NAME

string.lib – INMOS occam toolset string library

SYNOPSIS

```
#USE "string.lib"
```

SUMMARY

The occam toolset library string.lib provides various string handling functions and procedures.

OVERVIEW

To use the functions in this library do the following:

```
#USE "string.lib"
```

Procedures

```
PROC append.char (INT len, []BYTE str, VAL BYTE char)
PROC append.hex.int (INT len, []BYTE str, VAL INT number,
                     VAL INT width)
PROC append.hex.int64 (INT len, []BYTE str, VAL INT64 number,
                       VAL INT width)
PROC append.int (INT len, []BYTE str, VAL INT number,
                 VAL INT field)
PROC append.int64 (INT len, []BYTE str, VAL INT64 number,
                   VAL INT field)
PROC append.real32 (INT len, []BYTE str, VAL REAL32 number,
                    VAL INT Ip, VAL INT Dp)
PROC append.real64 (INT len, []BYTE str, VAL REAL64 number,
                    VAL INT Ip, VAL INT Dp)
PROC append.text (INT len, []BYTE str, VAL []BYTE text)
```

Append the given data type to an array `str` at position `len` using formatting where necessary and possibly padding with spaces to a given field `width` or printing `Ip` integer places and `Dp` decimal places for `REAL32` and `REAL64` types. Hex numbers are formatted prefixed by a '#'. STOPs if the array overflows. `len` is updated to the new position.

```
INT FUNCTION char.pos (VAL BYTE search, VAL []BYTE str)
```

Return the position of the first occurence of character `search` in `str` or -1 if not found. See also `string.pos`.

```
INT FUNCTION compare.strings (VAL []BYTE str1, VAL []BYTE str2)
```

Return result of comparsion:

rf(CR) l. -2 if str1 is earlier than str2 -1 if str1 is a leading substring of str2
0 if the strings are identical
1 if str2 is a leading substring of str1
2 if str2 is earlier than str1

```
PROC delete.string (INT len, []BYTE str, VAL INT start,
                    VAL INT size, BOOL not.done)
```

Delete `size` characters from position `start` of string `str` of size `len` characters. `not.done` is `TRUE` if the range is invalid. See also `str.shift`.

```
BOOL FUNCTION eqstr (VAL []BYTE s1, VAL []BYTE s2)
```

Return `TRUE` if the strings are identical.

```
PROC insert.string (VAL []BYTE new.str, INT len, []BYTE str,
                    VAL INT start, BOOL not.done)
```

Insert `new.str` into `str` of size `len` characters at position `start`. `not.done` is `TRUE` if the range is invalid. See also `str.shift`.

```
BOOL FUNCTION is.digit (VAL BYTE char)
BOOL FUNCTION is.hex.digit (VAL BYTE char)
```

TRUE if the given character is a valid decimal or hex digit respectively.

```
BOOL FUNCTION is.id.char (VAL BYTE char)
```

TRUE if the given character can be part of an occam identifier.

```
BOOL FUNCTION is.in.range (VAL BYTE char, VAL BYTE bottom, VAL BYTE top)
```

Return TRUE if bottom <= char <= top.

```
BOOL FUNCTION is.lower (VAL BYTE char)
BOOL FUNCTION is.upper (VAL BYTE char)
```

Return TRUE if the given character is an upper or lower case letter.

```
PROC next.int.from.line (VAL []BYTE line, INT ptr,
                         INT number, BOOL ok)
PROC next.word.from.line (VAL []BYTE line, INT ptr,
                          INT len, []BYTE word, BOOL ok)
```

Return the next white-space separated integer/word from the array line, using ptr as a index into it. Updates ptr and sets ok to FALSE if the operation fails. len is the returned length of word or zero on end of line.

```
INT,BYTE FUNCTION search.match (VAL []BYTE possibles, VAL []BYTE str)
```

Search str for any of the character in possibles. Return the position and character or -1,255(BYTE) if none are found.

```
INT,BYTE FUNCTION search.no.match (VAL []BYTE possibles, VAL []BYTE str)
```

Search str for any character *not* in possibles. Return the position and character or -1,255(BYTE) if none are found.

```
PROC str.shift ([]BYTE str, VAL INT start, VAL INT len, VAL INT shift,
                BOOL not.done)
```

Move the substring [str FROM start FOR len right shift places. not.done is TRUE if the operation is invalid. See also delete.string and insert.string.

```
INT FUNCTION string.pos (VAL []BYTE search, VAL []BYTE str)
```

Return the position of the string search in str or -1 if not found. See also char.pos.

```
PROC to.lower.case ([]BYTE str)
PROC to.upper.case ([]BYTE str)
```

Convert the given string to lower/upper case respectively.

SEE ALSO

INMOS occam 2 toolset user manual - part 2 (occam libraries and appendices) INMOS document number 72 TDS 276 02.

AUTHOR