The research reported in this paper was sponsored in part by the Advanced Research Projects Agency Information Processing Techniques Office and was monitored by the Electronic Systems Division, Air Force Systems Command under contract F196287C0004 Information Processing Techniques, with the System Development Corporation.

TECH MEMO *a working paper*

System Development Corporation / 2500 Colorado Avenue / Santa Monica, California 90406
Information International Inc. / 11161 Pico Boulevard / Los Angeles, California 90064

TM-3417/600/00

AUTHOR Donna Firth
Donna Firth

TECHNICAL Jeff Barnett
J. Barnett

RELEASE Clark Weissman
C. Weissman, S.D.C.
B. Anschultz, I.I.I.

for J. I. Schwartz

DATE 4/26/67 PAGE 1 OF 23 PAGES

(Page 2 is blank)

LISP 2 Input/Output Specifications

Abstract

This document describes the capabilities required to perform input/output functions in the LISP 2 system being designed for the IBM 360/65 computer. It includes definitions of logical units and files; a discussion of file properties; a list of standard input/output functions; and specification of the monitor capabilities required for LISP 2 input/output.

TABLE OF CONTENTS

			Page
Section 1.		Introduction	1
	1.1	Overview	1
	1.2	Basic Mechanisms--Units and Files	5
	1.3	Design Goals	6
	2.	Units	9
	3.	Files	12
	3.1	Permanent Properties of Files	13
	3.1.1	Name	13
	3.1.2	Line Length	13
	3.1.3	Buffer Size	13
	3.2	Changeable Properties of Files	13
	3.2.1	Left and Right Margins	13
	3.2.2	Right Margin Overflow Action	13
	3.2.3	Page Size	14
	3.2.4	Code Form	14
	3.2.5	Unit Connection	14
	4.	User Input/Output Functions	14
	4.1	Functions Affecting Files	14
	4.1.1	Functions Applicable to Any File	14
	4.1.1.1	OPEN$LISP	14
	4.1.1.2	INPUT$LISP and OUTPUT$LISP	16
	4.1.1.3	CHANGE$LISP	17
	4.1.1.4	SHUT$LISP	18
	4.1.1.5	REWIND$LISP	19
	4.1.1.6	WRITEOF$LISP	19
	4.1.1.7	ADVLINE$LISP and TABCOL$LISP	19
	4.1.2	Functions Applicable to the Currently Selected Input File	19
	4.1.2.1	READCH$LISP	19
	4.1.2.2	READ$LISP	20
	4.1.2.3	IADVL$LISP and ITABCOL$LISP	20
	4.1.3	Functions Applicable to the Currently Selected Output File	20
	4.1.3.1	PRINCH$LISP	20
	4.1.3.2	PRIN$LISP	20
	4.1.3.3	OADVL$LISP and OTABCOL$LISP	20

TABLE OF CONTENTS (Cont.)

		Page
Section 4.2	Functions Affecting Units	21
5.	Monitor Capabilities Required for LISP 2 Input/Output	22
5.1	Minimum I/O Device Configuration	22
5.2	Hardware Restrictions	22
5.3	Non-Parallel I/O Mode With Sequential File Organization	22
5.3.1	Sequential Devices	22
5.3.2	Random Devices	23
5.3.3	Display Devices	23
5.4	Parallel I/O Mode or Non-Sequential File Organization	23

Fig. 1	General Input Scheme	7
Fig. 2	General Output Scheme	8

Table 1	Default Values for Optional Unit Properties	10
Table 2	Possible Values for Unit Properties	11
Table 3	File Properties, Names, and Value Types	15
Table 4	CHANGE Function Property Key Words	17
Table 5	SHUT Function Disposition Parameters	18

1. INTRODUCTION

1.1 OVERVIEW

The descriptions given here are implementation-independent and of a general nature. Detailed programming descriptions of the variables, structures, and functions used for input/output are not provided. Any restrictions caused by a specific implementation will be announced in an amendment document for that implementation, which will also include detailed program descriptions.

The purpose of input/output in LISP 2 is to exchange blocks of information between main memory and other storage devices. Although many operations are involved in this process, the goals of LISP 2 I/O are to minimize user effort, provide maximum flexibility, and utilize all available resources.

The input of information is divided into three independent areas, as shown in Figure 1. The result of processing in any area is used by the system for more than one purpose. The master functions that produce results from these areas are TOKEN, READCH, and READU. The function READ, a higher-level function, is also provided for general data reading. Areas 1 and 2 constitute the I/O portion of the LISP system. Area 3 is completely independent and may in fact operate on characters from other sources.

The output of information follows an equivalent scheme. Internal data structures are represented as character sequences; the character sequences are put into file buffers; and the file buffers are transferred as physical records to auxiliary storage. The primary difference is that there is no explicit equivalent of Area 3 on the output side.

1.2 BASIC MECHANISMS--UNITS AND FILES

Actual hardware devices, each with its own peculiarities, exist in the computer environment. Since LISP 2 operates within the environment of an executive or monitor program (the capabilities of which are specified in Section 5), the actual devices are not of concern. Instead, something to be called a unit is regarded as the basic input/output entity in the LISP 2 system.

A unit is roughly defined as a particular hardware device as seen through the monitor system. The characteristics of units are completely dependent on what the monitor will allow and are usually a subset of the features of the physical device. Within LISP 2, separate units are established for each tape reel, on-line typewriter, display, connected portion of disc[1] storage, etc., that the user wants to access. Internally, a unit is a collection of information, including the type and state of the device but not including any space for records of the unit.

There are two types of I/O available to the user. The first type will be called binary; it involves the direct transfer of blocks of computer words without additional processing by the I/O routines. The second type will be called printable; it requires the I/O routines to process streams of characters.

---

[1] Usually called a disc file in monitor terminology.

Binary I/O is quite simply handled; the blocks of words are moved between a unit and main memory in a manner chosen by the user. Printable I/O requires more complicated handling since data formatting is desired. While each unit may have different properties, it is not desirable (from the user point of view) to have to use different functions to read or write the different units. To achieve unit independence and formatting capabilities, LISP 2 uses files for printable I/O.

A file is roughly defined as an extra level of organization imposed between a unit and a user which permits more convenient character-oriented input/output. Depending on its declared organization, a file may reference part or all of the information on the unit it is connected to. (For example, a tape reel declared as a unit may contain many physical files, while the LISP 2 file connected to this unit might reference only one of them.) Internally, a file is a collection of information describing all its properties and including space for records on the unit it is connected to.

As many files as the user desires may be declared within LISP, but only one file at a time may be read or written. When a file is to be read or written it is selected for that purpose and becomes the currently selected input or output file; all the primitives for reading or printing implicitly reference the currently selected file. Each file--whether selected or not--is connected to exactly one unit, although the unit connection is changeable whenever desired.

When "standard" I/O operations (such as reading and printing data) are attempted, only files need be considered by the user, since the associated units will be automatically dealt with by the I/O functions.

The exact nature of units and files will be discussed below in Sections 2 and 3 respectively. The user functions that establish, change, and delete units and files are described in Section 4.

1.3 DESIGN GOALS

The main reason for the existence of units is convenience of implementation. The machine- and device-dependent parts of the LISP 2 I/O package are concentrated in the functions concerned with units. A unit is the object of a command to move to (or from) auxiliary storage a certain number of computer words; no pre- or post-processing of any kind is done on these words by unit-oriented functions. The words handled by units are regarded as raw information and are not interpreted by the units. Extracting meaning from this information is done through the use of files. In particular, translation of codes--ASCII to BCD, EBCDIC to ASCII--and data formatting are done through files. The file operations are machine-independent and need not be rewritten for different implementations.

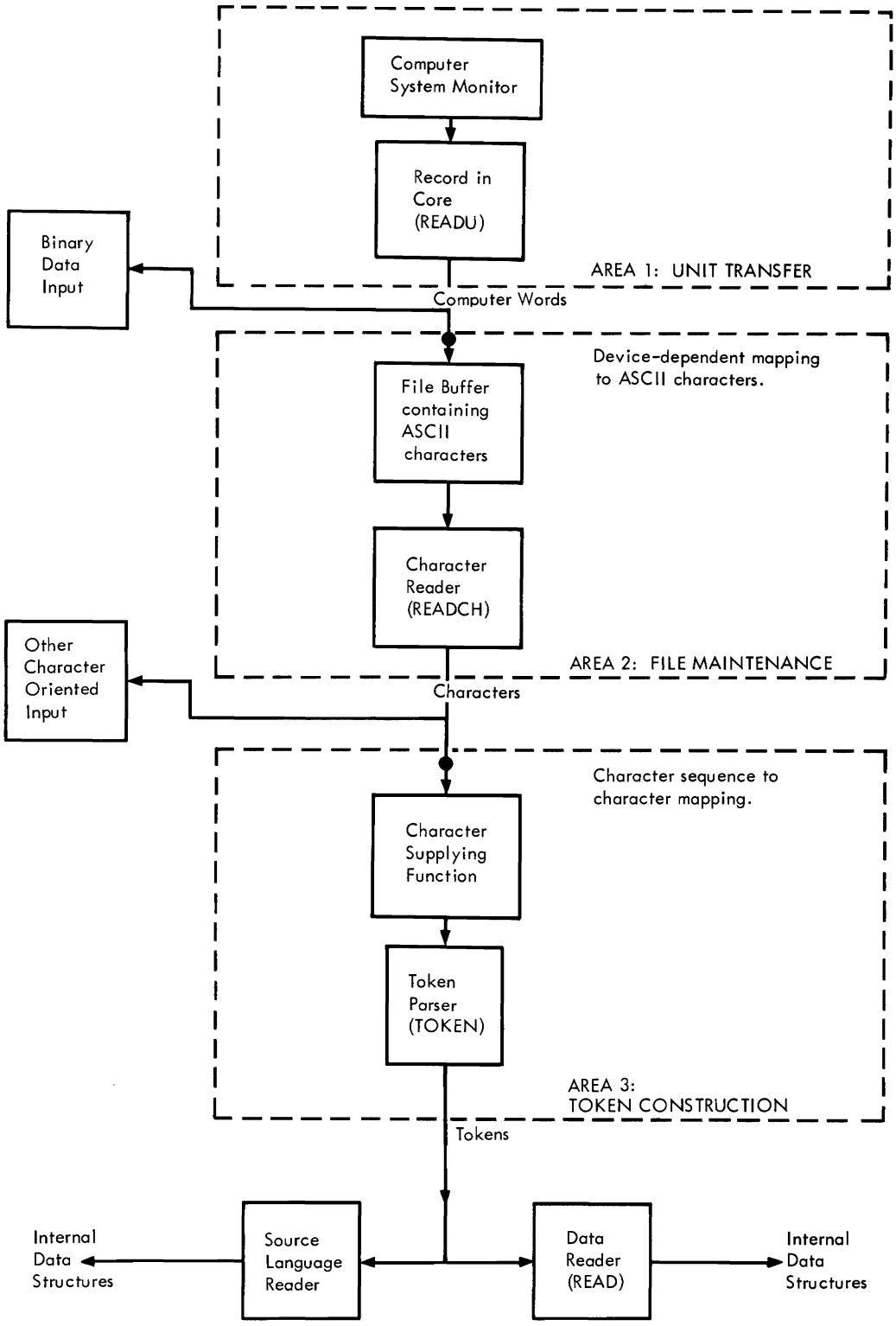

Figure 1. General Input Scheme

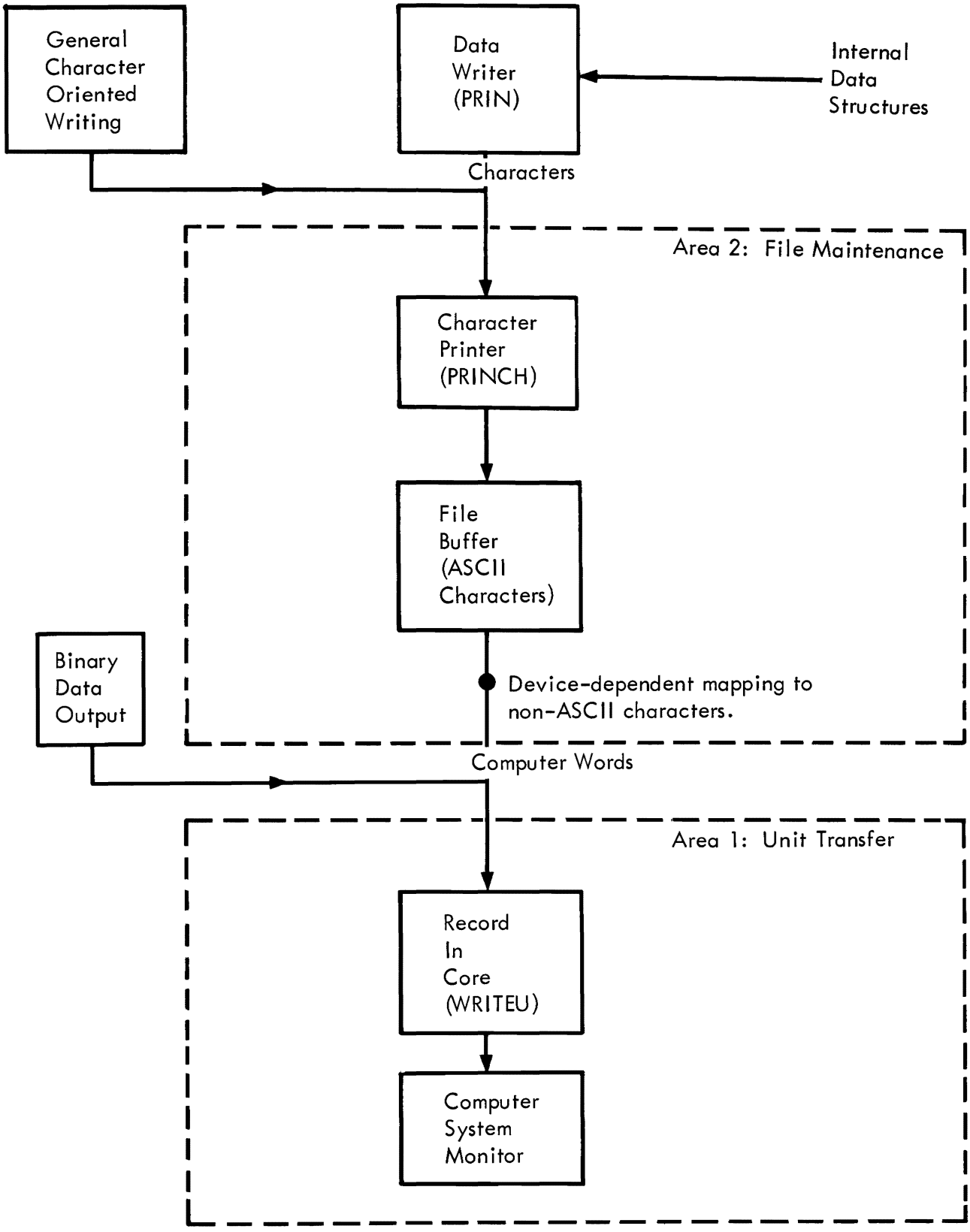

Figure 2. General Output Scheme

For some kinds of I/O operations, efficiency in execution can be obtained by having separate units and files. An example of this is copying physical files of data. If a LISP file is used for this purpose, it is necessary to parse the data as it is read and written. Parsing consumes execution time and free storage. By using units for the copying operation, no parsing is required. The records from the input unit can be read into memory and directly moved to the output unit.

Although core is included as a type of unit, it has none of the features of other devices and exists primarily so that files, which in effect are not connected to any device, may be used within LISP.

2. UNITS

A unit declaration within the LISP 2 system activates a previously inactive device. Once activated, a unit may be read or written by invoking the functions appropriate for the type of I/O (binary or printable) desired on that unit. In the case of printable I/O, a file must also be declared and selected before the unit becomes useful.

When a unit is declared, there are two properties that must be given by the user and several that are optional. The required properties are the unit name and device, where the device specification consists of the type--such as 7-track tape, disc, etc.--and a further specification for reel number or disc identification (as appropriate).

The optional properties are parity, protection, organization, transfer mode, and end-of-medium action. They have default values when they are not originally specified.[2] The optional properties may be changed while the unit is active, but the required properties are permanent as long as the unit is active.

Units may be directly positioned backward and forward or written with physical file marks by using the functions for that purpose. When a file is connected to a unit, positioning and physical file marks are ordinarily indirectly obtained through operations on the file.

The system maintains additional information concerning a unit, which includes status of the last I/O transfer and current physical position. This information may be examined by the user but should never be directly reset.

---

[2] The default values depend on the type of device and are given in Table 1.

Table 1. Default Values for Optional Unit Properties

	Property				
Device Type	Parity	Protection	End-of-Medium Action	Physical Record Organization	Record Transfer Mode
teletype	printable	none	no-operation	sequential	wait
7-track tape	binary	none	locked	sequential	wait
9-track tape	printable	none	locked	sequential	wait
disc	printable	none	locked	sequential	wait
display	binary	none	no-operation	sequential	wait
core	printable	none	locked	sequential	wait

Table 2. Possible Values for Unit Properties

Property	Values
Name	An identifier not in use as the name of another active unit
Device Type	One of the identifiers TTY, TAPE7, TAPE9, DISC, DISPLAY, CORE
Device Sub-specification	When type is TTY--either the channel number of the teletype or nothing (meaning the user's primary teletype) When type is TAPE7 or TAPE9--either a reel identification or nothing (meaning a scratch tape) and optionally a physical file number [3] When type is DISC--either a monitor disc inventory identification or nothing (meaning the unit name will be used for this purpose) and optionally, a starting position designation. When type is DISPLAY--the display identification (usually a number) When type is CORE--no sub-specification necessary
Parity	One of the identifiers BINARY or PRINT
Protection	Either, both, or none of the identifiers READ and WRITE
End-of-Medium	The name of the function to be executed if EOM occurs
Physical Record Organization	Options are implementation-dependent
Record Transfer Mode	Options are implementation-dependent

[3] Although it is legal to declare different physical files of a tape as different units, it is not advisable if the units are to be used concurrently.

3. FILES

A file is a channel for printable information. It either supplies or absorbs characters, depending on whether it is being read or written. The same file can be used for either process by selecting it for the operation desired. In actuality, character handling is not so interesting to users as is the processing of data of a language (such as S-expressions). The functions to handle such data (e.g., READ and PRINT for S-expressions) will rely on the primitive character handling functions to achieve their intended effects. To describe files, then, all that is necessary is a description of their properties and a definition of the effect of the character functions, READCH and PRINCH, on the state of a file.

The permanent properties of a file are its name, line length, and buffer size[4]. The changeable properties are left margin, right margin, right margin overflow action, page size[5], top of page, bottom of page, bottom of page overflow action, code form, and unit connection. These properties are changeable in that they may receive different values during the life of the file by executing the appropriate functions. New values may not take effect immediately (see the function descriptions in Section 4 which describe when a change becomes effective).

All changeable properties will have a default if they are not given when a file is established. Permanent properties have no default value; hence, they must be specified when a file is established. Some standard file descriptions are available within the system, so it is not necessary to manufacture a new description for every file.

The system maintains some additional information for each file. This includes buffer location, buffer or file status, physical position information, variables used by the language functions, number of characters in the buffer, and a group of column and line index quantities. These indices specify line number within a buffer, line number within a page, and current column of the line. The index quantities are automatically updated as a file is read or printed and should never be directly accessed by the user. Functions for line and column positioning are available and should be used for this purpose. Positioning within a file is usually only possible in a forward direction.

---

[4] Buffer size corresponds to record length for the unit connected to the file. Only one record of a file will be accessible at a time.

[5] Page size is independent of buffer size.

3.1 PERMANENT PROPERTIES OF FILES

3.1.1 Name

This is a LISP identifier not currently in use as the name of another file. All further references to the file after it has been declared use this name either explicitly or implicitly. The implicit uses occur after a file has been selected.

3.1.2 Line Length

This is specified by number of characters per line. Although a file can be thought of and used as a sequence of characters, it is sometimes necessary or convenient to format the file into lines. Lines actually have physical meaning for many storage devices, and information will be lost if the identical line length is not used for the file. When a line format is not wanted, the line length should be given the same value as buffer size.

3.1.3 Buffer Size

Corresponding to the size of the maximum physical record on the unit connected to the file, buffer size is specified in lines per record. If S is the product of characters-per-line and lines-per-record, and R is record size, then information will be lost when a unit connected to this file supplies records of $R > S$ size.

3.2 CHANGEABLE PROPERTIES OF FILES

3.2.1 Left and Right Margins

When lines are being used, it is possible to set margins on the line. The settings are given by column (character) number. The following restrictions will be enforced at the time of specification:

(1) $1 \leq$ left margin $<$ line length

(2) $1 <$ right margin $\leq$ line length

Either or both left and right margins may be used. If both are used, the further restriction that the left margin be less than the right margin will be enforced. When left margin is not specified, it defaults to 1; right margin defaults to infinity.

3.2.2 Right Margin Overflow Action

This occurs when the column about to be printed is the right margin. An action, such as hyphenating the line or printing a line number, may be taken at this point. The user specifies right margin action by supplying a function that will be executed when overflow occurs. The default for right margin overflow is line termination.

3.2.3 Page Size

Lines may be grouped by pages with a margin at the top and/or bottom. When used, these parameters are specified by the number of lines per page, and line numbers of the "top" and "bottom." When the bottom of a page is reached, an action--such as printing a page number or skipping to the next page and printing a header--can be effected by a bottom overflow function. If page format is not specified, the default will be to continuous use of every line. That is, top = 1, bottom = lines per buffer, and overflow is a "no-operation."

3.2.4 Code Form

Internal to LISP, all files are in ASCII form. When a file is connected to auxiliary storage where a different form (such as BCD) is used, this form must be specified. The default is to ASCII form.

3.2.5 Unit Connection

The types of storage devices available are tape, disc, teletype, display scope, and core. A unit is considered to be a block of storage on one of these devices (see Section 2). Connection is specified by unit name or by a complete declaration for the unit. Standard unit descriptions, which will be sufficient for normal usage, are available in the system. Default will be to core if no specification is given.

4. USER INPUT/OUTPUT FUNCTIONS

4.1 FUNCTIONS AFFECTING FILES

4.1.1 Functions Applicable to Any File

4.1.1.1 OPEN$LISP

This is a function of two arguments: file name and a description list for the file. The description list contains dotted pairs consisting of the file property name and the desired value for that property. Sections 3.1 and 3.2 describe the properties of a file; the file property names are given in Table 3.

When the description list has more than one occurrence of a property name, the first occurrence is used. This makes it possible to use standard file descriptors with any exceptions CONS'ed on the front.

Table 3. File Properties, Names, and Value Types

Property	Name	Value Type When Used
Line Length	LINE	Integer
Buffer Size	BUFFER	Integer
Left Margin	LINEL	Integer
Right Margin	LINER	Integer
Right Margin Overflow	LINEO	Function Name
Page Size	PAGE	Integer
Page Top	PAGET	Integer
Page Bottom	PAGEB	Integer
Page Overflow	PAGEO	Function Name
Code Form	FORM	Identifier (implementation-dependent)
Unit Connection	UNIT	Either the name of an active unit or a list suitable for the function OBTAIN (see OBTAIN, Section 4.2)

Use of the function OPEN causes one of the following:

- If the given file name is already in use as a file name, an error occurs.
- If any permanent property is absent, an error occurs.
- Otherwise, all properties are checked for legality; defaulting occurs if any changeable properties are not specified or a specified property is out of range (such as LINEL ≥ LINER).

After the legality check is completed, structures to represent the file are created,[6] If an active unit name is supplied, the unit will be positioned at the beginning of the region given in the device sub-specification; file position information will be initialized to coincide. If a unit description is supplied, the function OBTAIN will be executed.

The new file is added to the system inventory and a list of all active file names is returned.

4.1.1.2 INPUT$LISP and OUTPUT$LISP

These two functions are used to select a file (already OPEN'ed) for reading or printing, respectively. The file name to be selected is the sole argument; the function value is the name of the previously read- or print-selected file. By saving the name of the old file, it is possible to reselect it and continue at a later time.

If a file has never been selected after it was OPEN'ed, either function INPUT or OUTPUT can be used, since files are fundamentally direction-independent. However, certain restraints are imposed on the selecting of files under other conditions.

The status of every file is maintained by the LISP system. When a file is to be selected, its status is checked, and if the selection is illegal, an error occurs. The illegal combinations are: attempted OUTPUT when the file status is reading, attempted INPUT when the file status is either writing or has just written an end of file. Rewinding a file initializes its status, and any operation is then legal.

---

[6] A detailed description of these structures is found in the implementation amendment for each computer.

4.1.1.3 CHANGE$LISP

This function is used to change the value of any or all of the changeable properties of a previously OPEN'ed file. There are two arguments: file name and a description list. The description list has the same form and key words as used in OPEN. The new values are immediately placed in the file description and become effective as shown in Table 4.

Table 4. CHANGE Function Property Key Words

Property Key Word	New Value Becomes Effective
LINEL	After the current line is terminated
LINER	The first time the current column has the new value
LINEO	When the next right margin overflow occurs
PAGE	Immediately
PAGET	After the current page is finished
PAGEB	The first time line No. within page has this value
PAGEO	When the next bottom of page overflow occurs
FORM	At the next reading or writing of a unit
UNIT	At the next reading or writing of a unit

4.1.1.4 SHUT$LISP

The effect of this function is to purge a file from the LISP system. The file name to be shut is one of the arguments, and the other is a disposition specification for the file. The disposition list allows inserting or deleting this file in the monitor inventory, renaming the file in the monitor inventory, changing the protection of the file, and sending a message to the computer operator. The value of SHUT is a list of the remaining active file names in the LISP system.

SHUT has some implicit effects as well. If the file to be shut is in writing status, any remaining partial buffer will be written onto the unit connected to the file, and an end of file will be written. This occurs before the disposition specification is acted on. After the file is shut, the unit it was connected to may be released. This happens when no other file in the LISP system is connected to that unit.

File disposition is specified by a list of dotted pairs with default values assumed for those parameters not supplied.

Table 5. SHUT Function Disposition Parameters

Disposition Parameter	Name	Possible Values	Default Value
Monitor inventory insertion or deletion	INVENTORY	One of the identifiers INSERT or DELETE	DELETE
File name change	RENAME	An identifier	Current name of the file
File protection	PROTECT	Either or both of the identifiers READ and WRITE	No protection
Operator message	LOG	A single datum to be printed on the operator's console	No message

4.1.1.5 REWIND$LISP

The file specified as the argument to this function is initialized effectively as if it had just been OPEN'ed.

4.1.1.6 WRITEOF$LISP

The file specified as the argument of this function will be written with an end of file mark, providing the file status is not reading.

4.1.1.7 ADVLINE$LISP and TABCOL$LISP

Changing column and line indices of a file is accomplished by using these functions. Both take a file name and an integer for arguments and return integer values.

Line positioning in a forward direction is accomplished by calling ADVLINE with the file name and the number of lines to be skipped. The function value is the line number (within a page) before the skipping took place. If zero is supplied for the number of lines to be skipped, the present position is returned and no action takes place. Arguments less than zero cause an error.

Column positioning within the current line is achieved by calling TABCOL with the file name and the column number to be positioned to. If the desired column number is ≥ 1 and $\leq$ line length, the positioning will be done and the previous column number returned as the function value. When the desired column number is outside of these bounds, an error occurs.

4.1.2 Functions Applicable to the Currently Selected Input File

4.1.2.1 READCH$LISP

This is a function of no arguments that always returns a character as its value. The character returned is either from the currently selected input file or is a control character indicating end of line, end of file, or some other condition.

READCH may indirectly cause a unit to be read. This can happen if either the file buffer is empty or if right margin overflow causes the last line of the buffer to be terminated.

The effect of READCH on the state of a file is:

- Current column index is always changed.
- Current line index may change.
- File status may change. (This ordinarily occurs only at the beginning or end of a file, but may also result from an error in attempted unit reading.)

4.1.2.2 READ$LISP

This is a function of no arguments that returns either the next datum on the currently selected input file or a control character, or it causes an error. The syntax of LISP data is followed in parsing the input character stream; corresponding internal structures are created.

After READ has finished, the current character of the file is the one immediately after the last parsed character. The line is not terminated.

4.1.2.3 IADVL$LISP and ITABCOL$LISP

These functions operate on the currently selected input file and are analogous to the functions ADVLINE and TABCOL, except that file name is not supplied as an argument.

4.1.3 Functions Applicable to the Currently Selected Output File

4.1.3.1 PRINCH$LISP

This is a function of one argument, the character to be entered in the file; it also returns the argument as the function value. Before the character is inserted in the file, the file control indices are tested; right margin overflow or line termination may occur. If neither of these occur, the character is entered in the file and the current column index is incremented. If line termination occurs, either directly or through right margin action, the character is entered in the left margin position of the next line, providing no error occurs. An error can result if the terminated line was the last one in the buffer and an error occurred in attempting to write the unit.

The effect of PRINCH on the state of a file is exactly the same as READCH.

4.1.3.2 PRIN$LISP

This function has one argument, a LISP 2 datum, which it prints into the currently selected output file. Printing is done in a symmetric fashion; that is, the re-reading of the printed datum will cause an equal internal structure to be created. For example, the string #A B C# will print as #A B C#, and not as A B C.

PRIN does not cause a line termination; when it is through printing, the current column is the one immediately after the last character in the representation of the datum.

4.1.3.3 OADVL$LISP and OTABCOL$LISP

These are the currently selected output file counterparts of the functions described in Section 4.1.2.3.

4.2 FUNCTIONS AFFECTING UNITS

The functions OBTAIN and RELEASE are analogous to the OPEN and SHUT functions for files. Both are in section LISP and have an indefinite number of arguments.

OBTAIN requests from the monitor the units that are given as its argument. If they are all gotten successfully, OBTAIN then establishes the unit descriptions for them and returns a list of all active unit names. If one or more of the units can not be gotten from the monitor, OBTAIN does nothing and returns the empty list as its value.

RELEASE deletes all the unit descriptions and tells the monitor to disconnect the units (where appropriate). No LISP error is possible, and all the units supplied as arguments are then inaccessible in LISP.

The argument form for both functions is the same, and consists of a list whose first element is the unit name and whose remaining elements are dotted pairs of either unit descriptors (in the case of OBTAIN) or disposition parameters (in the case of RELEASE).

Unit descriptors contain as their first element a key word--DEVICE, PARITY, PROTECT, EOM, ORG, or TRANSFER--corresponding to the properties given in Table 2. Examples of legal unit descriptors are (DEVICE . TTY), (DEVICE . (TAPE7 1234)), and (PARITY . BINARY). Disposition parameters are the same as for SHUT.

The function ALTER has a single argument of the same form as the arguments for OBTAIN. The changeable properties of the unit named will be reset to the values given, and the new values will become effective at the next unit transfer.

The function POSITION has two arguments: the unit name and the action desired. The effect of this function is to physically position the unit as specified. The possible actions are: skip a record or physical file, backspace a record or physical file, write an end of file, and rewind.

The functions READU and WRITEU are used to transfer records between main memory and the unit. Each function has three arguments: the unit name, the number of words to transfer, and the memory location of the first word. The effect of executing either of these functions is to transfer either the number of words specified or a physical record, whichever is smaller. Parity will be according to the unit's current specification.

5. MONITOR CAPABILITIES REQUIRED FOR LISP 2 INPUT/OUTPUT

5.1 MINIMUM I/O DEVICE CONFIGURATION

The following capabilities are required for input/output by the LISP 2 system:

- A teletype, console typewriter, or other means of direct user interaction.
- A bulk, random-access, secondary storage with at least 100,000 computer words available for use by the LISP 2 system.
- Magnetic tape handling capability.

5.2 HARDWARE RESTRICTIONS

In addition to the I/O devices required, the following limitations on these devices are necessary:

- The interactive typewriter must accept the full 7-bit ASCII code. Other devices must have a "binary" mode so that both blocks of instructions from memory and ASCII blocks can be exchanged without code translation.
- Error detection features must provide the option of continuing operation after an error is encountered.
- I/O interrupts of the LISP system must not be permitted unless specifically requested by the system.

5.3 NON-PARALLEL I/O MODE WITH SEQUENTIAL FILE ORGANIZATION

When LISP handles all files internally as if they were purely sequential (the standard implementation), the system appears to the user as if it were device-independent. The actual means of record access and device positioning is, of course, device-dependent.

There are three main classes of devices, each with their own monitor requirements. These classes are sequential, random, and display.

5.3.1 Sequential Devices

This class includes tapes and typewriters, and requires the least monitor capability. The software has to provide automatic transfer of physical records, some transfer error-recovery procedure, unambiguous status reports (ok, end-of-file mark, sub-specified error, etc.), ability to position to arbitrary records and files for reading, ability to write multi-file and multi-reel tape output, and file protection capability. When the monitor system allows more than one user at a time, it must also provide automatic queuing of I/O requests so that the I/O devices appear to the LISP system to be available at all times.

5.3.2 Random Devices

"Random" here means that any record can initially be accessed in roughly the same time. This class includes drum and disc storage and requires all the monitor capabilities specified in Section 5.3.1, a "dictionary" linkage for all records in a file, and the ability to expand or contract the space assigned to a file.

The dictionary capability allows LISP to access a file by name, instead of physical location, and to use physically non-contiguous segments of a file as if the first record in one segment physically followed the last record of the logically previous segment.

5.3.3 Display Devices

The capabilities described in Section 5.3.1 as well as an automatic renewing of displays must be provided. When LISP has prepared a record that represents the display, it expects to give this to the monitor for transfer and maintenance once and for all.

Input from display devices must not interrupt program operation unless this mode has been requested by LISP.

5.4 PARALLEL I/O MODE OR NON-SEQUENTIAL FILE ORGANIZATION

More monitor capabilities will be necessary for parallel I/O and non-sequential file organization, but since these modes of operation are options within the LISP system, they will not be discussed here.