuLY 22, 2023

Bill Ragsdale

Today

We will examine the history and use of CREATE DOES>.

1968 <BUILDS DOES>

1982 CREATE DOES>

2023 CREATE DOES> In Win32Forth

What?

CREATE DOES>   creates words that create words.

Can be used to create CONSTANT, VARIABLE, ARRAYS.   [Or a complete Forth.]

And data-base fields.

And assembler op-codes.

Generally to create words with a common similarity.

How?

PARENT will create a family of child words that share a common execution but have individual parameters.

```
: PARENT
   CREATE    ,              \ ← the creator portion
   DOES>    @   DROP ;      \ ← the run-time portion


0x1234 PARENT CHILD   \ an example defined word
```

Examples

```
: CONSTANT  CREATE ,  DOES>  @ ;
0x10 CONSTANT HEX-BASE


: OP-CODE   CREATE  ,  DOES>  @ , ;
0x5F OP-CODE  CLC,    0xBB OP-CODE  PUSH,


: FIELD  CREATE OVER , +  DOES @ ORIGIN +  ;
0 20 FIELD NAME   CELL FIELD AGE   CELL FIELD WEIGHT
DROP
```

History: 1960s

The form created by Charles Moore in the 1960s and carried through until 1982.

```
: PARENT   <BUILDS  ,   DOES>  @   DROP  ;

0x1234  PARENT  CHILD


<PARENT-header> <docol> <BUILDS , DOES>  @  DROP  ENDCODE


<CHILD-header)  <dodoes> <pointer>  0x1234
```

History: 1960s

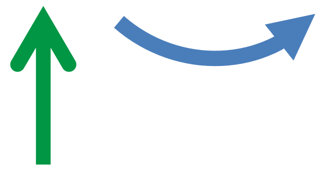
```
: <BUILDS   0  CONSTANT  ;  \ Create header and
                            \ one parameter for DOES>

: DOES>  \ Rewrite PFA with calling hi-level code address
         \ Rewrite CFA pointing to this dodoes code.
     R>  LATEST  PFA  !
     ;CODE                                  \ dodoes follows
     IP 1+ LDA,  PHA,  IP LDA,  PHA,  \ begin Forth nesting
     2 # LDY,  W )Y LDA,  IP STA,      \ fetch address low byte
     INY,  W )Y LDA,  IP 1+ STA,       \ then high byte to W
     CLC,  W LDA,  4 # ADC,  PHA,      \ address of code into IP
     W 1+ LDA,  00 # ADC,  PUSH JMP,  \ interpret in PARENT word
```

History: 1982 new DOES>

```
PARENT   CREATE   ,   DOES>   @   ,   ;
0x1234   PARENT CHILD

<PARENT-header> <docol>  CREATE   ,
     (DODOES>)  here+cell JSR, DODOES   @  DROP  ENDCODE
```



```
<CHILD-header>  <cfa>  0x1234
```

DOES> creates a ‘fake’ code word: here+cell JSR dodoes

When CHILD executes the JSR, DODOES locates of the in-line code pfa address of CHILD (holding 0x1234) placed on the stack.

Advantages

- CREATE replaces <BUILDS
- Uses simulated in-line code for interpretation.
- The extra pointer in the child word is not needed.
- Tick (‘) properly returns the parameter address in CHILD

In Win32Forth

All code must be in the CODE memory allocation. Split headers.

Therefor the simulated in-line code can't be used.

The answer is to place support in the CODE memory, specific to each CREATE DOES> defining word. Used to locate the run-time portion for the child word.

A common DODOES is used.

(DODOES>) Is The Key

- Creates an unnamed code fragment:
  <proto-dodoes>.
- In <proto-dodoes> places MOV W, <execution code in PARENT>
- Compile a long relative jump to the existing DOCOL.
- DOCOL: Places the CHILD's parameter address on the stack and directs execution to address in W, high-level code in the PARENT.

W32F How

```
<PARENT-header>  CREATE   ,
        (DODOES>) <proto-does>  @  DROP  UNNEST

<CHILD-header> <proto-does>  0x1234



 <proto-does>
    C7 C1   MOV W,  <addr after <proto-does> \ destination
   FC E9   JMP dodoes
dodoes:
        53          push  TOS           \ make room on stack
        89 75 FC    mov   -4 [RP], IP   \ push IP to return stack
        8B F1       mov   IP, ecx       \ new IP
        8D 58 04    lea   TOS, 4 [W]    \ push address of parameter field
        8B 46 FC    mov   W, -4 [IP]    \ x on return stack
        ED 04       sub   RP, # 4       \ confirm space on return stack
        FF 20       exec  c;
```

Summary

The New DOES> was introduced by Chuck at the memorable 1982 FORML conference. We were immediately astonished.

Another approach uses ;CODE. Maybe we'll discuss this another time.

CREATE, DOES> and ;CODE could be used as the core of a meta-compiler. Now they are just adjuncts.