Prof. Dr. H. Stoyan
Universität Erlangen-Nürnberg
Institut für Mathematische Maschinen
und Datenverarbeitung (Informatik VIII)
Am Weichselgarten 9
91058 Erlangen

TM-2337/103/00

TECHNICAL MEMORANDUM

(TM Series)

This document was produced by SDC in performance of contract AF 19(628)-5166 with the Electronic Systems Division, Air Force Systems Command, in performance of ARPA Order 773 for the Advanced Research Projects Agency Information Processing Techniques Office.

THE Q-32 LISP 1.5 MOD. 2.6 SYSTEM

Operating System, Input-Output, File, and Library Functions

S. L. Kameny
C. Weissman

11 April 1966

SYSTEM DEVELOPMENT CORPORATION
2500 COLORADO AVE.
SANTA MONICA
CALIFORNIA
90406

The views, conclusions or recommendations expressed in this document do not necessarily reflect the official views or policies of agencies of the United States Government.

(Page 2 blank)

CONTENTS

Section		Page
1.	Introduction	5
2.	Table of Functions	5
3.	Input-Output Concept and Operation .	5
	3.1 File Activation	7
	3.2 File Deactivation	9
	3.3 File Selection	10
	3.4 File Control	11
	3.5 I/O Primitive Changes	12
	3.6 Scope Functions	14
	3.7 Library Functions	16
4.	System Modifications	17
	4.1 Initialization	17
	4.2 Supervisor	17
	4.3 Growing Pains	18
5.	Primitives	19
6.	Error Messages	22
	6.1 Input/Output Error Messages . .	22
	6.2 Other Error Messages	23
	6.3 Time-Sharing Error Messages . .	26
	Table 1 Functions Discussed in this Document	6

(Page 4 blank)

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. 2.6. It also describes differences between Mod. 2.6 and the previous Mod. 2.5 described in TM-2337/102/00 dated 9 August 1965.

1. INTRODUCTION

A new version of Q-32 LISP 1.5, Model 2.6 has been operational for a number of months. This new version extends the capability of the basic system described in the Q-32 LISP Reference Manual, TM-2337/101/00, in three significant ways. First, a new scheme for input-output is available that permits access to teletype, tapes, disc, and CRT devices in an integrated and consistent format. Second, certain modifications have been made to Evalquote to handle the new I/O features. Third, free space and array space have been increased to a maximum of nearly $(30,000)_{10}$ words by reducing binary program space to nearly nothing, and augmenting the system with a "growing pain" that extends binary program space dynamically by preempting array space when needed.

This document describes these new features as a separate document, supplementing the Q-32 LISP Reference Manual.

2. TABLE OF FUNCTIONS

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

3. INPUT-OUTPUT CONCEPT AND OPERATION

Model 2.6 LISP 1.5 I/O is based upon a set of primitive functions and macros that are available to the user. Each input or output operation references, either implicitly or explicitly, a specific file. A file is named by the user and associated with a particular device. A file can be any literal atom, e.g., INTAPE, A6., etc. The LISP meaning of the term "file" is the same as the meaning normally assigned by a time-sharing system. A file is device-dependent but direction-independent; the same file may be used for both input and output and can, with considerable caution, be used for both purposes simultaneously.

A symbolic file consists of a sequence of records, only one of which is ever in main memory at a time, thereby reducing buffer storage overhead. Records consist of one or more 72-character lines. When a file is activated, LISP dynamically creates from array space the buffer storage needed to hold the current record of the file. Also, when a file is deactivated, LISP reclaims this buffer storage. Many files may be active concurrently,* limited only by the LISP array space and the time-sharing devices available.

---

* Whenever a LISP program is loaded and set into operation, all previously declared files (i.e., files in existence at the time the program was saved) are removed, and the system starts off with only a single teletype file declared, named *TTY.

Table 1. Functions Discussed in this Document

Function Name	No. of Arguments	Function Type	Page No.
*IOINIT	0	System initialization	17
*IOSHUT	1	System initialization	17
LOADEXP	1	Library loading	16
OPEN	2 or 3	Macro	7
OPENFILE	3	File primitive	19
PLOT	3	CRT buffer filling	12
POSITION	2	File positioning	11
PRIN1, PRINØ PRINT, PRINTCH	2	Printing	14
*RATOM, READ READ1, READCH	0	Reading	13
READCRT	2	Light pen read	15
READFILE	1	Library handling	16
PRINTCRT	2	CRT display output	15
PRINTFILE	2	Library handling	16
RDS	1	Read select	10
SHUT	1 or 2	Macro	9
*SHUTF	2	File primitive	9
STERPRI	0	Output	13
*SUPV	0	System supervisor	17
TABIN	1	Reading	12
TABOUT	1	Printing	12
TEREAD	0	Reading	13
TERPRI	0	Printing	12
*TEVALQT	2	Evalquote	17
WRS	1	Write select	10

To access a file, the user must first select the file, either for reading or for writing. The user's program always accesses the same file for each call to standard LISP I/O primitives, e.g., PRINT, READ, TERPRI, etc. To access another file, that file must be activated. (The system supervisor always uses the input teletype console, regardless of user file selection.)

This scheme allows LISP programs to be written in a device-independent manner. A program is composed which evaluates I/O read and print primitives as necessary without immediate concern as to the source or destination of the information. Once composed, the program can be exercised from a teletypewriter file one moment, and from tape or disc files at a later time by simply activating and selecting such files in higher level programs or at the top level itself.

3.1 FILE ACTIVATION

Before a file can be selected for reading or writing, it must first be activated. This activation can most easily be done using the macro OPEN.

OPEN (f d m)

OPEN is a LISP macro that expands, using the function *OPENF, into the primitive OPENFILE, described in Section 5. It returns as its value a list of all currently active file names. Here, f is the user-specified name of the file being activated; it is a literal atom whose first six characters are used for internal identification between LISP and the time-sharing system. The argument d specifies the device associated with the file f. While the optional parameter m is used to indicate the nature of the file to be created, the permissible values of d and m are given, together with their meanings, in the following table.

Value of d	Device	Value of m	Meaning
TTY	Teletype	-	-
DISC or DISK	disc unit	-	new file
		PERM	file already in inventory
Ø (numeric zero)	tape	WRITE	scratch tape, read/write
number n ≥ 7	tape	-	tape reel n, read only
		WRITE	tape reel n, read/write
number 1 ≤ n ≤ 6	CRT	- or LOW	680 character maximum
		MEDIUM	1360 character maximum
		HIGH	2000 character maximum

If greater control over file activation is desired, the user may always use the primitive OPENFILE. OPEN is provided as a convenience, and presupposes values in the macro expansion to OPENFILE. These values are tabulated below:

Unit	Record Size	Internal Buffer Size	Unit Reservation
TTY	1 line	10 words	none
Tape	30 lines	300 words	1 tape drive
DISC	41-51 lines	512 words	1 track

Some examples of the use of OPEN are given below:

Example	Meaning
OPEN(TTYFILE TTY)	activates teletype file TTYFILE
OPEN(MYFILE DISC)	activates local disc file MYFILE
OPEN(YOURFILE DISC PERM)	activates permanent disc file YOURFILE, found in inventory as file YOURFI
OPEN(TAPE 3 1234)	activates read-only tape file TAPE3 on physical reel 1234
OPEN(TAPE2 Ø WRITE)	activates read/write tape file TAPE2 on a scratch tape
OPEN(SCOPEX 5)	activates 680 character CRT file SCOPEX on physical CRT console 5
OPEN(SCOPEX 5 HIGH)	activates 2000 character CRT file SCOPEX on physical CRT console 5

Before opening a file, the user should make sure that the physical file is available. For example, make sure that there are tape units available before opening a tape file, and make sure that a disc file of the proper name is available to the user and on the disc before opening a disc file in PERM mode. If these precautions are not observed, an error return from OPENFILE will be generated.

A disc file name within TSS can consist of up to 6 characters; LISP can use any literal atom. Consequently, if a LISP program uses a file name containing more than six characters, only the first six characters are meaningful to TSS. E.g., DISKØØ1 as a file name in LISP becomes DISCØØ in TSS, and hence is identical, as far as TSS is concerned, with file name DISCØØ2. One final remark concerning disc files is that a permanent disc file can be opened only if it belongs to the user, or is public, or is allowed to the user. It can be read and written by a user if it belongs to him, is allowed to him, or is public and not protected from reading or writing. Any attempt to read a read-protected file or write on a write-protected file, or to open a private file in PERM mode, will result in an error.

The atom *FILES* contains as its value a list of all active file names.

3.2 FILE DEACTIVATION

Whenever a file is no longer needed, it should be deactivated so that its core storage can be returned to the LISP system and the physical file unit, i.e., tape, disc, or CRT, can be released. For temporary disc files, it is essential to deactivate the file in order to add the file name to the user's disc inventory. File deactivation can be accomplished through the macro SHUT.

SHUT (f d)

SHUT is a LISP macro that expands, using the function *SHUTF, into the primitive SHUTFILE described in Section 5. Like OPEN, it returns as its value a list of all currently active file names. Again, f is the name of the file; it is to be deactivated and must previously have been used with OPEN to activate the file. Whenever any file is shut, its buffer storage is reclaimed, and the associated unit is released (physical tape or CRT and reserved disc tracks). The argument d is optional and can be absent; it is only used to specify the disposition of a disc file and is ignored for other than disc files. For disc files only, if d evaluates to the reserved word DELETE, the file is purged from the disc inventory. For disc files only, if d is absent or is not the reserved word DELETE, the file is purged from LISP (its buffer storage is reclaimed); however, it is saved on the disc and the first six characters of the file name f are placed in the user's disc inventory as the name of this file. The file may subsequently be reactivated with OPEN by OPEN (f DISC PERM) where f is the file name.

The user is cautioned that SHUT does not write an end-of-file on either tape or disc: the user must do this himself using POSITION (f WEOF). Note that for simple uses of OPEN and SHUT with disc files (i.e., optional parameter is absent), the user is protected against unintentional disruption or loss of permanent disc files. For example:

OPEN(f DISC)	activates a temporary file
SHUT(f)	saves file as a permanent file

In other words, one must take positive action with both OPEN and SHUT to access or destroy a file, respectively. In this regard, to free disc storage congestion and as a courtesy to other users, always use SHUT(f DELETE) to remove unwanted disc files.

Note further, that if you QUIT LISP with files still active (no SHUT used on them), all CRT, TTY, tape, and temporary disc files are deleted and all permanent disc files are saved. When a disc file is saved, its contents on disc are as of the last time it was written. This means that if no end-of-file was written before the QUIT occurred, the disc file will in general be unreadable.

Note that all permanent disc files created by LISP through the SHUT function are private, write-protected files. To change their mode to public, or to change the protection mode, the user must use the appropriate TSS commands, given directly to the TSS executive, not to LISP.

3.3 FILE SELECTION

Except for CRT files (which are really binary files whose primitives are described in Section 3.6) active files may be selected and a prior file deselected for writing or reading by use of two primitives, WRS and RDS, respectively.

RDS(f)

RDS is entirely analogous to WRS, but for input file selection.

WRS(f)

Argument f is the name of the active file being selected for output. The value of WRS is the name of the active output file being deselected. When a file is selected, the record, line, and column controls for the deselected file are preserved with that file, and the new file record, line, and column controls are reestablished. Thus, WRS may be used with complete freedom at any time, even following partially composed lines.

Note that WRS called by a LISP user affects only the user's program and does not confuse the supervisor; all supervisor outputs are still given on the teletype console. For example:

```
OPEN (OUTTAP Ø WRITE)          opens scratch tape under name OUTTAP
$ WAIT                         }
$FILE OUTTAP DRIVE 12 REEL ØØØØ}   TSS replies
(OUTTAP *TTY)                  LISP reply, value of OPEN
WRS (OUTTAP)                   User write selection
*TTY                           Value of WRS
(LAMBDA (A)
  (PROG () (PRINT A) (RETURN (FIRST A))))   }
((NOW IS THE TIME FOR ALL GOOD MEN TO COME  }  User's program for test
 TO THE AID OF THEIR PARTY))                }
NOW                            LISP supervisor reply when tape has been
                               written with (NOW IS THE ... PARTY)
WRS (*TTY)                     User write selection
OUTTAP                         Value of WRS
```

Note: If a file is shut while selected for either read or write, the selection reverts to the NIL file, a teletype file used by ERROR.

3.4 FILE CONTROL

POSITION (f m)

POSITION is a LISP 1.5 function that is useful for controlling active tape and disc files; it acts as a NOP for CRT and TTY files. Argument f is the name of an active file, and argument m is a reserved word describing the action desired. These actions are tabulated below. (Note that any other value of m does (TEREAD) on the file f.)

m	Tape Action	Disc Action
SKIPR	Position file to first line next record.	Position file to first line of next sector.
SKIPF	Position file to first line of next file.	Position file to the end-of-file and erase end-of-file.
WEOF	Write tape end-of-file.	Write disc end-of-file.
WEOT	Write tape end-of-tape.	Same as WEOF.
REWIND	Position file to first line of first file.	Position file to first line of file.
BACKR	Position file to first line of prior record.	Position file to first line of prior sector.
BACKF	Position file to first line of prior file.	Same as REWIND.

For POSITION, tape records and disc sectors (there are eight, 512-word sectors per disc track) are treated alike. Furthermore, only one physical file may exist in a logical disc file, whereas multiple physical files may exist in a logical tape file.

POSITION returns one of four possible values according to the action taken:

1. f is returned if m is not a SKIPR or SKIPF. If m is SKIPR or SKIPF, the following values derive.
2. A positive integer, representing the number of records (or sectors) skipped (not counting an end-of-file record).
3. The atom EOF if the next record on tape (or the current sector on disc) is an end-of-file.
4. The atom EOT if the next tape record is an end-of-tape.

Note: You can never POSITION or READ past the end-of-file on a disc file since a disc file always contains only one physical file. You are permitted to do so with tape files, at your own risk, since they may contain multiple physical files.

TABIN(n), TABOUT(n)

These primitives, TABIN and TABOUT, may be used for format reading and writing, respectively. They advance (forward or backwards) the column pointer of the selected file to column n, where n is a positive integer less than 71. (For n outside the range $1 \leq n \leq 70$, a value of n = 1 is assumed.) For example, TABOUT(33), will tab tne output line to column 33, and subsequent printing will begin at that column.

Both primitives return a positive integer value, corresponding to the column number prior to the tab.

TABIN and TABOUT do not influence the data content of any line, and may be used to backup or skip forward within a line without disturbing the contents.

3.5 I/O PRIMITIVE CHANGES

To accommodate the new I/O mechanisms, a number of existing primitives have been modified in their side-effects only.

TERPRI ( )

TERPRI now performs as follows:

1. Write an end-of-record mark in the current line of the selected output file.
2. Move the current line to the next available "slot" in the internal record of the selected output file.
3. Write the internal record of the selected output file onto the associated external unit.
4. Clear the current line to blanks and reset the column control to 1 for the selected output file.
5. For tape files, position file to the next record. For disc files, position file to the next sector only if the current sector is full.

Thus, TERPRI prints each teletypewriter line it sees; it writes blocked tape records of variable length, up to 30 lines per record, whenever it is called; and it writes blocked disc records, packed fully to 41 lines per record, whenever evaluated. A special TERPRI, called STERPRI, is available that always blocks maximum size records, thus speeding production runs.

STERPRI ( )

STERPRI performs as follows:

1. Move the current line to the next available slot in the internal record of the selected output file.

2. If, and only if, the internal record is full, write it out on the associated external unit.

3. Clear the current line to blanks and reset the column control to 1 for the selected output file.

Note: Whenever STERPRI has been used in output formatting, TERPRI must be called finally to print the last partial record from the output buffer.

*RATOM ( )

The basic atom read primitive *RATOM is used by all read functions, and it has been modified to read from the selected input file. End-of-medium conditions are flagged by *RATOM as follows:

Unit	Condition	*RATOM Effect
TTY	end-of-line (carriage return)	ring bell for more input
DISC	end-of-file	return literal atom EOF
tape	end-of-file	return literal atom EOF
tape	end-of-tape	return literal atom EOT

READ, READ1 and READCH which use *RATOM, read from the selected input file. TEREAD also works on the selected input file.

PRIN1(a)

The basic atom print primitive, PRIN1 is used by all print functions and it has been modified to write in the selected output file. Disc files will automatically grow larger, by one track, whenever printing overflows the last sector (subject to the availability of free disc tracks). PRINØ, which uses PRIN1, similarly works on the selected output file, as does PRINCH().

3.6 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. Scope files are opened just as other files; however, scope files are never selected by WRS or RDS but rather use the following primitives:

PLOT (f i s)

Procedure PLOT inserts a vector or character specified in list i into the file f based on a search criterion specified in list s. The search is made by comparing for equality all non-NIL items in s with data already in file f. If s = NIL, search is for first available word in file f. If the search fails, the value of the procedure is NIL. If successful, the value of the procedure is a list ℓ of the contents of the search match word of file f. The form of i, s, and ℓ is:

((x.Δx) (y.Δy) (char . size) id)

where;

x and y must satisfy $0 \leq x, y \leq 1023$

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

char = a character atom

size must lie in the range Ø - 3 where Ø designates the smallest character size, and 3 indicates the largest size character (size is ignored 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 = the character atom blank ('♭).

Note 2: To display data, PRINTCRT must be used. The scope number in OPEN 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 file BUFF1 there is found on Scope 3 (or 6) an E at any column of 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 (s = NIL), then the vector would be inserted at the first empty entry, i.e., a search for first NIL entry of the file.

PRINTCRT (f)

PRINTCRT dumps the file named f on 0-32 drums for display on the SDC scope. The value of the procedure is NIL if print is not possible; otherwise the value is f.

READCRT (f)

READCRT is used to obtain a light-pen input from LISP file f. After the user lightpens a character or vector on the CRT, READCRT returns a list of the information describing that character or vector in the same format used for PLOT i or s; i.e., READCRT returns a list of the form

((x . Δx) (y . Δy) (char . size) id) .

3.7 LIBRARY FUNCTIONS

All of the library functions of Mod. 2.5 have disappeared from Mod. 2.6. As a result of availability of the LISPED* program for file editing and manipulation, we have found that most library functions are more readily performed with LISPED. The following three functions have been added to Mod. 2.6 to facilitate the loading (operation) of LISPED files into LISP:

LOADEXP (filename)

Performs loading (operation) of a single LISPED file (one S-expression in the format

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

where name is an arbitrary name, and f_i a_i are function and arguments constituting an Evalquote pair)

filename is the name used in OPEN.

(Note that the file must be opened before LOADEXP is called.) LOADEXP does its own read selection using RDS. LOADEXP prints the value of each Evalquote pair on the user's console, and returns name.

READFILE (filename)

Reads all S-expressions on the file designated by filename, and returns a list of the expressions read. READFILE does its own read selection using RDS.

PRINTFILE (filename list)

Prints onto the file filename each S-expression in list, and then writes an end-of-file on the file filename. PRINTFILE does its own write selection using WRS.

---

* LISPED is described in TM-2337/100/01 .

4. SYSTEM MODIFICATIONS

The LISP 1.5 Mod. 2.6 operating system consists of an initialization package and a supervisor which differ from those in the previous version of LISP.

4.1 INITIALIZATION

The initialization package, called whenever the user says GO after loading, consists of the function "IOINIT(), which

1. Cleans up the system by removing all previous file declarations and file buffers from LISP memory, by using the function *IOSHUT to delete all files found as the value of *FILES*.

2. Makes a file declaration for the file *TTY that is used by the supervisor; sets *TTY to the value (QUOTE *TTY); and sets the variables *TRDS and *TWRS, which govern the input and output from the supervisor, to *TTY.

4.2 SUPERVISOR

In general, the LISP 1.5 Mod. 2.6 supervisor performs the same function as the LISP 1.5 Mod. 2.5 supervisor, and allows the same input flexibility, including the use of *FUNC to refer to the last top-level function compiled into scratch program area. However, the operation is smoother and more efficient, and the functions *DEFQ, and *MGSYM are not used.

*SUPV

*SUPV() is the LISP supervisor, which does a RDS (*TTY) and WRS (*TTY) and thereafter does a loop using *TEVALQT.

```
(*SUPV (LAMBDA () (PROG (X Y)

  (RDS *TTY)

  (WRS *TTY)

A (TEREAD)

  (SETQ X (READ))

  (SETQ X (READ))

  (SETQ Y (READ))

  (TEREAD)

  (PRINT (*TEVALQT X Y)) (GO A))))
```

*TEVALQT (x y)

*TEVALQT performs housekeeping of scratch program space and of read-selection and write-selection and calls *EVALQT as follows:

(*EVALQT X Y (QUOTE *FUNC))

*EVALQT (x y n)

*EVALQT works as it did in Mod. 2.5, except that whenever the third argument is identically (QUOTE *FUNC), a sweep is made through the atom head and quote cell area, removing the bindings of all atoms that point into the scratch program area, and removing all temporary quote cells.

4.3 GROWING PAINS

In LISP 1.5 Mod. 2.6, the higher address boundary of binary program space (BPS) is not fixed, but is moved by LAP whenever a new program being compiled at any time would otherwise overflow the current boundary of BPS. This "growing pain" is accomplished by the function FIXEM, which relocates all full word space structures toward higher core addresses, and adjusts all code references to full word space and binary program space boundaries appropriately.

The amount by which BPS is adjusted on each application of FIXEM is determined by the value of the atom *BUMP. The value of *BUMP in the standard LISP system is 1Q3 or 512. In assembling a very long function, LAP may call FIXEM repeatedly.

Whenever FIXEM is called, the message

(n IS NEW TBPS)

is printed on the teletype.

Since the system boundaries change continuously, and free space may be used for binary programs, full word storage, or list storage, the user can find out the amount of available space by calling the function FREESPACE ().

FREESPACE ()

FREESPACE () calls the garbage collector, then computes the amount of available free storage, and prints the result as a decimal integer.

The value of FREESPACE () in a clean LISP system is approximately 29,000 cells.

5. PRIMITIVES

OPENFILE (f b)

OPENFILE creates a file whose name is the first six characters of the literal atom f. OPENFILE returns f as its value. Argument b is a list of property-value pairs describing the file. The requisite pair information is tabulated below.

Property-Value Pairs for OPENFILE

Properties	TTY	Tape	Disc	CRT
UNIT	8	3	11	10
FORM	19 (BCD)	18 (binary) 19 (sortable)	18 (binary) 19 (sortable)	18 (binary)
SIZE	10	10n($1 \leq n \leq 30$),(BCD) variable,(binary)	4096	680n($1 \leq n \leq 3$)
ID	*	reel no.	*	$1 \leq$ SCOPE NO ≤ 6
MODE	*	NIL (read-only) T (read/write)	NIL(permanent) T (temporary)	*
BUFF	GENSYM-named buffer supplied by OPENFILE			
IO	OPENFILE places the list of above property value pairs as the value of property IO on the property list of file.			

* Property-value pair not required

OPENFILE does four things:

1. It creates an internal, GENSYM-named buffer of specified size, with all the necessary control mechanisms for the unit specified.

2. It sets up all the necessary I/O communication linkages with the time-sharing monitor.

3. It attaches list b (augmented by the property-value pair "BUFF" and GENSYM-named buffer) as the value of property "IO" on the property list of f.

4. It returns a list of all file names opened to date.

Note: For DISC, a MODE value of NIL (permanent) means that the specified file already exists as a permanent disc file in the time-sharing disc-file inventory. T(temporary) means that the specified file does not already exist and must be created. Disc files should never exceed one track (4096 words), since for reading, only one sector (512 words) is ever in core at any moment. For printing files larger than one track (408 lines), the LISP print programs will automatically extend the file size by one track as needed. Therefore, be frugal with disc size requests. For example, OPENFILE (TAPE1 (UNIT 3 FORM 23Q SIZE 300 ID 1234 MODE NIL)) would create a read-only (MODE=NIL), 300 word (SIZE=300), sortable (FORM=23Q), tape (UNIT=3) file, on reel 1234 (ID=1234) named TAPE1. OPEN(TAPE1 1234) would achieve the same result.

SHUTFILE (f b)

SHUTFILE deactivates a file. It returns as its value a list of all file names still opened. It purges the file, whose name is the first six characters of the literal atom f, from both the LISP system and the time-sharing I/O communication tables, and dismisses the unit reserved by this file. The property IO is removed from the property list of the atom f. The argument b is the disposition mode for the file f and only has significance for disc files. If b = NIL (permanent), the file f will be deleted from the LISP system; but the name and contents of the file will be added to the permanent time-sharing disc inventory. The contents of this file will be exactly as they were at the time of the last TERPRI. If b = T(temporary), the file f will be deleted from both the LISP system and the time-sharing disc inventory, regardless of the OPENFILE MODE value.

Therefore:

1. Always evaluate TERPRI prior to SHUTFILE to dump any leftovers in the file on to the external unit. Then write an end-of-file with POSITION.

2. Permanent disc files may be deleted from the disc by SHUTFILE with b = T (temporary).

3. Temporary disc files may be made permanent on the disc by SHUTFILE with b = NIL (permanent). (Note: Q-32 time-sharing permanent disc files will be automatically deleted by the time-sharing system after about two days of dormant residence.)

4. For b = NIL (permanent), SHUTFILE may request another file name under which to save this file. This may be necessary because of a name conflict with other users' permanent files in the time-sharing disc inventory. In such cases, all future LISP references to this file must use the new name.

(TNSTAT)

TNSTAT (transfer status) returns as its value the status code (an integer) of the last I/O transfer according to the schedule below. It is useful for error detection and for distinguishing the terminating condition with READCH, or any READ primitive.

Status Code	Condition
≤ 3	End-of-line
4	End-of-file
5	End-of-tape
≥ 6	Transfer errors

Note that (TNSTAT) can meaningfully be used only inside a program, never at the top level of Evalquote, since the last I/O transfer seen by Evalquote is always a teletype read.

SQUOZE (ℓ)

SQUOZE is a primitive used by function COMPRESS. The value of SQUOZE is a literal atom formed from the list of character atoms ℓ. SQUOZE is undefined if ℓ is anything other than a list of character atoms.

READC (f b)

READC is a primitive used by EXPLODE. The value of READC is the character atom at the bth byte (starting with Ø) of the print name of literal atom f. If b exceeds the maximum byte of the print name of f, READC returns the value NIL.

6. ERROR MESSAGES

The error messages given in Sections 6.1 and 6.2 are LISP error messages which result in a LISP unwind, so that in general the user can continue his operations normally. The time-sharing error messages given in Section 6.3 require special action, as discussed therein:

6.1 INPUT/OUTPUT ERROR MESSAGES

Message	Meaning
NO FILE	Read or write functions do not find an internal buffer for the currently selected I/O unit. This buffer is created and attached to the property list of the file by OPEN (OPENFILE) .
NO SCOPE	The CRT is currently not available as a selectable I/O unit with WRS or RDS. The SCOPE may be used with functions READCRT, PRINTCRT, and PLOT.
UNITERR	Most probably caused by reading or writing a tape beyond EOT, or beyond last EOF. Also induced by trying to read a binary tape in Hollerith mode. May also be induced by a disc or unit malfunction.
UNITBUSY	Unit requested by OPEN (OPENFILE)--tape, disc, or CRT--is in use by others and temporarily unavailable. Type !TAPES (cr) or !TRACKS (cr) to see current time-sharing unit availability. Query the system and try again if either query shows unit availability.
FILEGONE	Permanent disc file named with OPEN (OPENFILE) or SHUT (SHUTFILE) cannot be found in the time-sharing disc file inventory. Either file name is in error, or file has been deleted by periodic time-sharing system purging of overloaded disc.
PREEMPTED DISC NAME - ENTER ANOTHER NAME	The temporary file named with OPEN (OPENFILE) cannot be saved under that name by SHUT (SHUTFILE) as the name conflicts with an existing disc file. Enter another file name for this file following this message. SHUT (SHUTFILE) will repeat the request until successful.
(x NOT OPENFILED)	File name x has been used with WRS or RDS but has not been opened previously with OPEN OPENFILE).

(UNIT NOT SPECIFIED)
RDS or WRS has been unable to find "UNIT" on the value list of property "IO" on the property list of the named file. See OPEN (OPENFILE).

(x HAS NO BUFF)
RDS or WRS has been unable to find "BUFF" on the value list of property "IO" on the property list of file x. See OPEN (OPENFILE).

6.2 OTHER LISP ERROR MESSAGES

(CAR NIL UNDEFINED)
CAR of a character atom is undefined.

(CDR NIL UNDEFINED)
CDR of a character atom is undefined.

(EXPLODE x UNDEFINED)
EXPLODE error if x is a nonatomic S-expression.

(COMPRESS x UNDEFINED)
COMPRESS error if x is a list of other than character atoms.

(x REDUNDANT FILE NAME)
A file with the same file name has already been opened by OPEN (OPENFILE). Try another file name.

x NOT BOUND AS FN
This usually means that x has been called as a function without having been defined. Correct the condition and try again.

x SETQ'ED - NOT BOUND
This usually means that an attempt was made to perform (SETQ x expression) where x was neither bound, nor CSET previously. To use a variable free, it must be given a CSET (or CSETQ) previously. SPECIAL((x)) will not solve the problem, but CSET (x NIL) will.

(COND ERROR A3)
COND used in expression context had no true clause, and evaluation "fell through" at run time.

((PAIR ERROR F2) x y)
((PAIR ERROR F3) x y)
The function PAIR was called with lists of unequal length.

(x NOT AN ATOM (CSET))
First argument given to CSETQ was not a literal atom, or first argument of CSET did not evaluate to an atom.

(OUT OF SCRATCH)
LAMBDA-expression given to Evalquote was too long to compile into scratch program area: define the function, then call it.

(x NOT A NUMBER)
An attempt was made to use a non-numeric argument for an arithmetic function.

(x NOT A LABEL (COMPROG))
The expression (GO x) was encountered in a PROG where x was not used as a label. The function must be corrected and redefined.

(SET ILLEGAL)
SET is not defined in Q-32 LISP 1.5.

(x NOT FUNCTION)
x is an expression other than (LAMBDA ...) or (LABEL ... (LAMBDA)), used in a context where a LAMBDA-expression is called for, e.g.

```
DEFINE (((A B)))
```

will not work, although

```
DEFINE (((A (LAMBDA (Y) (B Y)))))
```

will.

(x NOT DECLARED)
This message is printed by the compiler during compilation, as a warning only. It means that x was used free without having been declared SPECIAL before compilation, or was used as a functional argument without being preceded by the word FUNCTION. This is the only compilation error message which does not prevent completion of compilation, since the compiled code may be correct.

For example:

```
(LAMBDA (X) (MAPCAR X ADD1)) ((Ø 1 2))
```

will work, but the message:

```
(ADD1 NOT DECLARED)
```

will occur.

```
(LAMBDA (X) (MAPCAR X (FUNCTION ADD1))) ((Ø 1 2))
```

will not produce the error message.

Similarly,

```
DEFINE ((
  (AA (LAMBDA (X) (CSETQ B X)))
  (BB (LAMBDA (Y) (CONS (AA Y) B)))))
```

will produce the error message:

```
(B NOT DECLARED)
```

during compilation, even though the function BB cannot cause an error.

RESCUE n

The meaning of this error message is given in the following table:

n	Meaning	Remarks
1	Illegal branch to FIX program	(1)
2	Illegal instruction	(1)
3	Illegal address reference	(2)
4	Illegal division	Attempt to divide by zero
5	Program halt	(1)
6	BCH ZERO - tape write	Should never occur unless LISP is bad
7	Wrong mode - tape read	Wrong tape mounted?
8	Memory protect violation	(1)
9	Illegal PER instruction	(1)
10 or *	I/O trap	(1)
11 or =	Attempt to store into input memory	(1)
12 or '	Memory protect register or Interrupt content register store	(1)
13 or "	Manual Break action	Break key or !STOP used

(1) These rescue messages should never occur unless the user has used LAP incorrectly, or unless the system has been damaged in some way.

(2) This message will not usually appear. Instead, LISP will print out one of the two messages,

x SETQ'ED - NOT BOUND or

x NOT BOUND AS FUNCTION

depending upon whether an illegal store or an illegal indirect jump was encountered. x is usually an atom name.

6.3 TIME-SHARING ERROR MESSAGES

The following time-sharing error messages may be encountered by the user due to his attempt to shut or write on a write-protected public disc file which is not his own, or to read a read-protected disc file which is not his own.

```
$      NO FILE       LISP    ØØØnnnnn     DISPATCHER CALL ERROR ****

$     BAD MOVE       LISP    ØØØnnnnn     DISPATCHER CALL ERROR ****
```

The program will at this point be trapped by the Time-Sharing System. To extricate himself from this situation, the user will then have to reload or else execute the following set of time-sharing commands.

```
!$LIV=4ØØØ2'* cr      reset live register

$MSG  IN.             TSS reply

GO                    "go" command to TSS
```

LISP will output two bells when it is ready to receive more input. If the user does not receive two bells he must reload to continue.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

System Development Corporation
Santa Monica, California

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE

THE Q-32 LISP 1.5 MOD. 2.6 SYSTEM Operating System, Input-Output, File, and Library Functions.

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(Last name, first name, initial)*

Kameny, S. L. and C. Weissman.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
11 April 1966	29	

8a. CONTRACT OR GRANT NO. AF 19 (628)-5166
b. PROJECT NO. Electronic Systems Division, Air Force Systems Command, in performance of ARPA Order 773 for the Advanced
c. Research Projects Agency Information Processing Techniques Office.
d.

9a. ORIGINATOR'S REPORT NUMBER(S)
TM-2337/103/00

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. AVAILABILITY/LIMITATION NOTICES

This document has been cleared for open publication and may be disseminated by the Clearing House for Federal Scientific & Technical Information.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

13. 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. 2.6 It also describes differences between Mod. 2.6 and the previous Mod. 2. 5 describes in TM-2337/102/00 dated 9 August 1965.

TM-2337/101/00 - AD - 622 022
TM-2337/102/00 - AD - 622 018

DD FORM 1 JAN 64 1473

Unclassified

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
AN/FSQ-32 LISP Input/Output Library Functions						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS), (S), (C), or (U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.