String Library for Nasforth	

Constants

:="	($Caddr "string" --)	Assigns the string "string" to the string constant. e.g. msg :=" hello mother!" N.B. Don't use in a definition only interpreted
>$	($Caddr --)	Moves a string constant to the string stack
>$CONST	($Caddr --) (ss: str --)	Move top of string stack to the string constant
$CONST	(max_len "name" --)	Creates a string constant. When "name" is referenced the address of the max_len field is pushed to the stack. e.g. 100 string msg A $const consists of a 'maxlength' long, 'currentlength' long followed by the actual characters. No termination character. So the address of the first char is msg 2 cells +
CLEN$	($Caddr -- len)	Given the address of a string constant, returns its length
MAXLEN$	($Caddr -- max_len)	Given the address of a string constant, returns its maximum length

Display

.$	(--) (ss: str --)	Pop and display the topmost string from string stack
.$CONST	($Caddr --)	Displays the string constant. e.g. fred .$const
$.S	(--) (ss: --)	Non-destructively displays the string stack

Manipulation

+$	(--) (ss: s1 s2 -- s2+s1)	Replaces the top most two strings on the string stack with their concatenated equivalent
TRIM$	(--) (ss: s1 -- s2)	Remove both leading and trailing spaces from s1, resulting in s2
LCASE$	(--) (ss: STR -- str)	On the topmost string, converts all upper case characters to lower case
LEFT$	(len --) (ss: str1 -- str1 str2)	The leftmost len characters are pushed to the string stack as a new string. The original string is retained
LTRIM$	(--) (ss: s1 -- s2)	Removes leading spaces from s1, resulting in s2
MID$	(start len --) (ss: str1 -- str1 str2)	The word mid$ produces a sub-string on the string stack, consisting of the characters from the topmost string starting at character start and ending at character end
REPLACE$	(-- pos) (found: ss: s1 s2 s3 -- s4 not found: s1 s2 s3 -- s1 s2)	In string s2 find s3 and replace with s1, resulting in s4. If a replacement is made, the starting position of the replacement is returned, otherwise -1 is returned
REV$	(--) (ss: s1 -- s2)	Reverse topmost string on string stack
RIGHT$	(len --) (ss: str1 -- str1 str2)	The rightmost len characters, pushed to the string stack as a new string. The original string is retained
RTRIM$	(--) (ss: s1 -- s2)	Removes trailing spaces from s1, resulting in s2
UCASE$	(--) (ss: str -- STR)	On the topmost string, converts all lower case characters to upper case

Search

FIND$	(offset -- pos|-1) (ss: s1 s2 -- s1)	Searches string s1, beginning at offset, for the substring s2. If the string is found, returns the position of the string relative to the offset, otherwise returns -1
FINDC$	(char -- pos|-1) (ss: --)	Returns the first occurance of the character char in the top string. The string is retained. Returns -1 if the char is not found

Stack

-ROT$	(--) (ss: s3 s2 s1 -- s1 s3 s2)	Rotates the top three string to the right
==$?	(-- flag) (ss: --)	Performs a case-sensitive comparison of the topmost two strings on the string stack, returning true if their length and contents are identical, otherwise returning false
$"	("string" --)	Pushes a string directly to the string stack. e.g. $" hello world" .$
DEPTH$	(-- $sDepth)	Returns the depth of the string stack
DROP$	(--) (ss: str --)	Drops the top string from the string stack
DUP$	(--) (ss: s1 -- s1 s1)	Duplicates a string on the string stack
LEN$	(-- len) (ss: --)	Returns the length of the topmost string
NIP$	(--) (ss: s1 s2 -- s2)	Remove the string under the top string
OVER$	(--) (ss: s1 s2 -- s1 s2 s1)	Move a copy of s1 to top of string stack
PICK$	(n --) (ss: -- strN)	Given an index into the string stack, copy the indexed string to the top of the string stack. 0 $pick is equivalent to $DUP 1 $pick is equivalent to $OVER etc.
ROT$	(--) (ss: s3 s2 s1 -- s2 s1 s3)	Rotates the top three string to the left
SWAP$	(--) (ss: s1 s2 -- s2 s1)	Swaps the top two string items on the string stack

String ← → Number

NO	**$>D**	(-- d) (ss: str --)	Interprets the topmost string as a number, returning its value on the data stack as a double length signed integer
NO	**D>$**	(d --) (ss: -- str)	Pushes the double length number on the data stack to the string stack

$Const internals

Maximum length – set when $Const created	← cell
Present length – set whenever string changed	← cell
S t r i n g	← byte / chr

String Stack Internals

With two strings on the stack, the contents is:-

Length of string1	← ($sp@) points here	← Lowest address is top of stack
s t r I n g 1 padding if reqd	← byte / chr	
Length of string2		
S t r I n g 2 padding if reqd	← byte / chr	← Highest address

N.B. the length cells contain (No. of chrs + length cell + any padding) in bytes