

# RfD: Enhanced local variable syntax

---

*Source:* <http://newsgroups.derkeiler.com/Archive/Comp/comp.lang.forth/2007-07/msg00154.html>

---

- *From:* Peter Knaggs <[pknaggs@xxxxxxxxxxxxxxxxxxxx](mailto:pknaggs@xxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Wed, 11 Jul 2007 17:55:06 +0100
- 

Stephen Pelc – 7 June 2007

20070607 Wordsmithing. Corrected reference implementation.  
20060822 Added explanatory text.  
Corrected reference implementation.  
Updated ambiguous conditions.

## Problem

=====

- 1) The current LOCALS| ... | notation explicitly forces all locals to be initialised from the data stack.
- 2) The current LOCALS| ... | notation defines locals in reverse order to the normal stack notation.
- 3) When programming large applications, especially those interfacing with a host operating system, there is a frequent need for temporary buffers.
- 4) Current implementations show that creation and destruction of local buffers are much faster than using ALLOCATE (14.6.1.0707) and FREE (14.6.1.1605).

## Solution

=====

Base version

-----

The following syntax for local arguments and local variables is proposed. The sequence:

```
{ ni1 ni2 ... | lv1 lv2 ... -- o1 o2 ... }
```

defines local arguments, local variables, and outputs. The local arguments are automatically initialised from the data stack on entry, the rightmost being taken from the top of the data stack. Local arguments and local variables can be referenced by name within the word during compilation. The output names are dummies to allow a complete stack comment to be generated.

The items between { and | are local arguments.

The items between | and -- are local variables or buffers.

The items between -- and } are outputs for formal comments only.

The outputs are provided in the notation so that complete stack comments can be produced. However, all text between `--` and `}` is ignored. The facility is there to permit the notation to form a complete stack comment. This eases documentation and current users of the notation like this facility.

Local arguments and variables return their values when referenced, and must be preceded by `TO` to perform a store.

Local buffers may be defined in the form:

```
arr[ <expr> ]
```

Any name ending in the `'[` character will be treated as a buffer, the expression up to the terminating `']` will be interpreted to provide the size of the buffer. Local buffers only return their base address, all operators such as `TO` generate an ambiguous condition.

In the example below, `a` and `b` are local arguments, `a+b` and `a*b` are local variables, and `arr[` is a 10 byte local buffer.

```
: foo { a b | a+b a*b arr[ 10 ] -- }
a b + to a+b
a b * to a*b
cr a+b . a*b .
arr[ 10 erase
s" Hello" arr[ swap cmove
;
```

### Local types

Some current Forth systems use indicators to define local variables of sizes other than a cell. It is proposed that any name ending in a `:` (colon) be reserved for this use.

```
: foo { a b | F: f1 F: f2 -- c }
...
;
```

### Discussion

The `'|` (ASCII 0x7C) character is widely used as the separator between local arguments and local variables. Other characters accepted in current Forth implementations are `'\` (ASCII 0x5C) and `'|` (ASCII 0xA6).. Since the ANS standard is defined in terms of 7 bit ASCII, and with regard to internationalisation, we propose only to consider the `'|` and `'\` characters further. Only recognition of the `'|` separator is mandatory.

The use of local types is contentious as they only become useful

if TO is available for these. In practice, some current systems permit TO to be used with floats (children of FVALUE) and other data types. Such systems often provide additional operators such as +TO (add from stack to item) for children of VALUE and FVALUE. Standardisation of operators with (for example) floats needs to be done before the local types extension can be incorporated into Forth200x. Apart from forcing allocation of buffer space, no additional functionality is provided by local types that cannot be obtained using local buffers. More preparatory standardisation needs to be done before local types can be standardised.

Apart from { (brace) itself, the proposal introduces one new word BUILDLV. The definition of this word is designed for future enhancements, e.g. more local data types, without having to introduce more new words.

It has been noted that one widely used implementation uses brace for multi line comments. However, inspection of the vendor's code shows that this use only occurs during interpretation. The interpretation semantics of brace in this proposal are undefined in order for that implementation to coexist with this proposal.

#### Proposal

=====

13.3 and 13.4

Add "and BUILDLV" where (LOCAL) is referenced.

13.6.2.xxxx {  
brace LOCAL EXT

Interpretation: Interpretation semantics for this word are undefined.

Compilation:

( "<spaces>arg1" ... "<spaces>argn" | "<spaces>lv1" ... "<spaces>lvn" --- )

Create up to eight local arguments by repeatedly skipping leading spaces, parsing arg, and executing 13.6.2.yyyy BUILDLV. The list of local arguments to be defined is terminated by "|", "---" or "}". Append the run-time semantics for local arguments given below to the current definition. If a space delimited '|' is encountered, create up to eight local variables or buffers by repeatedly skipping leading spaces, parsing lv, and executing 13.6.2.yyyy BUILDLV. The list of local variables and buffers to be defined is terminated by "---" or "}". Append the run-time semantics for local variables and local buffers given below to the current definition. If "---" has been encountered, further text between "---" and } is ignored.

Local buffers have names that end in the '[' character. They define their size by parsing the text string up to the next ']' character,

## RfD: Enhanced local variable syntax

and passing that string to 7.6.1.1360 EVALUATE to obtain the size of the storage in address units.

Local argument run-time: ( x1 ... xn -- )

Local variable run-time: ( -- )

Local buffer run-time: ( -- )

Initialize up to eight local arguments as described in 13.6.2.yyyy BUILDLV. Local argument arg1 is initialized with x1, arg2 with x2 up to argn from xn, which is on the top of the data stack. When invoked, each local argument will return its value. The value of a local argument may be changed using 13.6.1.2295 TO.

Initialize up to eight local variables or local buffers as described in 13.6.2.yyyy BUILDLV. The initial contents of local variables and local buffers are undefined. When invoked, each local variable returns its value. The value of a local variable may be changed using 13.6.1.2295 TO. The size of a local variable is a cell. When invoked, each local buffer will return its address. The user may make no assumption about the order and contiguity of local variables and buffers in memory.

Ambiguous conditions:

- a) The { ... } text extends over more than one line.
- b) The expression for local buffer size does not return a single cell.
- c) { ... } is declared more than once in a word.
- d) Parsing units '|', '|', '--' and '}' are not white space delimited.

13.6.2.yyyy BUILDLV  
build-1-v LOCAL EXT

Interpretation: Interpretation semantics for this word are undefined.

Execution: ( c-addr u +n mode -- )

When executed during compilation, BUILDLV passes a message to the system identifying a new local argument whose definition name is given by the string of characters identified by c-addr u. The size of the data item is given by +n address units, and the mode identifies the construction required as follows:

0 – finish construction of initialisation and data storage allocation code. C-addr and u are ignored. +n is 0 (other values are reserved for future use).

1 – identify a local argument, +n = cell

2 – identify a local variable, +n = cell

3 – identify a local buffer, +n = storage required.

4+ – reserved for future use

-ve – implementation specific values

The result of executing BUILDLV during compilation of a definition is to create a set of named local arguments, variables and/or

## RfD: Enhanced local variable syntax

buffers, each of which is a definition name, that only have execution semantics within the scope of that definition's source.

local argument execution: ( -- x )

Push the local argument's value, x, onto the stack. The local argument's value is initialized as described in 13.6.2.xxxx { and may be changed by preceding the local argument's name with TO.

local variable execution: ( -- x )

Push the local variable's value, x, onto the stack. The local variable is not initialised. The local variable's value may be changed by preceding the local variable's name with TO.

local buffer execution: ( -- a-addr )

Push the local buffer's address, a-addr, onto the stack. The address is aligned as in 3.3.3.1. The contents of the buffer are not initialised.

Note: This word does not have special compilation semantics in the usual sense because it provides access to a system capability for use by other user-defined words that do have them. However, the locals facility as a whole and the sequence of messages passed defines specific usage rules with semantic implications that are described in detail in section 13.3.3 Processing locals.

Note: This word is not intended for direct use in a definition to declare that definition's locals. It is instead used by system or user compiling words. These compiling words in turn define their own syntax, and may be used directly in definitions to declare locals. In this context, the syntax for BUILDLV is defined in terms of a sequence of compile-time messages and is described in detail in section 13.3.3 Processing locals.

Note: The LOCAL EXT word set modifies the syntax and semantics of 6.2.2295 TO as defined in the Core Extensions word set.

See: 3.4 The Forth text interpreter

Ambiguous conditions:

a local argument, variable or buffer is executed while in interpretation state.

Reference implementation

=====

(currently untested)

: TOKEN \ -- caddr u

\ Get the next space delimited token from the input stream.

BL PARSE

;

```

: LTERM? \ caddr u --- flag
\ Return true if the string caddr/u is "---" or "}"
2DUP S" ---" COMPARE 0= >R
S" }" COMPARE 0= R> OR
;

: LBSIZE \ -- +n
\ Parse up to the terminating ']' and EVALUATE the expression
\ not including the terminating ']'.
POSTPONE [ [CHAR] ] PARSE EVALUATE ]
;

: LB? \ caddr u --- flag
\ Return true if the last character of the string is '['.
+ 1 CHARS - C@ [CHAR] [ =
;

: LSEP? \ caddr u --- flag
\ Return true if the string caddr/u is the separator between
\ local arguments and local variables or buffers.
2DUP S" |" COMPARE 0= >R
S" \" COMPARE 0= R> OR
;

: { ( --- )
0 >R \ indicate arguments
BEGIN
TOKEN 2DUP LTERM? 0=
WHILE \ --- caddr len
2DUP LSEP? IF \ if '|'
R> DROP 1 >R \ change to vars and buffers
ELSE
R@ 0= IF \ argument?
CELL 1
ELSE \ variable or buffer
LB?
IF LBSIZE 3 ELSE CELL 2 THEN
THEN
BUILDLV
THEN
REPEAT
BEGIN
S" }" COMPARE
WHILE
TOKEN
REPEAT
0 0 0 0 BUILDLV
R> DROP
; IMMEDIATE
.

```