NAME

streamio.lib – INMOS occam toolset stream file library

SYNOPSIS

```
#INCLUDE "streamio.inc"
#USE "streamio.lib"
```

SUMMARY

The occam toolset library streamio.lib provides a stream file interface between the host and the transputer network.

OVERVIEW

To use the functions in this library do the following:

```
#INCLUDE "streamio.inc"
#USE "streamio.lib"
```

The functions in this library have three protocols in general.

The first is `CHAN OF SP fs, ts` which is the protocol to and from the host server and refers to most procedures in `hostio.lib`.

The second is `CHAN OF KS` which is the `keystream` or *key stream* protocol, providing input from the keyboard.

The third is `CHAN OF SS` which is the `scrstream` or *screen stream* protocol which provides output to the screen.

Error returns from procedures are passed out a variable called `result` which can take the following values:

lf(CR) lw(5i). spr.ok The operation was successful spr.bad.name Invalid name parameter spr.bad.packet.size Some data was larger than the built in buffer >=spr.operation.failed Server returned a failure

KS (Key Stream) Procedures

```
PROC ks.keystream.sink (CHAN OF KS keys)
```

Throw away the *key stream* (`KS`) input until a termination.

```
PROC ks.keystream.to.scrstream (CHAN OF KS keyboard, CHAN OF SS scrn)
```

Convert *key stream* (`KS`) protocol to *screen stream* (`SS`) protocol.

```
PROC ks.read.char (CHAN OF KS source, INT char)
PROC ks.read.int (CHAN OF KS source, INT number, INT char)
PROC ks.read.int64 (CHAN OF KS source, INT64 number, INT char)
PROC ks.read.line (CHAN OF KS source, INT len, []BYTE line,
                   INT char)
PROC ks.read.real32 (CHAN OF KS source, REAL32 number, INT char)
PROC ks.read.real64 (CHAN OF KS source, REAL64 number, INT char)
```

Read data from the keyboard using *key stream* (`KS`) protocol.

SO (Screen Output) Procedures

```
PROC so.keystream.from.file (CHAN OF SP fs, ts,
                             CHAN OF KS keys.out,
                             VAL []BYTE filename,
                             BYTE result)
PROC so.keystream.from.kbd (CHAN OF SP fs, ts,
                            CHAN OF KS keys.out,
                            CHAN OF BOOL stopper,
                            VAL INT ticks.per.poll)
```

```
PROC so.keystream.from.stdin (CHAN OF SP fs, ts,
                              CHAN OF KS keys.out, BYTE result)
```

Read characters from a file, the keyboard or standard input respectively and convert to the *key stream* (KS) protocol. Termination for the file and standard input is on end of file but the keyboard handler must be terminated by sending any BOOL along stopper.

```
PROC so.scrstream.to.ANSI (CHAN OF SP fs, ts, CHAN OF SS scrn)
PROC so.scrstream.to.TVI920 (CHAN OF SP fs, ts, CHAN OF SS scrn)
PROC so.scrstream.to.file (CHAN OF SP fs, ts, CHAN OF SS scrn,
                           VAL []BYTE filename, BYTE result)
PROC so.scrstream.to.stdout (CHAN OF SP fs, ts, CHAN OF SS scrn,
                             BYTE result)
```

Convert the *screen stream* (SS) protocol to output on an ANSI terminal, TVI920 terminal, a file or to standard output respectively.

SS (Screen Stream) Procedures

```
PROC ss.beep (CHAN OF SS scrn)
```

Output a terminal bell/beep.

```
PROC ss.clear.eol (CHAN OF SS scrn)
```

Make the terminal clear to end of line.

```
PROC ss.clear.eos (CHAN OF SS scrn)
```

Make the terminal clear to end of the screen.

```
PROC ss.del.line (CHAN OF SS scrn)
```

Make the terminal delete the current line.

```
PROC ss.delete.chl (CHAN OF SS scrn)
```

Make the terminal delete the character to the left of the cursor.

```
PROC ss.delete.chr (CHAN OF SS scrn)
```

Make the terminal delete the character to the right of the cursor.

```
PROC ss.down (CHAN OF SS scrn)
```

Move the terminal cursor down.

```
PROC ss.goto.xy (CHAN OF SS scrn, VAL INT x, y)
```

Move the terminal cursor to the given x,y coordinate (from top left).

```
PROC ss.ins.line (CHAN OF SS scrn)
```

Make the terminal insert a new line at the cursor position.

```
PROC ss.insert.char (CHAN OF SS scrn, VAL BYTE ch)
```

Make the terminal insert a new character at the cursor position.

```
PROC ss.left (CHAN OF SS scrn)
```

Move the terminal cursor left.

```
PROC ss.right (CHAN OF SS scrn)
```

Move the terminal cursor right.

```
PROC ss.scrstream.copy (CHAN OF SS scrn.in, scrn.out)
```

Copy the output of one *screen stream* onto another.

```
PROC ss.scrstream.fan.out (CHAN OF SS scrn,
                           CHAN OF SS screen.out1,
```

```
                                        screen.out2)
```

Copy the output of one *screen stream* onto two others.

```
PROC ss.scrstream.from.array (CHAN OF SS scrn,
                              VAL []BYTE buffer)
```

Provide a *screen stream* from the given array.

```
PROC ss.scrstream.multiplexor ([]CHAN OF SS screen.in,
                               CHAN OF SS screen.out,
                               CHAN OF INT stopper)
```

Multiplex several *screen streams* onto one until any `BOOL` is received along `stopper`.

```
PROC ss.scrstream.sink (CHAN OF SS scrn)
```

Throw away the *screen stream* until a terminate is received.

```
PROC ss.scrstream.to.array (CHAN OF SS scrn, []BYTE buffer)
```

Output a *screen stream* to the given array.

```
PROC ss.up (CHAN OF SS scrn)
```

Move the terminal cursor up.

```
PROC ss.write.char (CHAN OF SS scrn, VAL BYTE char)
```

Write a character to the terminal.

```
PROC ss.write.endstream (CHAN OF SS scrn)
```

Terminate a *screen stream* (`SS`).

```
PROC ss.write.hex.int (CHAN OF SS scrn, VAL INT number,
                       VAL INT field)
PROC ss.write.hex.int64 (CHAN OF SS scrn, VAL INT64 number,
                         VAL INT field)
PROC ss.write.int (CHAN OF SS scrn, VAL INT number,
                   VAL INT field)
PROC ss.write.int64 (CHAN OF SS scrn, VAL INT64 number,
                     VAL INT field)
PROC ss.write.nl (CHAN OF SS scrn)
PROC ss.write.real32 (CHAN OF SS scrn, VAL REAL32 number,
                      VAL INT Ip, VAL INT Dp)
PROC ss.write.real64 (CHAN OF SS scrn, VAL REAL64 number,
                      VAL INT Ip, VAL INT Dp)
PROC ss.write.string (CHAN OF SS scrn, VAL []BYTE str)
PROC ss.write.text.line (CHAN OF SS scrn, VAL []BYTE str)
```

Write the given data type to a screen stream 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 written prefixed by a '#'. PROCedures ending in `.nl` write a newline sequence.

SEE ALSO

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

AUTHOR