13. The optional Locals word set

13.1 Introduction

See: Annex A.13 The Locals Word Set

13.2 Additional terms and notation

None.

13.3 Additional usage requirements

See: A.13.3 Additional usage requirements

13.3.1 Locals

A local is a data object whose execution semantics shall return its value, whose scope shall be limited to the definition in which it is declared, and whose use in a definition shall not preclude reentrancy or recursion.

13.3.2 Environmental queries

Append table 13.1 to table 3.5.

See: 3.2.6 Environmental queries

Table 13.1 - Environmental query strings

String	Value data type	Constant?	Meaning
#LOCALS	n	yes	maximum number of local variables in a definition
LOCALS	flag	no	locals word set present
LOCALS-EXT	flag	no	locals extensions word set present

13.3.3 Processing locals

To support the locals word set, a system shall provide a mechanism to receive the messages defined by (LOCAL) and respond as described here.

During the compilation of a definition after : (colon), :NONAME, or DOES>, a program may begin sending local identifier messages to the system. The process shall begin when the first message is sent. The process shall end when the **last local** message is sent. The system shall keep track of the names, order, and number of identifiers contained in the complete sequence.

13.3.3.1 Compilation semantics

The system, upon receipt of a sequence of local-identifier messages, shall take the following actions at compile time:

a) Create temporary dictionary entries for each of the identifiers passed to (LOCAL), such that each identifier will behave as a local. These temporary dictionary entries shall vanish at the end of the definition, denoted by ; (semicolon), ;CODE, or DOES>. The system need not maintain these identifiers in the same way it does other dictionary entries as long as they can be found by normal dictionary searching processes. Furthermore, if the Search-Order word set is present, local identifiers shall always be searched before any of the word lists in any definable search order, and none of the Search-Order words shall change the locals' privileged position in the search order. Local identifiers may reside in mass storage.

b) For each identifier passed to (LOCAL), the system shall generate an appropriate code sequence that does the following at execution time:

1. Allocate a storage resource adequate to contain the value of a local. The storage shall be allocated in a way that does not preclude re-entrancy or recursion in the definition using the local.
2. Initialize the value using the top item on the data stack. If more than one local is declared, the top item on the stack shall be moved into the first local identified, the next item shall be moved into the second, and so on.

The storage resource may be the return stack or may be implemented in other ways, such as in registers. The storage resource shall not be the data stack. Use of locals shall not restrict use of the data stack before or after the point of declaration.

c) Arrange that any of the legitimate methods of terminating execution of a definition, specifically ; (semicolon), ;CODE, DOES> or EXIT, will release the storage resource allocated for the locals, if any, declared in that definition. ABORT shall release all local storage resources, and CATCH / THROW (if implemented) shall release such resources for all definitions whose execution is being terminated.

d) Separate sets of locals may be declared in defining words before DOES> for use by the defining word, and after DOES> for use by the word defined.

A system implementing the Locals word set shall support the declaration of at least eight locals in a definition.

13.3.3.2 Syntax restrictions

Immediate words in a program may use (LOCAL) to implement syntaxes for local declarations with the following restrictions:

a) A program shall not compile any executable code into the current definition between the time (LOCAL) is executed to identify the first local for that definition and the time of sending the single required **last local** message;

b) The position in program source at which the sequence of (LOCAL) messages is sent, referred to here as the point at which locals are declared, shall not lie within the scope of any control structure;

c) Locals shall not be declared until values previously placed on the return stack within the definition have been removed;

d) After a definition's locals have been declared, a program may place data on the return stack. However, if this is done, locals shall not be accessed until those values have been removed from the return stack;

e) Words that return execution tokens, such as ' (tick), ['], or FIND, shall not be used with local names;

f) A program that declares more than eight locals in a single definition has an environmental dependency;

g) Locals may be accessed or updated within control structures, including do-loops;

h) Local names shall not be referenced by POSTPONE and [COMPILE].

`See: 3.4 The Forth text interpreter`

13.4 Additional documentation requirements

13.4.1 System documentation

13.4.1.1 Implementation-defined options

- maximum number of locals in a definition (13.3.3 Processing locals, 13.6.2.1795 LOCALS|).

13.4.1.2 Ambiguous conditions

- executing a named local while in interpretation state (13.6.1.0086 (LOCAL));
- name not defined by VALUE or LOCAL (13.6.1.2295 TO).

13.4.1.3 Other system documentation

- no additional requirements.

13.4.2 Program documentation

13.4.2.1 Environmental dependencies

- declaring more than eight locals in a single definition (13.3.3 Processing locals).

13.4.2.2 Other program documentation

- no additional requirements.

13.5 Compliance and labeling

13.5.1 ANS Forth systems

The phrase **Providing the Locals word set** shall be appended to the label of any Standard System that provides all of the Locals word set.

The phrase **Providing name(s) from the Locals Extensions word set** shall be appended to the label of any Standard System that provides portions of the Locals Extensions word set.

The phrase **Providing the Locals Extensions word set** shall be appended to the label of any Standard System that provides all of the Locals and Locals Extensions word sets.

13.5.2 ANS Forth programs

The phrase **Requiring the Locals word set** shall be appended to the label of Standard Programs that require the system to provide the Locals word set.

The phrase **Requiring name(s) from the Locals Extensions word set** shall be appended to the label of Standard Programs that require the system to provide portions of the Locals Extensions word set.

The phrase **Requiring the Locals Extensions word set** shall be appended to the label of Standard Programs that require the system to provide all of the Locals and Locals Extensions word sets.

13.6 Glossary

13.6.1 Locals words

13.6.1.0086 **(LOCAL)**
paren-local-paren LOCAL

```
Interpretation: Interpretation semantics for this word are undefined.

Execution: ( c-addr u -- )
```

When executed during compilation, (LOCAL) passes a message to the system that has one of two meanings. If u is non-zero, the message identifies a new local whose definition name is given by the string of characters identified by c-addr u. If u is zero, the message is **last local** and c-addr has no significance.

The result of executing (LOCAL) during compilation of a definition is to create a set of named local identifiers, each of which is a definition name, that only have execution semantics within the scope of that definition's source.

```
        local Execution: ( -- x )
```

Push the local's value, x, onto the stack. The local's value is initialized as described in 13.3.3 Processing locals and may be changed by preceding the local's name with TO. An ambiguous condition exists when local is executed while in interpretation state.

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 (LOCAL) 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 Locals 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

13.6.1.2295 **TO**
LOCAL

Extend the semantics of 6.2.2295 TO to be:

```
        Interpretation: ( x "<spaces>name" -- )
```

Skip leading spaces and parse name delimited by a space. Store x in name. An ambiguous condition exists if name was not defined by VALUE.

```
        Compilation: ( "<spaces>name" -- )
```

Skip leading spaces and parse name delimited by a space. Append the run-time semantics given below to the current definition. An ambiguous condition exists if name was not defined by either VALUE or (LOCAL).

```
        Run-time:       ( x -- )
```

Store x in name.

Note: An ambiguous condition exists if either POSTPONE or [COMPILE] is applied to TO.

See: 3.4.1 Parsing, A.13.6.1.2295 TO

13.6.2 Locals extension words

13.6.2.1795 **LOCALS|**
locals-bar LOCAL EXT

```
        Interpretation: Interpretation semantics for this word are undefined.

        Compilation: ( "<spaces>name1" "<spaces>name2" ... "<spaces>namen" | -- )
```

Create up to eight local identifiers by repeatedly skipping leading spaces, parsing name, and executing 13.6.1.0086 (LOCAL). The list of locals to be defined is terminated by |. Append the run-time semantics given below to the current definition.

```
        Run-time:       ( xn ... x2 x1 -- )
```

Initialize up to eight local identifiers as described in 13.6.1.0086 (LOCAL), each of which takes as its initial value the top stack item, removing it from the stack. Identifier name1 is initialized with x1, identifier name2 with x2, etc. When invoked, each local will return its value. The value of a local may be changed using 13.6.1.2295 TO.

See: `A.13.6.2.1795 LOCALS|`

Table of Contents

Next Section