

U7. Strings

UtilIt supports several commands that perform common string-related operations: setting substrings, trimming strings, setting parameter text, converting strings from one format to another, finding strings within text blocks, and interconverting strings and numbers.

SetSub 451 a,b,c,d,uString,uName

Moves the contents of uString into uName beginning at character a of uName and ending at character b (i.e. sets a substring in uName). Undefined characters are replaced with spaces so that SetSub works even if uName is less than b characters in length, or uString is smaller than the substring. You can optionally use parameter c to pass the address of a Pascal-type source string (instead of using uString), and/or d to pass the address of a Pascal-type destination string (instead of using uName).

NOTE: You can get substrings using the command GetStr described in the "String Lists" topic.

TrmStr 452 a,b,uString,uName

Moves or removes leading or trailing spaces within the string uString, uName, or some other string designated by a, according to the type of adjustment specified by b.

a = string to adjust

0 or 1 = uString

2 = uName

other = address of a Pascal string

b = type of adjustment to make

1 = removes (trims) trailing spaces

0 = removes both leading and trailing spaces

-1 = removes leading spaces

-2 = moves leading spaces to end of string

SetPrm 453 a,b,c,d,uString

Resets the four parameter text strings (^0, ^1, ^2, and ^3 items in dialogs and alerts) to the strings designated by a, b, c, and d where.

0 = empty string

1 to 255 = size of substring from uString

other = address of a Pascal string

If using substrings from uString, these are obtained in succession, and trimmed of both leading and trailing spaces before being assigned to parameter text. For example, the call

"Facelt(nil,SetPrm,10,20,10,30)" would set the ^0 item to the first 10 characters of uString, ^1 to the next 20 characters, ^2 to the next 10 characters, ^3 to the next 30 characters (= 70 total characters of uString used). This approach can be mixed with that of passing Pascal string addresses. SetPrm is primarily provided for those programmers who do not have access to the toolbox routine ParamText, but is also useful when "writing" multiple parameter text items to a single string.

NOTE: This command is largely obsolete now since the basic control driver used with ViewIt provides a more powerful "parameter text" scheme based on the use of string lists.

CnvStr 454 a,b,c,d,uString,uName

Converts the string designated by parameter c from string type a to type b. Parameter d designates the total number of bytes occupied by the string variable (its "storage size") which is usually larger than the number of characters in the string (the "string length").

a = source type (0 = Pascal, 1 = C, 2 = Fortran)

b = converted type (0 = Pascal, 1 = C, 2 = Fortran)

c = source string (0 or 1 = uString, 2 = uName, other = string address)

d = source string storage size ($0 \leq d \leq 256$ bytes)

(if d = 0, UtilIt assumes d = 256 bytes)

FndTxt 455 a,b,c,d,uResult

Searches within the text block defined by a and b for the string defined by c and d. Note that the text block must be locked in memory during the search since FndTxt can move heap memory. uResult

returns the position of the found string as a byte offset from a, or -1 if not found.

a = address of text block to search

b = size of text block to search (bytes)

c = address of search text

d = size of search text (use -d for case sensitive search)

NumToS 471 a,b,c,d,uString

Converts a number from the variable designated by b to the string designated by a, using the format indicated by c and d. Infinities will appear as the character "∞".

a = destination string

0 = uString

other = address of a Pascal string

b = source of number (as a data type)

1 = ul1 5 = uR4

2 = ul2 6 = uR8

3 = ul4 7 = uR10

4 = ul8 8 or 12 = uR12 (12 = THINK C "universal")

c = format

0 = general (= fixed point unless very large or small)

1 = floating point

2 = fixed point

d = digits (use -d to also trim trailing decimal zeroes)

if c = 0 or 1, d = significant figures (-4 used if d = 0)

if c = 2, d = decimals to display

SToNum 481 a,b,uString,uResult,f1Err...

Converts a string designated by a to an integer or real number in the variable designated by b. The "∞" character can be used in strings to represent infinities. Leading and trailing spaces are ignored.

a = source string

0 = uString

other = address of a Pascal string

b = destination variable (as a data type)

1 = ul1 5 = uR4

2 = ul2 6 = uR8

3 = ul4 7 = uR10

4 = ul8 8 or 12 = uR12 (12 = THINK C "universal")

If no error occurs, then uResult is set equal to zero, else uResult returns with a value less than zero and the variable (ul1...uR12) is set equal to the corresponding error value in fRec (f1Err..fR12Err). The default error values are zero, but these can be changed at any time to other values if you prefer a special non-zero value to be returned when an error occurs.

Conditions causing an error include an empty string or a string that cannot be evaluated as a number (an NAN). A number that is out of the range of values represented by the destination variable's numerical type is either set equal to the maximum or minimum integer (for integer types), or to ±∞ or 0 (for real types).