TM-2337/102/00

TECHNICAL MEMORANDUM

(TM Series)

This document was produced by SDC in performance of contract SD-97

Input-Output File and Library Functions

The Q-32 LISP 1.5 Mod. 2.5 System

S. L. Kameny

22 September 1965

SYSTEM DEVELOPMENT CORPORATION
2500 COLORADO AVE.
SANTA MONICA
CALIFORNIA

TABLE OF CONTENTS

TABLES

ABSTRACT

This document supplements TM-2337/101/00 by describing the input-output, file-handling and library functions of Q-32 LISP 1.5 Mod. 2.5.

1. INTRODUCTION

The Q-32 LISP 1.5 Mod. 2.5 system is the LISP system that has existed relatively unchanged since March 1965. This document describes the input-output and associated file-handling and library functions of Mod. 2.5. It is written as a separate document, supplementing the Q-32 LISP Reference Manual; this is because, in Mod. 2.6, the input-output functions have been rewritten and expanded to allow for use of disk files and generally greater flexibility in file use.

2. TABLE OF FUNCTIONS

Table 1 contains a list of names of functions described in this document, together with function types, tape primitives, file handling, library handling, scope primitives and miscellaneous functions.

3. TAPE PRIMITIVES

The LISP tape-handling functions read and write Hollerith magnetic tapes in a form of S-expressions. A tape written by LISP can, with one exception, always be read by LISP (assuming no real tape parities occur). Also, any magnetic tape produced with S-expressions can be read by LISP, but the behavior of the LISP system given a tape containing an illegal expression cannot be guaranteed.

It is possible to use LISP to write a tape that does not contain S-expressions and so cannot subsequently be read by LISP, by using atoms whose print names contain illegal characters or unsyntactic combinations. For example, assuming tape Ø has been declared, PRINTAPE ($$/))/Ø) will produce an unreadable record containing only)), which is not an S-expression.

3.1 REQUESTING AND RELEASING TAPES

GETFILE (n)

Here, n is, in general, the physical reel number (must be a number). To obtain a scratch tape, use GETFILE (Ø).

In general, GETFILE declares a Hollerith tape file n to TSS, and returns n.

GETFILE must be used following LOAD and before the tape file n can be used by the LISP system for either reading or writing. Any attempt to use a tape without previously using GETFILE will result in an essentially irrecoverable error. The only exception to this rule is the use of the function SAVE (see below).

Table 1. Input-Output Function Types

Function Name	No. of Args.	Function Type
BACKFILE	1	tape primitive
BACKRECORD	1	tape primitive
COPYF	2	file handling
COPYS	3	file handling
DEFILE	1	tape primitive
GETFILE	1	tape primitive
GETSCOPE	1	scope primitive
LOADF	1	library handling
LOADLIB	1	library handling
LOADS	2	library handling
LOCBUF	1	miscellaneous
PLOT	3	scope primitive
PRINTAPE	2	tape primitive
PRINTSCOPE	2	scope primitive
READC	2	miscellaneous
READF	1	file handling
READS	2	file handling
READSCOPE	2	scope primitive
READTAPE	1	tape primitive
REWIND	1	tape primitive
SCANF	1	file handling
SKIPF	1	tape primitive
SKIPS	2	file handling
TTEREAD	∅	tape primitive
WEOF	1	tape primitive
WEOT	1	tape primitive
WRITF	2	file handling
WRITS	2	file handling

After the function GETFILE (n) has been called, TSS will reply $WAIT, and later $FILE n00 DRIVE d REEL n. Finally, LISP will reply n, then 2 bells (you can then continue).

DEFILE (n)

Defiles and rewinds tape file n, returns n.

DEFILE should always be used to release tapes as soon as the user is finished with them.

3.2 WRITING AND READING TAPES

PRINTAPE (s n)

Writes the S-expression s on tape file n, returns s. (Note: A single S-expression may occupy many tape records.)

While PRINTAPE is the primitive Hollerith tape-writing function in Q-32 LISP, most users will generally find the functions WRITF and WRITS, which use PRINTAPE, to be more useful (see section 4).

READTAPE (n)

READTAPE reads the next S-expression from tape input buffer. If the buffer is exhausted, it reads the next record from tape n into the buffer, then reads one S-expression from the buffer.

If an end-of-file mark or an end-of-tape mark is encountered at the beginning of the first read, READTAPE returns EOF or EOT, respectively. (Note: No distinction is made by READTAPE between an actual EOF or EOT or a record starting with the word EOF or EOT.) In all other cases, it returns the S-expression read. (Cards may be prestored up to 30 per record.)

The primitive function READTAPE can get the user into trouble if he attempts to read past the end-of-file. The file-manipulating functions given in section 4 are generally more useful. However, if the user wishes deliberately to read an end-of-file when the tape is positioned immediately before end-of-file, he must use READTAPE or SKIPF.

TTEREAD ( )

READTAPE loads an internal buffer that is refreshed from tape each time READTAPE encounters the end of record. TTEREAD effectively terminates the buffer so the next execution of READTAPE will load the buffer from the next tape record.

3.3 OTHER TAPE PRIMITIVES

BACKFILE (_n_)

Backs up tape file _n_ until the first end-of-file or end-of-tape is encountered, or, if none is found, rewinds tape file _n_.

Leaves tape in position to read that end-of-file or end-of-tape, or the first record on tape.

Returns _n_.

(Note: If the tape is initially positioned immediately after an and-of-file or end-of-tape, BACKFILE may miss it, while BACKRECORD will always succeed.)

BACKRECORD (_n_)

Backs up one record (not necessarily one S-expression) on tape file _n_; returns _n_.

REWIND (_n_)

Rewinds tape file _n_; returns _n_.

If tape is already rewound, returns _n_.

SKIPF (_n_)

SKIPF skips one file on tape file _n_. It makes a series of tape-read-move calls to the Time-Sharing System, until the first end-of-file or end-of-tape is encountered, then returns EOF or EOT with the tape positioned after the end-of-file or end-of-tape.

WEOF (_n_)

Writes an end-of-file on tape _n_; returns _n_.

WEOT (_n_)

Writes an end-of-tape on tape _n_; returns _n_.

4. FILE-HANDLING FUNCTIONS

The file-handling functions described here are written in terms of the tape primitives described earlier. In general, all tape file functions (ending in letter F) read or write, leaving the tape positioned after the end-of-file or end-of-tape. All selective functions ending in letter S read or write, leaving the tape positioned before an S-expression or end-of-file. Reading will not go beyond an end-of-file, while writing always leaves the written tape positioned before an end-of-file.)

COPYF (m n)

COPYF copies one file from tape file m to tape file n. (It produces an error if m = n.)

If only end-of-file or end-of-tape is found on m, it returns EOF or EOT and leaves both tapes in their initial position. In all other cases, it copies one file from tape file m to tape file n, then writes two end-of-files on tape n and positions n after the first end-of-file written. Finally, it leaves tape m positioned after the first end-of-file or before the first end-of-tape read, and returns a list of first atom or dot pair from each S-expression copied, with EOF or EOT at the end of the list.

COPYS (m n p)

COPYS copies up to m S-expressions from tape file n to tape file p. It first reads up to m S-expressions from tape n. (If end-of-file or end-of-tape is encountered, reading stops and tape n is positioned just before the end-of-file or end-of-tape.) Then it writes onto tape file p all S-expressions read, writes two end-of-files onto tape file p, and positions p just before the first end-of-file written. Finally, it returns list of S-expressions copied. If end-of-file or end-of-tape was encountered during the read, EOF or EOT is the end of the list.

READF (n)

READF reads one full file from tape file n (without checking the initial position). If only an end-of-file or end-of-tape is found, it returns EOF or EOT and leaves the tape positioned before the end-of-file or end-of-tape. Otherwise, it returns a list of all expressions read, with EOF at the end of the list, and leaves tape n positioned after the end-of-file or before the end-of-tape.

READS (m n)

READS reads m S-expressions from tape file n unless EOF or EOT is encountered sooner. It returns the list of all S-expressions read (EOF or EOT appear at end of list if encountered), and leaves tape file n positioned either before the first end-of-file or at the end of the mth S-expression. Finally, it returns a list of S-expressions copied. If end-of-file or end-of-tape was encountered during the read, EOF or EOT is the end of the list.

SCANF (n)

SCANF scans or surveys one file on tape file n. It reads one file from tape file n. If only an end-of-file or end-of-tape is found, the tape is left positioned before the end-of-file or end-of-tape and EOF and EOT is returned. Otherwise, it returns a list of the first atom or dot pair from each S-expression read from the file, with EOF or EOT at the end of the list, and leaves the tape positioned after the end-of-file read or before the end-of-tape read.

```
(SCANF (LAMBDA (N) (MAPCAR (READF N) (FUNCTION FIRST))))
```

SKIPS (m n)

SKIPS skips over m S-expressions on tape file n. It reads up to m S-expressions on tape file n, but stops reading if EOF or EOT is encountered first. The tape file n is left positioned after the mth S-expression or before end-of-tape or end-of-file. Finally, SKIPS returns a list of the first atom or dot pair from each S-expression read. If end-of-file or end-of-tape is encountered, EOF or EOT is at the end of the list.

```
(SKIPS (LAMBDA (M N) (MAPCAR (READS M N) (FUNCTION (FIRST)))))
```

(Note: There is no function SCANS since SKIPS performs a function similar to SCANF.)

WRITF (ℓ n)

WRITF writes one full file on tape file n. Here ℓ must be a list of expressions to be written on tape. In ℓ, EOF and EOT are ignored at the top level (i.e., do not produce any effect on the tape). All other S-expressions in ℓ are written on the tape file n. Then two end-of-files are written on tape file n, and the tape is left positioned after the first end-of-file written. Finally, the expression ℓ is returned as the value of WRITF.

WRITS (ℓ n)

WRITS writes a series of S-expressions on tape file n. Here ℓ must be a list of S-expressions to be written on tape. EOF and EOT are ignored on the top level of ℓ. All other S-expressions in ℓ are written on tape file n. Then two end-of-files are written on tape file n, and tape file n is left positioned just before the first end-of-file written. Finally, the list ℓ is returned as the value of WRITS.

```
(WRITS (LAMBDA (L N) (PROG NIL (WRITF L N) (BACKRECORD N) (RETURN L))))
```

5. LIBRARY-HANDLING FUNCTIONS

The library-handling functions described here are used to load library expressions which can be in one of two formats.

If the atom *LDLC is set to true (non-NIL), then the format of a library record is

(name f_1 a_1 f_2 a_2 ... f_n a_n)

where name is used as the name of the library expression (the file name in EDLISP) and f_i a_i represents a pair of S-expressions to be given to Evalquote. This is the standard form of a library record and is compatible with EDLISP.

The other format, accepted if *LDLC is NIL, is

(_name_ _function_ _arguments_)

where _argument_ is the series of arguments required by _function_, not a list of arguments. This format is less useful than the standard format, since it accepts only a single function. It has been retained for compatibility with previously-produced library tapes.

LOADLIB (s)

LOADLIB loads or performs one library expression _s_, applying each function f_i to its arguments a_i, and returns a list of the form

```
(name (FIRST f1) (EVALQT f1 a1)
      (FIRST f2) (EVALQT f2 a2)
      ...
      (FIRST fn) (EVALQT fn an))
```

LOADF (_n_)

LOADF performs or loads an entire file of library expressions from tape file _n_ and returns a list of the outputs of LOADLIB. If the tape was initially positioned immediately before an end-of-file or an end-of-tape, it is left in its initial position. Otherwise, the tape will be left positioned after an end-of-file or before an end-of-tape.

(Note: If an error is found in format of _ℓ_, the tape will be positioned after the read from which the error occurred.)

```
(LOADF (LAMBDA (N) (MAPCAR (READF N) (FUNCTION  LOADLIB))))
```

LOADS (m n)

LOADS performs or loads m library expressions from tape file n and returns a list of the outputs of LOADLIB.

If end-of-file or end-of-tape is encountered before m expressions are read from tape file n, the read will stop, and the tape will be positioned before that end-of-file or end-of-tape.

```
(LOADS (LAMBDA (M N) (MAPCAR (READS M N) (FUNCTION  LOADLIB))))
```

6. SCOPE FUNCTIONS

The scope functions described here are used to plot displays on the dd-19 CRT display consoles and to accept light-pen inputs.

GETSCOPE (n) $\emptyset \le n \le 6$

GETSCOPE requests TSS to reserve scope number n. This procedure must be called prior to the use of any scope procedures; however, it need be called only once. n is the value of the procedure.

PLOT (m i s)

Procedure PLOT inserts a vector or character specified in list i into an array m based on a search criterion specified in list s. If m is not bound to an array, or the search fails, the value of procedure is NIL. If successful, the value of procedure is a list ℓ of the contents of search match word of array m. The form of i, s, and ℓ is:

((x.Δx) (y.Δy) (char . size) id);

where

x and y must satisfy $0 \le x, y \le 1\emptyset23$

Δx and Δy must lie in the range $-127 \le \Delta x, \Delta y < +127$

char = any LISP atom whose first character will be displayed

size must lie in the range $\emptyset \le 3$ where $\emptyset$ designates the smallest character size, and 3 indicates the largest size character (size is disregarded for vectors).

For id, the following conventions are used:

1 = Scope 1 or 4

2 = Scope 2 or 5

3 = Scopes (1 and 2) or (4 and 5)

4 = Scope 3 or 6

5 = Scopes (1 and 3) or (4 and 6)

6 = Scopes (2 and 3) or (5 and 6)

7 = Scopes (1, 2, and 3) or (4, 5, and 6)

Note 1: A vector is designated when either Δx or Δy is not NIL, and char = BLANK.

Note 2: To display data, PRINTSCOPE must be used. The scope number in PRINTSCOPE determines whether Scopes 1, 2, 3 or 4, 5, 6 are used.

Example:

```
PLOT (BUFF1
     ((100 . 127) (100) ($$$ $) 4)
     ((NIL) (453) (E) 4))
```

This will enter a maximum sized vector for display on scope 3 or 6 at coordinate (100, 100) if in array BUFF1 there is found on scope 3 or 6 an E at any column row 453. Note that any x or SIZE will satisfy the search, since these parameters were NIL. If the whole third argument of PLOT were NIL ($\underline{s}$ = NIL), then the vector would be inserted at the first empty array word, i.e., a search for first NIL word of the array.

PRINTSCOPE ($\underline{m}$ $\underline{n}$) $\emptyset \le n \le 6$

PRINTSCOPE dumps the array named $\underline{m}$ on Q-32 drums for display on SDC scope number $\underline{n}$. The value of the procedure is NIL if $\underline{m}$ is not bound to an array; otherwise the value is $\underline{m}$. If $\underline{n}$ is not a previously requested scope (GETSCOPE), LISP will be stopped, and a TSS error will result.

READSCOPE (m n)

READSCOPE is used to obtain a light-pen input for LISP. m is the buffer used in PLOT. n is the console id as noted in PLOT. After the user light pens a character or vector, READSCOPE returns a list of the form ((x.Δx) (y.Δy) (char . size) id) that describes the character or vector in the same form used for PLOT.

7. MISCELLANEOUS FUNCTIONS

The two functions READC and LOCBUFF described here are included in this document rather than the LISP 1.5 Reference Manual because they are peculiar to Mod. 2.5.

LOCBUF (a)

LOCBUF is a primitive used by EXPLODE and necessary to call READC. The value of LOCBUF is the octal location of the head of the print name buffer for the atom a.

READC (p b)

READC is a primitive used by EXPLODE. The value of READC is the character atom at the bth byte address (from o to b) of the print name buffer at location p. If b exceeds the maximum byte address of buffer p, READC returns the value NIL. p should be computed with LOCBUF.

For example, to obtain the third character in the print name of the atom to which A is bound, use (READC (LOCBUF A) 2).