Table of Contents

1. KOTO RELEASE OVERVIEW

Introduction

The Koto release of Interlisp-D provides a wide range of added functionality, increased performance and improved reliability. Central among these is that Koto is the first release of Interlisp that supports the new Xerox 1185/1186 artificial intelligence work stations, including the new features of these work stations such as the expanded 19" display and the PC emulation option. Of course, like previous releases of Interlisp, Koto also supports the other current members of the 1100 series of machines, specifically the 1132 and various models of the 1108.

Notable new software features include a growing collection of library and system support for features of CommonLisp, including CommonLisp arrays, bignums, catch and throw, and a variety of special forms and functions; numerous improvements to the XNS and TCP/IP networking code; a revised CHAT interface which permits different communication options and a variety of different terminal handlers, including VT100 and TEK 4010 emulation; a complete rewrite of the RS232 code for improved speed and functionality; support for the Xerox extended character set, which permits the use of an enormous variety of different character sets and fonts in strings, documents and Lisp data structures; a range of user interface improvements to the text editor; significant extensions and generalizations to the device independent graphics interface; support for the low-cost Xerox 4045 laser printer; and, via the Busmaster, interface support for a wide variety of industry standard bus peripherals.

In addition, the Koto release is accompanied by the first complete revision of the Interlisp Reference Manual since 1983. This revision is a comprehensive and total rewrite which has packaged the manual into three conveniently sized and organized binders covering both the Interlisp language, its environment and its I/O. Collectively, these new features and the many minor bug fixes and improvements that have been made to the system constitute a major expansion of its functionality. The rest of this document reviews all of these changes in a comprehensive manner, and provides detailed information as to the way in which the system has changed.

Summary of Major Incompatibilities

The Koto release contains fixes for over 450 problem reports. Many of these were minor details which were never noticed by the average user. Some, however, were of major impact to one or more users and are summarized in this document.

In Koto, as with other Interlisp releases, we have made a strong effort to retain a high degree of backward compatibility with past releases. However, there are a number of areas in which we have felt it necessary to make small incompatible changes in existing interfaces to extend their functionality. Incompatible changes are clearly indicated for each area in the sections that follow. The principal instances of this are summarized below:

Keyboard

- New meaning for META key
- Several key assignments have changed

Window System

- New semantics for:

  DOATTACHEDWINDOWCOM,
  DOATTACHEDWINDOWCOM2, and
  DOMAINWINDOWCOMFN (internal subfunctions of ATTACHEDWINDOW package)

- Semantics of WINDOWCOMACTION = MAIN in ATTACHWINDOW changed slightly

Arithmetic

- New arbitrary precision arithmetic; negative arguments to RADIX are no longer allowed

Storage

- CDR of non-list causes error (setting of CAR/CDRERR)
- RPLSTRING and RPLCHARCODE not guaranteed to modify substrings

Communications

- New meaning for SPP.OPEN argument WHENCLOSED
- SPP.EOMP superseded by EOFP

Changes Affecting Intermezzo Release Users

The following forward incompatibilities should also be noted by those users wishing to switch freely between the Intermezzo and Koto releases:

- The Koto 1108 Local File System cannot be read by Intermezzo sysouts.
- Code compiled with the Koto compiler will not necessarily run in Intermezzo.
- The TEdit file format has changed, so that documents written in Koto TEdit cannot be read by Intermezzo TEdit, unless the "Old Format" Put command is used.

How to Use this Document

All changes are described in more detail in the sections that follow. Three major categories are defined. They are:

- Features
- Bug Fixes
- Library Packages

Within these categories, descriptions are organized roughly by system and then by importance. Most of the principal changes have already been summarized above. At the beginning of each section, you will find a list of system areas that have changed for the Koto release.

[This page intentionally left blank.]

2. FEATURES

The following areas are addressed in this section:

- Characters
- Keyboard
- TTYIN
- Input/Output
- Printing
- Graphics
- Window System
- Arithmetic
- Common Lisp Support
- Storage
- Interlisp-D Environment
- File Package
- Masterscope
- Record Package
- Code Editor
- 1108 Local File
- Communications
- Virtual Memory
- Miscellaneous

Characters

Interlisp-D now supports the full 16-bit NS character set.

Interlisp-D now supports the Xerox corporate character code standard, commonly referred to as the NS (Network Systems) encoding, described in the document *Character Code Standard* [Xerox System Integration Standards, XSIS 058404, April 1984]. Previous to the Koto release, Interlisp-D used the ASCII (American Standard Code for Information Interchange) encoding. While the extended-ASCII encoding provided for 8-bit (256 available) characters (primarily Latin alphabet and computer-specific symbols), the NS encoding supports 16-bit (65536 available) characters comprising many foreign alphabets and special symbols. For instance, Interlisp-D supports the display and printing of the following:

Le système d'information Xerox 11xx est remarquablement polyglotte.

Das Xerox 11xx Kommunikationssystem bietet merkwürdige multilinguale Nutzmöglichkeiten.

Xerox 11xx имеет замечательную много-язычную способность.

$M \models \Box[w] \Leftrightarrow \forall v \text{ with } Rwv: M \models [v]$

The benefit of having this large character set, in contrast to approaches that use a small set of character codes and a multiplicity of fonts (e.g., a Greek font, a math font), is that each semantically distinct character is represented by its own character code, completely independent of the character's appearance (font). Thus, the Greek character upper-case Beta is always character code 9794, independent of whether it appears in printed form in a serif style, sans-serif style, italic, etc., and it is unrelated to the Roman letter B (character code 66).

NS characters can be used in strings, litatom print names, symbolic files, or anywhere else that characters can be used. All of the standard string and print name functions (RPLSTRING, GNC, NCHARS, STRPOS, etc.) accept litatoms and strings containing NS characters. For example:

```
←(STRPOS "char" "this is an 8-bit character string")
18
←(STRPOS "char" "celui-ci comporte des charactères NS")
23
←(SETQ МЯГКИЙ-ЗНАК 'Ь)
Ь
```

Characters are organized into 256-member *character sets*, each of which generally consists of semantically related characters. For example, character set 38 is the Greek character set and contains the Greek alphabet and punctuation characters needed to print Greek text. A 16-bit character code thus consists of an 8-bit character set and an 8-bit character number within that set. The ASCII character set is contained in NS character set zero; thus, ASCII characters are still represented by the same 8-bit character codes as previously (i.e., 16-bit character codes whose high 8 bits are zero). Most strings and atoms still consist entirely of characters from character set zero and are represented just as space-efficiently in memory and on files as in earlier releases of Interlisp-D that used only ASCII characters.

In almost all cases, a program does not need to know that it is dealing with 16-bit characters rather than 8-bit characters—the higher level system functions all treat them transparently. The exception is in character-level input/output, where the important fact to be aware of is that *characters are not bytes*. The file pointer of a random-access file still counts bytes, and the function NCHARS still counts characters, but the two are no longer directly related. This is discussed in more detail below.

Character-level Input/Output

Incompatible Change: **BIN and BOUT are no longer appropriate for character input/output.**

A character is no longer generally representable in 8 bits. Therefore, characters can no longer, in general, be read or written with the functions BIN and BOUT, which read and write 8-bit quantities. The change is mostly transparent to user programs, especially if those programs use only the higher level functions, such as READ and PRINT. However, it is likely that user programs that manipulated a file character by character using BIN and BOUT should now use the following functions, which may produce or consume more than a single byte:

(READCCODE *STREAM*) [Function]

(PEEKCCODE *STREAM*) [Function]

(PRINTCCODE *CODE STREAM*) [Function]

These functions are documented in the new Interlisp-D Reference Manual. The functions BIN and BOUT are still appropriate for use when reading and writing strictly binary (rather than character) data.

Interlisp-D supports two ways of writing NS characters on files. One way is to write the full 16-bits (two bytes) every time a character is output. The other way, which is the system default, is to use "run-encoding," in which a run of characters in the same character set is written as a sequence of 8-bit character numbers within the character set, preceded by a "change character set" command. The byte 255 (illegal as either a character set number or a character number) followed by a character set number is used to signal a change to a given character set; the following bytes, up until the next change-character set sequence, are all interpreted as coming from the specified character set. Run-encoding can reduce the number of bytes required to encode a string of NS characters, as long as there are long sequences of characters from the same character set, which is usually the case.

Most characters in common use, including those in the ASCII character set, are in character set zero; a file containing only these characters is thus in exactly the same format as in previous releases, viz., one byte per character. However, this should not be relied on.

The fact that the file representation of a character may be more than a single byte has important consequences for any program that uses random access on text files whose characters are run-encoded. First, and most obviously, you cannot count the characters in a string being printed and use that number to derive the file pointer of where the string ends—you must use

GETFILEPTR. Second, programs that use SETFILEPTR need to be aware of possible character set changes. At any point when a file is being read or written, it has a "current character set," viz., the character set specified in the most recent "change character set" command written on the file. If the file pointer is changed with SETFILEPTR to a part of the file with a different character set, any characters read or written may have the wrong character set. Programs that use COPYBYTES to copy blocks of characters must ensure both that they are copying on character boundaries and copying to a place that is in the correct character set.

The current character set can be accessed with the following function:

(CHARSET *STREAM CHARACTERSET*) [Function]

Returns the current character set of the stream *STREAM*, or T if *STREAM* is not run-encoded. If *CHARACTERSET* is non-NIL, the current character set for *STREAM* is set. For output streams this causes bytes to be written to the stream if *CHARACTERSET* is different from the current character set; for input streams it merely changes the reader's belief about the current character set. If *CHARACTERSET* is T, run encoding for *STREAM* is disabled—henceforth each character printed to the stream is printed as exactly two bytes (the character set and the character number).

Programs that wish to count characters or avoid worrying about character set changes can thus disable run encoding for a particular stream and count each character as two bytes. There is, however, a cost in file space.

Extensions to CHARCODE

CHARCODE has been extended to allow specifying NS Characters.

CHARCODE has been extended to allow the specification of 16-bit NS characters in multiple character sets. It also uses two new variables, CHARACTERNAMES and CHARACTERSETNAMES, so characters and character sets can be specified symbolically. The new definition is the following:

(CHARCODE *CHAR*) [NLambda Function]

Returns the character code specified by *CHAR* (unevaluated). If *CHAR* is a one-character atom or string, the corresponding character code is simply returned. Thus, (CHARCODE A) is 65, (CHARCODE 0) is 48. If *CHAR* is a multi-character litatom or string, it specifies a character code as described below. If *CHAR* is NIL, CHARCODE simply returns NIL. Finally, if *CHAR* is a list structure, the value is a copy of *CHAR* with all the leaves replaced

by the corresponding character codes. For instance, (CHARCODE (A (B C))) = > (65 (66 67)).

If a character is specified by a multi-character litatom or string, CHARCODE interprets it as follows:

CR, SPACE, etc. The variable CHARACTERNAMES contains an association list mapping special litatoms to character codes. Among the characters defined this way are CR (13), LF (10), SPACE or SP (32), ESCAPE or ESC (27), BELL (7), BS (8), TAB (9), NULL (0), and DEL (127). Examples: (CHARCODE SPACE) returns 32, and (CHARCODE CR) returns 13.

CHARSET,CHARNUM, CHARSET-CHARNUM If the character specification is a litatom or string of the form CHARSET,CHARNUM or CHARSET-CHARNUM, the character code for the character number CHARNUM in the character set CHARSET is returned. CHARSET is either an octal number, or a litatom in the association list CHARACTERSETNAMES (which defines GREEK, CYRILLIC, etc.). CHARNUM is either an octal number, a single-character litatom, or a litatom from the association list CHARACTERNAMES. Examples: (CHARCODE 12,6), (CHARCODE 12,SPACE), (CHARCODE GREEK,A) and (CHARCODE ↑ GREEK,A)

Note that if CHARNUM is a single-digit number, it is interpreted as an octal character code, *not* as a character. Thus (CHARCODE GREEK,3) denotes the fourth character in the Greek character set, not the character "3" in that character set.

↑ CHARSPEC If the character specification is a litatom or string of one of the forms above, preceded by the character "↑", this indicates a "control character," derived from the normal character code by clearing the seventh bit (100Q) of the character code (normally set in alphabetic characters). Example: (CHARCODE ↑ A)

#CHARSPEC (8-bit character codes) If the character specification is a litatom or string of one of the forms above, preceded by the character "#", the eighth bit (200Q), normally zero for 7-bit ASCII characters, is set. This is the way to get character numbers greater than 127. ↑ and # can both be set at once. Examples: (CHARCODE #A), (CHARCODE # ↑ GREEK,A)

Note: In Intermezzo (and in some other operating systems), characters with the eighth bit set were considered "meta" characters. In the Koto release, however, "meta" means character set 1, and the meta key produces characters with the 400Q bit set, not 200Q.

Keyboard

New keyboard interpretations have been assigned.

With this release, we have assigned actions to the various function keys on the 1108 and 1186 keyboards, so that applications like TEdit can use them sensibly. In the process, we tried to minimize the differences among keyboards, and to remain consistent with a later implementation of a user interface for Lisp.

Four kinds of keyboards are supported:

Lisp keyboard on 1108s

Old office keyboard on 1108s

Lisp keyboard on 1186s

Office keyboard on 1186s

New 1108 and 1186 Lisp keyboards will be configured as indicated in Figures 1 and 2, respectively. The other two keyboards require that some functions be rearranged.

1108 office keyboard: The latch of the "Lock" key is to be defeated, as that key is now a lock-shift toggle key (i.e. hitting it once turns shift-lock on, hitting it again turns shift-lock off).

1186 office keyboard: The shift-lock key stays a shift-lock (it becomes the CONTROL key on the Lisp version of the keyboard.) The PROP'S key becomes the CONTROL key, and the CAPS-LOCK key becomes EDIT.

Most of the function keys have been assigned codes in the Xerox Character Code Standard's character set 2 range, 512-767. These codes are represented below as 2,ooo, where ooo are octal digits corresponding to the character's offset from 512. E.g., Character 524 = 2,14.

Generally, only a single character code is shown for each key. To get the shifted character code (if it's not specified another way), add 40 octal to the value shown. Sometimes, codes have only been assigned to the shifted state of a key, or only the unshifted state (e.g., on the 1186's numeric keypad, the digit keys are digits when shifted, and have special functions when unshifted); these are noted.

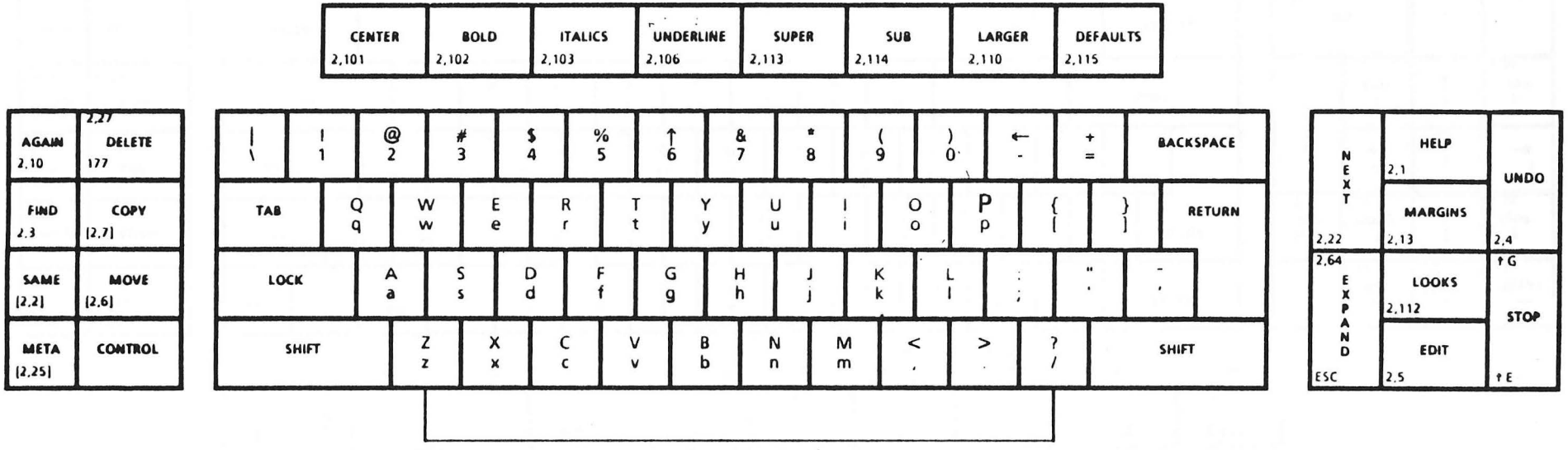

Figure 1 1108 Lisp Keyboard

Use this page to replace pages 11 and 12 of your Koto Release Notes.

Use this page to replace pages 11 and 12 of your Koto Release Notes.

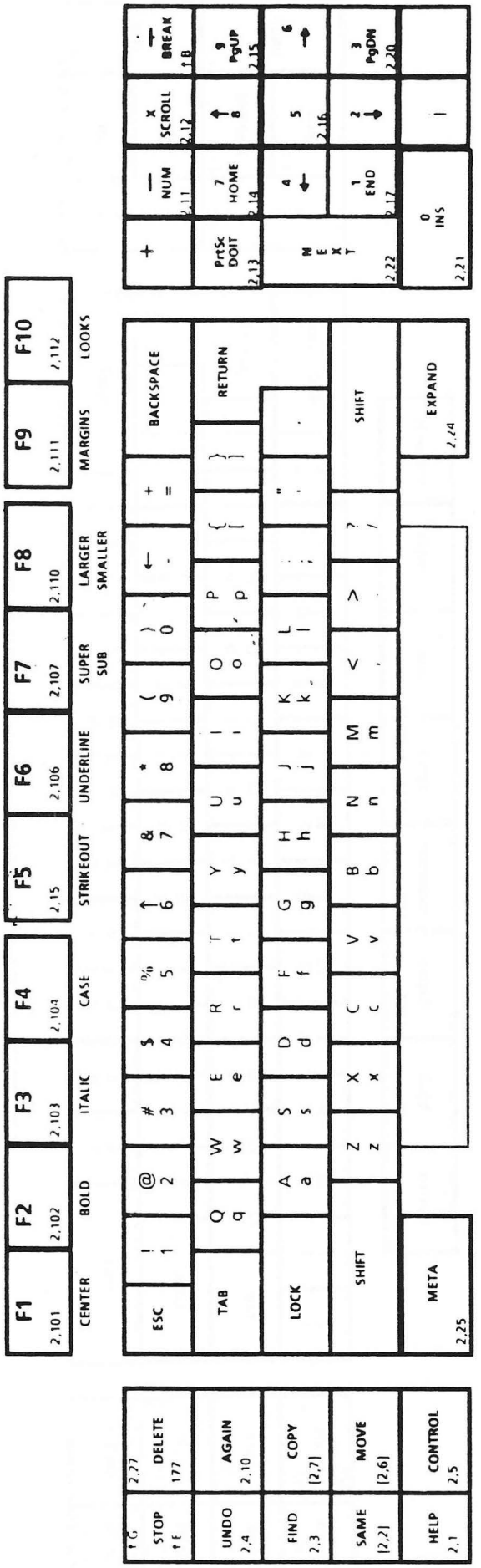

Figure 2 1186 Lisp Keyboard

Four keys were treated specially: META, SAME, COPY, and MOVE. The META key is a SHIFT-like key, with the effect of shifting the typed character into NS character set 1. The SAME key is equivalent to the META key for this release, to retain compatibility with the TEdit use of the META key for copying looks. The SAME, COPY and MOVE keys are left without key actions. The intent is that these keys be used as selection modifiers; in future releases, we plan to install shift states to correspond to them, so that the functionality can be remapped to other keys at the user's discretion. For those who wish to assign real character codes to these keys for special use, the assigned codes are:

META 2,25

SAME 2,2

MOVE 2,6

COPY 2,7

These codes are shown in brackets in the keyboard tables below to indicate that these keys initially have no code assignments.

Although we have attempted to minimize incompatibilities, users may wish to note the following effects of the new keyboard assignments:

- The BS key now generates ↑H rather than ↑A, making it ASCII-compatible.
- There is no longer a single key on the 1108 or 1185/6 that generates the ASCII character line feed. To type a line-feed, use CONTROL-J.
- There are no longer any keys on the 1108 keyboard that generate the same codes as the three blank keys on the 1100/1132 keyboard (these used to correspond to KEYBOARD, OPEN, AND STOP).
- Since the EXPAND key (on 1108s) now generates the ASCII Escape character, the default TEdit syntax for the character Escape, even on the the 1132, is now EXPAND. Users who prefer the previous semantics of ESCAPE should execute

  (TEDIT.SETSYNTAX (CHARCODE ESC) 'REDO).

- The META key generates characters in character set 1. In earlier releases, it generated characters with the 200Q bit set. Programs that tested META codes need to be changed.

1185/6 LISP Keyboard

Keys in the left-hand cluster:

Key legend	unshifted code	shifted code
HELP	2,01	
SAME	[2,02]	
FIND	2,03	
UNDO	2,04	
STOP	↑E	↑G
EDIT	2,05	
MOVE	[2,06]	
COPY	[2,07]	
AGAIN	2,10	
DEL	177	2,27

Keys in the right-hand cluster:

Key legend	unshifted code	shifted code
CAPS LOCK	[lock toggle]	[lock toggle]
NUM LOCK	2,11	-
SCROLL LOCK	2,12	×
BREAK	↑B	÷
DOIT	2,13	
7	2,14	7
8	↑	8
9	2,15	9
4	←	4
5	2,16	5
6	→	6
1	2,17	1
2	↓	2
3	3,20	3
0	2,21	0
NEXT	2,22	

Function Keys:

Key legend	unshifted code	shifted code
F1/Center	2,101	
F2/Bold	2,102	
F3/Italics	2,103	
F4/Case	2,104	
F5/Strikeout	2,105	
F6/Underline	2,106	
F7/Super-Sub	2,107	
F8/Larger-Smaller	2,110	
F9/Margins	2,111	
F10/Looks	2,112	

Keys in the main cluster:

Key legend	unshifted code	shifted code
Backspace	↑H	↑H
Return	CR	CR
EXPAND	2,24	2,64
META	[2,25]	

1108 Lisp Keyboard

Keys in the left-hand cluster:

Key legend	unshifted code	shifted code
SAME	[2,02]	
FIND	2,03	
MOVE	[2,06]	
COPY	[2,07]	
AGAIN	2,10	
DEL	177	2,27

Keys in the right-hand cluster:

Key legend	unshifted code	shifted code
HELP	2,01	
UNDO	2,04	
STOP	↑E	↑G
EDIT	2,05	
DOIT	2,13	
NEXT	2,22	
LOOKS	2,111	
EXPAND	ESC	2,64

(note that this means there are two EXPAND characters)

Function Keys:

Key legend	unshifted code	shifted code
F1/Center	2,101	
F2/Bold	2,102	
F3/Italics	2,103	
F4/Underline	2,106	
F5/Superscript	2,113	2,153
F6/Subscript	2,114	2,154
F7/Larger-Smaller	2,110	
F8/Defaults	2,115	2,155

Keys in the main cluster:

Key legend	unshifted code	shifted code
Backspace	↑H	↑H
Return	CR	CR

Setting Up Your Keyboard

If you have an older 1108 keyboard, or the office version of the 1185/6 keyboard, you should execute (or have your personal INIT file execute)

(SETUP.OFFICE.KEYBOARD)

Keyboard Programmer Changes

The internal handing of keyboard buffering has changed in Koto. For the most part, the changes were an internal restructuring which will allow future enhancements.

New function, SHIFTDOWNP, for sensing shift flags.

(SHIFTDOWNP *SHIFT*) [Function]

Returns T if the internal "shift" flag specified by SHIFT is on; NIL otherwise. If *SHIFT* = 1SHIFT, 2SHIFT, LOCK, META, or CTRL, SHIFTDOWNP returns the state of the left shift, right shift, shift lock, control, and meta flags, respectively. If *SHIFT* = SHIFT, SHIFTDOWNP returns T if either the left or right shift flag is on. If *SHIFT* = USERMODE1, USERMODE2, or USERMODE3, SHIFTDOWNP returns the state of one of three user-settable flags (see below) that have no other effect on key interpretation. These flags can be set or cleared on key transitions by using KEYACTION.

New KEYACTION types are accepted for changing the state of shift flags:

LOCKTOGGLE

complements the lock shift flag (turning it off if the flag is on; on if the flag is off) For keyboards where the lock-spring is removed, a reasonable configuration might be (KEYACTION 'LOCK '(LOCKTOGGLE . IGNORE)) so that the lock state can be changed just by hitting the key again instead of having 2 keys (superscript and subscript) tied to this one function.

USERMODE1UP, USERMODE1DOWN, USERMODE1TOGGLE

USERMODE2UP, USERMODE2DOWN, USERMODE2TOGGLE

USERMODE3UP, USERMODE3DOWN, USERMODE3TOGGLE

provide three user shift-modes. The transitions change an internal bit that the user can read using SHIFTDOWNP. These flags have no other effect on key interpretation.

TTYIN

TTYIN supports Copy/Move keys on 1108 or 1185/86.

TTYIN on an 1108 or 1185/86 now supports the COPY and MOVE keys as synonyms for SHIFT and CTRL-SHIFT when doing mouse selection.

TTYIN uses UNDO key to restore buffer or undo deletion.

This functionality is on the middle-blank key on the 1132, and used to be on the OPEN key on the 1108. When pressed as the first character of input, the UNDO key unreads the previous lines of input. When pressed following a deletion (e.g., after a series of backspaces), UNDO restores the deleted characters.

Input/Output

INTERRUPTCHAR specifies which process the interrupt occurs in.

The HARDFLG argument to INTERRUPTCHAR (previously ignored in Interlisp-D) is now used to determine which process the interrupt should run in. If HARDFLG is NIL, the interrupt will run in the TTY process, which is the process currently receiving keyboard input. If HARDFLG is T, the interrupt will occur in whichever process happens to be running. If HARDFLG is MOUSE, the interrupt will happen in the mouse process, if the mouse is busy, otherwise in the TTY process.

RESET.INTERRUPTS has been changed to permit setting HARDFLG.

The PERMITTEDINTERRUPTS argument to RESET.INTERRUPTS has been extended to accept a list of elements (CHAR TYP/FORM HARDFLG), instead of just (CHAR TYP/FORM).

SKREAD now takes a RDTBL argument.

SKREAD now takes a RDTBL argument like READ; unlike READ, the RDTBL defaults to FILERDTBL, not the current primary read table.

READFILE has new RDTBL and ENDTOKEN arguments.

(READFILE *FILE RDTBL ENDTOKEN*) [Function]

Reads successive expressions from *FILE* using READ (with read table *RDTBL*) until the single litatom *ENDTOKEN* is read, or an end of file encountered. Returns a list of these expressions.

If *RDTBL* is NIL, it defaults to FILERDTBL. If *ENDTOKEN* is NIL, it defaults to the litatom STOP.

EOF errors no longer automatically close the file.

This change actually took place in the Intermezzo release, but was not noted then. When an END OF FILE error occurs, the file now remains open. Most well-structured code will close the file

after you exit the error, but programs that depended on the file being closed by the system should be examined. To restore the old behavior, where the file is closed by the system before the END OF FILE error is signaled, (SETQ DEFAULTEOFCLOSE 'CLOSEF).

Mouse event functions have been removed.

The functions ENABLEMOUSE, GETMOUSEBUF etc. have been removed from the system. We believe these have not functioned for the past several releases.

New function SKIPSEPRS exists for skipping over "white space."

(SKIPSEPRS *FILE RDTBL*) [Function]

Advances the file pointer for *FILE* until it encounters a non-separator character (as defined by *RDTBL*). The file pointer is left positioned ready to read the character, which is also returned. If no non-separator character is found before the end of file is reached, SKIPSEPRS returns NIL and leaves the pointer positioned at the end of the file. Useful for skipping over "white space" when scanning a file, character by character.

DIRECTORY features "$", "?", "-" documentation has been clarified.

The DIRECTORY function pattern-matching characters "$" (escape) and "?" do not work for all file servers. In addition, compound file specifications such as (T* - *.DCOM) have never been totally debugged. These features never functioned properly, and were poorly designed, so we are withdrawing support for them, and removing them from the documentation. There are currently no plans to re-implement them.

DIR and NDIR are now functions, not macros.

This means they can be called from any place you can evaluate a Lisp expression (e.g., DEdit). However, it also means that DIR typed as a single atom to the executive does not work. You must type "DIR]" or "DIR *" instead of just "DIR" in order to enumerate all files on the connected directory.

DELVER option has been added in DIRECTORY.

DIR <spec> DELVER [LispX Command]

This will delete all but the highest version of all files matching <spec>.

Subdirectories are better supported.

The file system now understands partial names that contain a subdirectory specification without the main directory. In the case of a name being treated as a a file name, the name is considered to contain a subdirectory if it does not start with any directory punctuation, but there are terminating directory delimiters in it. In the case of a name being treated as a directory, the name is considered to be be a subdirectory if it does not start with any directory punctuation, but *ends* in a directory delimiter. CONN understands subdirectories, as does the code that fills in connected directory defaults.

For instance, if you are connected to {Server}<Foo>, then

"Bar>Fum.tmp" refers to {Server}<Foo>Bar>Fum.tmp.

"<Bar>Fum.tmp" refers to {Server}<Foo>Bar>.

"CONN Bar" or "CONN<Bar>" connects you to {Server}<Bar>.

Vertical-bar syntax has been extended and now allows entering numbers in arbitrary radices.

The read macro for vertical bar (|) has been extended to allow a number of new forms to be entered, including entering integers in arbitrary radices.

When followed by an end-of-line, tab or space, | is ignored, i.e., treated as a separator character. Otherwise it is a "dispatching" read macro whose meaning depends on the character(s) following it. The following are currently defined:

' (quote) – synonymous with back-quote.

. (period) – returns the evaluation of the next expression, i.e., this is a synonym for control-Y.

, (comma) – returns the evaluation of the next expression at load time, i.e., the following expression is quoted in such a manner that the compiler treats it as a literal whose value is not determined until the compiled function containing the expression is loaded.

"O" or "o" (the letter O) – treats the next number as octal, i.e., reads it in radix 8. For example, |o12 = 10 (decimal).

"B" or "b" – treats the next number as binary, i.e., reads it in radix 2. For example, |b101 = 5 (decimal).

"X" or "x" – treats the next number as hexadecimal, i.e., reads it in radix 16. The upper-case letters A though F are used as the digits after 9. For example, |x1A = 26 (decimal).

"R" or "r" – reads the next number in the radix specified by the (decimal) number that appears between the | and the R.

When inputting a number in a radix above ten, the upper-case letters A through Z can be used as the digits after 9 (but there is no digit above Z, so it is not possible to type all base-99 digits). For example, |3r120 reads 120 in radix 3, returning 15.

(, {, ↑ – used internally by HPRINT and HREAD to print and read unusual expressions.

Backquote read macro now transforms "," and ",@" prefixes.

The Backquote read macro transforms the "," and ",@" prefixes in the body of the form to be the wrappers "\," and "\,@". for example, '(A ,B ,@C) reads in as (BQUOTE (A (\, B) (\,@ C))). This change allows certain forms to be read and translated correctly that previously could not. However, the older forms of BQUOTE, in which the comma prefixes appeared at top level, are still recognized.

Printing

Landscape printing is now supported for Interpress.

Landscape printing is now supported on Interpress printers such as the 8044. To print a landscape document, include the option LANDSCAPE, value T, in the OPTIONS argument to OPENIMAGESTREAM or SEND.FILE.TO.PRINTER.

Note: When sending TEdit hardcopy to a landscape image stream, using TEDIT.FORMAT.HARDCOPY, it is still necessary to set the margins in the page layout menus relative to a portrait page.

Printing to Fax Print server is now supported.

Hardcopy output can now be sent to a telecopier if you have an NS server with Fax (telecopier) service enabled. Any place that Lisp needs a "printer name", such as in the "Hardcopy Server" field of the TEdit menu, you can supply the name in the form of *Person@Place*. The *Person* part of the name is optional; if present it is the name of the recipient of the telecopy (if omitted, the @ is still required.) *Place* somehow specifies how to get there: the name of the NS server to which the Fax request is to be sent, and the phone number the server must dial to get to the desired remote telecopier. Place takes one of two forms:

1. A phone number (sequence of digits and hyphens), exactly as the Fax server must dial it from its dial-out phone, or

2. A place name, a single mnemonic of your choosing. You register place names by adding them to the a-list

FAXADDRESSES. Each element in FAXADDRESSES is of the form (PlaceName PhoneNumber FaxServerName), where the first element is an uppercase litatom, the second is a string containing the phone number (as in (1)), and the optional third is the name of the desired NS Server.

The variable DEFAULTFAXHOST contains the name of the NS Fax Server to use if not otherwise specified, such as in case (1), or if the FaxServerName in the entry in FAXADDRESSES is missing in case (2).

Variable INTERPRESSFAMILYALIASES allows coercing font family names.

There is a new global variable to control the rendition of fonts on NS printers: INTERPRESSFAMILYALIASES. It is a property list of the nominal font name matched with the printer's name for a given font. For example, the initial value of this variable is

(LOGO Logotypes-Xerox)

PRINTERTYPE property can be used to determine printer type.

The function PRINTERTYPE uses a number of heuristics to determine the printer type of a printer. Another useful way of specifying the type of a printer is to put the type as the value of the PRINTERTYPE property of the printer name.

Graphics

Polygon filling is now provided by Device Independent Graphics (DIG) for the Display and Interpress Printers.

For details see the new revised InterLisp-D Reference Manual. Because many Interpress printers do not implement the polygon mask instruction, polygons are currently scan-converted in Lisp. In spite of this, large or complex polygons may exceed the capabilities of some printers. Scan-converted large or complex polygons also create extremely large Interpress masters, and thus may take a very long time to output. Future releases will provide a mechanism to better address various printer capabilities and restrictions.

Window System

New heuristic is used to reshape windows.

The new heuristic prefers to leave the upper left corner of the window visible. Under some conditions, the part of a window that is visible after it is reshaped to be smaller will be different than in previous releases.

GETREGION and SHAPEW have been changed to allow specification of initial "ghost" region.

Previously, the initial "ghost" region that GETREGION used to prompt the user was always MINWIDTH by MINHEIGHT. GETREGION has been changed to take a new argument, INITCORNERS, that specifies the initial ghost region. If INITCORNERS is non-NIL, it should be a list of the form (BASEX BASEY OPPX OPPY), where (BASEX, BASEY) describes the anchored corner of the box, and (OPPX, OPPY) describes the trackable corner (in screen coordinates). The cursor is moved to (OPPX, OPPY). If INITCORNERS is NIL, the initial ghost region is determined by MINWIDTH by MINHEIGHT, as before.

SHAPEW has been changed to use the INITCORNERS argument when it calls GETREGION. If the window property INITCORNERSFN is non-NIL, SHAPEW applies it to the window, and the result is passed as the INITCORNERS argument to GETREGION. This allows windows to specify their default size when reshaped.

Attached windows can except themselves from individual operations.

The attached window facility has been reworked to provide a cleaner way of specifying whether window operations apply to the whole window group, or just the individual window. Instead of directly changing the DOWINDOWCOMFN of the main and attached windows, each attached window has two new window properties lists, PASSTOMAINCOMS and REJECTMAINCOMS.

PASSTOMAINCOMS is a list of window commands (e.g. CLOSEW, MOVEW) which when selected from the attached window's right-button menu are actually applied to the central window in the group. If NIL, all window operations are directly applied to the attached window. If T, all window operations are passed to the central window.

REJECTMAINCOMS is a list of window commands that the attached window will not allow the main window to apply to it. This is how a window can say "leave me out of this group operation." If NIL, all window commands may be applied to this attached window. If T, no window operations are accepted.

These changes have been made to preserve backward compatibility as much as possible. ATTACHEDWINDOW still interprets its WINDOWCOMACTION argument the same way,

but implements it by setting the PASSTOMAINCOMS window property appropriately.

For more complete documentation on the attached window facility, see the Interlisp-D Reference Manual.

Incompatible change: **Calling ATTACHWINDOW with a WINDOWCOMACTION argument of MAIN will now cause all window operations selected from the attached window's right button menu to be applied to the central window of the group.**

In past releases MAIN caused the main window's RIGHTBUTTONFN to be called.

Incompatible Change: **New semantics exist for the functions DOATTACHEDWINDOWS, DOATTACHEDWINDOWCOM2, and DOMAINWINDOWCOMFN.**

Because of the changes described above, the meanings of these functions have changed. DOATTACHEDWINDOWCOM is exactly the same as DOATTACHEDWINDOWCOM2; both examine the PASSTOMAINCOMS properties of the window. Programs that call these functions directly should be examined to be sure the new semantics are consistent with the program's intent.

New attached window function has been added: REMOVEWINDOW.

The function (REMOVEWINDOW *WINDOW*) detaches and closes *WINDOW*, which is assumed to be a window attached to another. In addition, any other windows attached above or below the same window, but farther away from the main window, will be snuggled up closer to the main window to fill the gap. (Gaps on the left and right, however, are not yet handled.)

Extended scrolling capability has been implemented for EXTENT.

A new window property was added to control the way scrolling deals with the window's EXTENT. Previously, scrolling was limited to keep the X dimension of the EXTENT in the window and to keep the Y dimension visible in the window or just off the top of the window. This had two problems: the user was forced to choose between thumb scrolling and the ability to scroll into space that is outside of its EXTENT, and some applications didn't want Y to scroll off the top.

To correct these shortcomings, the window property SCROLLEXTENTUSE is used to indicate how the EXTENT field limits scrolling. The possible values for it are:

NIL — The current behavior (for backward compatibitity) which is limit in X, allow Y off top. In this mode the EXTENT is either in the window or just of the top of the window.

T — Doesn't limit in either dimension but does support thumb scrolling. The user can scroll the window to anywhere but can get back to the EXTENT by thumb scrolling.

LIMIT — Limits in both directions. The window is only allowed to view things within the EXTENT.

+ — Allow the EXTENT to scroll just off in the positive direction.

- — Allow the EXTENT to scroll just off in the positive direction.

+ - or - + — Allow the EXTENT to scroll just off in both positive and negative directions.

CONS of the X behavior and the Y behavior:

Same atoms as above except that NIL is equivalent to LIMIT.

Note: The default is equivalent to (LIMIT . +).

Inspector windows allow copy selection.

It is now possible to copy-select items from an Inspector window. For printable objects (atoms, strings, numbers), the print name is copy-selected: for unprintable objects, an expression of the form (\VAG2 x y), which when evaluated yields the object, is copy-selected.

Arithmetic

Arbitrary-size integers (bignums) are now supported.

Interlisp-D now supports arbitrary-size integers, or "bignums." Whenever an arithmetic operation tries creating an integer larger than MAX.FIXP, a bignum is automatically created. The changes to the system are primarily invisible to the user: integer functions manipulate bignums the same as other integers. NUMBERP and FIXP of a bignum are true. Bignums can be typed in and printed by the system, just like other integers. Some related changes made to support bignums are:

1. With the addition of bignums, the concept of MAX.INTEGER is meaningless. However, for some algorithms, it is useful to have an integer that is larger than any other integer. Therefore, the values of MAX.INTEGER and MIN.INTEGER are two special bignums; the value of MAX.INTEGER is GREATERP than any other integer, and the value of MIN.INTEGER is LESSP than any other integer. Trying to do

arithmetic using these special bignums, other than for comparison, will cause an error. The variables MAX.FIXP and MIN.FIXP can be used, if necessary, to refer to the largest and smallest integers representable without using bignums.

2. Since integer overflow automatically creates bignums, the function OVERFLOW now only affects floating point overflow, and division by zero (either integer or floating point).

3. With this implementation, there are very few operations that are sensitive to the "word size" of the machine; for the most part, the word size is virtually infinite.

However, for backward compatibility, the functions LLSH and LRSH are defined to be modulo 2↑32. For the most part, user calls to LLSH and LRSH should instead be converted to LSH and RSH. LLSH and LRSH cause errors if passed bignums. This may cause some incompatibilities in current user code.

Incompatible change: **Negative arguments to RADIX are no longer allowed.**

Negative radices, which caused the number printer to interpret negative integers as unsigned numbers in the machine's "word size,"no longer make sense with bignums, and have been removed.

(OVERFLOW T) is the default setting.

The default setting for OVERFLOW has been changed to T, meaning that divide by zero and floating overflow will cause an error.

Common Lisp Support

This release contains a number of changes and enhancements which support the use of the Common Lisp dialect. Some parts of Common Lisp have been added into the basic Interlisp-D system, although most of the new functions exist in the library package CML. Included in the Lisp sysout are the integration of Common Lisp style DEFMACRO with the Interlisp-D environment, and a few minor functions such as PSETQ, LET, LET*, etc..

The CML library now supports Common Lisp arrays and array functions (as an upward-compatible extension of Interlisp-D's arrays, strings, and bitmaps), characters, full Common Lisp argument passing syntax (including &OPTIONAL, &REST, &KEY, etc.), and many others. Please see the CML package's documentation for full details.

Storage

Incompatible Change: **In this release, attempting to take CDR of a non-list causes an error.**

This behavior is controlled by the global variable CAR/CDRERR. It is currently set to CDR. If you want CDR to behave as before and return "{CDR of non-list}", (SETQ CAR/CDRERR NIL).

Warning now appears before disabling of garbage collection.

If the garbage collector's reference count table fills up, thereby disabling garbage collection, a warning pops up, advising you to save and reload as soon as possible (because without garbage collection your storage is going to fill up more quickly and operations will get steadily slower). The warning is similar to the one you get when your virtual memory backing file fills up.

Garbage collection algorithm has been modified.

The garbage collector now reclaims in one sweep entire structures whose reference count goes to zero, even if the structure is "bushy". This is in contrast to the previous behavior, where only one "thread" through a structure was collected on the first pass. This means you don't have to do (RPTQ 100 (RECLAIM)) in order to ensure that everything was collected, and has good consequences for working set and not filling up the reference count table. On the other hand, it means that if a large structure gets dropped on the floor, your next pause for garbage collection may be noticeably longer than it used to be. Exception: pointer arrays are not chased—they still require a second collection pass before their contents are reclaimed as well.

HASHARRAY has new args HASHBITSFN and EQUIVFN for user-settable hash functions.

HASHARRAY has two additional arguments, HASHBITSFN and EQUIVFN to support the creation of hash arrays that use an equivalence notion other than EQ. EQUIVFN is a function of two arguments that should return non-NIL when its arguments are to be considered equal. HASHBITSFN must be a function of one argument that produces a positive integer in the range [0..65535] with the property that objects that are considered equal produce the same integer.

The default for HASHARRAY is the same as it always has been; in other words, the EQUIVFN is EQ, with a HASHBITSFN that extracts bits from the pointer.

For assistance in building hash tables that take strings as hash keys, there is a function STRINGHASHBITS that is a suitable HASHBITSFN when EQUIVFN = STREQUAL.

For more information, see the new Interlisp-D Reference Manual.

Incompatible change: **RPLSTRING and RPLCHARCODE are not guaranteed to modify substrings.**

There is a caution in the October 1983 Interlisp Reference Manual that the value of a call to SUBSTRING shares storage with the original string, and hence that destructively replacing the characters of a string (using RPLSTRING or RPLCHARCODE) has the side effect of smashing any substrings or superstrings of the string as well. In Koto, the implementation of strings has changed such that this side effect, while still possible, is no longer guaranteed in all cases. Therefore, programmers should not rely on RPLSTRING or RPLCHARCODE altering the characters of any string other than the one directly passed as arguments to those functions.

ALLOCSTRING has new argument to accomodate NS characters

ALLOCSTRING has a new optional argument, *FATFLG*. When *FATFLG* is true, the string is allocated at full 16-bit character width, so that "fat" (16-bit) characters can be stored into it efficiently (using RPLSTRING or RPLCHARCODE). This flag has no other effect on the operation of the string functions, and is completely optional—RPLSTRING or RPLCHARCODE can still be used to store fat characters into a "thin" string, albeit less efficiently.

A new function, STRING-EQUAL, has been added.

The function (STRING-EQUAL X Y) compares two strings or atoms case-insensitively; that is, it returns true if they are equal when all lowercase letters are taken to be their uppercase counterparts.

Interlisp-D Environment

Idle Mode has been added to basic Interlisp-D environment.

The Interlisp-D environment runs on single-user computers. Often, users leave their computers up and running for days, which can cause several problems. First, the phosphor in the video display screen can be permanently marked if the same pattern is displayed for a long time (weeks). Second, if the user

goes away, leaving an Interlisp-D system running, another person could attempt to use the environment, taking advantage of any passwords that had been entered. To solve these problems, the Interlisp-D environment implements the concept of "idle mode."

If no keyboard or mouse action has occurred for a specified time, the Interlisp-D environment automatically enters idle mode. While idle mode is on, the display screen is blacked out, to protect the phosphor. Idle mode also runs a program to display a moving pattern on the black screen, so the machine doesn't appear broken. Usually, idle mode can be exited by pressing any key on the keyboard or mouse. However, the user can optionally specify that idle mode should erase the current password cache when it is entered, and require the next user to supply a password to exit idle mode.

Idle mode can also be entered by calling the function IDLE, or by selecting the Idle menu command from the background menu. The Idle menu command has options that allow the user to interactively set the idle options (display program, erase password, etc.) specified by the variable IDLE.PROFILE:

IDLE.PROFILE [Variable]

The value of this variable is a property list which controls many aspects of idle mode. The most useful properties are:

TIMEOUT – value is a number that determines how long (in minutes) Interlisp-D will wait before automatically entering idle mode. If NIL, idle mode will never be entered automatically. Default is 10 minutes.

ALLOWED.LOGINS – determines who can exit idle mode. If the value is NIL, this means that exiting idle doesn't check or require any login. Otherwise, the value of ALLOWED.LOGINS is a list, specifying which users are allowed to exit idle mode. This list can include * (require login, but let anyone exit idle mode), T (let the previous user exit idle mode), user names (let the specified users exit idle mode), and group names (allow any members of this group to exit idle mode). The default value of ALLOWED.LOGINS is (*), which means that anyone is allowed to exit idle mode, but an authorized login is required.

FORGET – if FIRST, the user's password is erased when idle mode is entered; if T, the user's password is erased when idle mode is exited (to allow incomplete file operations to terminate during the idle.) This value is irrelevant if ALLOWED.LOGINS is non-NIL, since the new login will overwrite the previous user's password. Default is NIL.

DISPLAYFN – the value of this property, which should be a function name or lambda expression, is called to display a moving pattern on the screen while in idle mode. This

function is called with one argument, a window covering the whole screen.

For more information on idle mode, see the new Interlisp-D Reference Manual.

Top-level functions accept quoted or unquoted arguments; new function NLAMBDA.ARGS has been added.

Interlisp-D has a number of top level functions which take 'unquoted' arguments; for example, EDITF and BREAK. New users were frequently confused by these functions, and often would supply a QUOTE where one was not needed, e.g., as in (EDITF 'FOO). To compensate for this frequent error, the functions EDITF, EDITV, FILESLOAD, TRACE, BREAK, UNBREAK, CLEANUP, BREAKDOWN, SEE, SEE* will accept either unquoted arguments, or arguments of the form (QUOTE <arg>). Thus, (EDITF 'FOO) is equivalent to (EDITF FOO), and (BREAK 'A) will break A, just as (BREAK A) does.

This new capability was implemented using the following new function, which is available for general use:

(NLAMBDA.ARGS X) [Function]

This function interprets its argument as a list of unevaluated Nlambda arguments. If the elements in this list are of the form (QUOTE ...), the enclosing QUOTE form is stripped off. NLAMBDA.ARGS stops processing the list after the first non-quoted argument, on the assumption that the quoting was not accidental. Therefore, whereas

(NLAMBDA.ARGS '((QUOTE FOO) BAR)) -> (FOO BAR),
(NLAMBDA.ARGS '(FOO (QUOTE BAR))) -> (FOO (QUOTE BAR)).

INSPECTCODE uses TEdit-type windows.

When TEdit is loaded, INSPECTCODE uses it to create a window that is scrolled more efficiently, and from which you can copy-select text. Also, the "InspectCode" menu item in the break backtrace frame now calls INSPECTCODE on the actual code in the frame (which makes a difference when the frame name is different from the function name), and highlights the location in the code where the frame's PC indicates it was executing at the time.

A warning has been added against modifying critical system functions.

Many users got into various kinds of trouble when they accidentally redefined or broke system functions. In order to prevent accidental modification of internal system functions, the following feature was added.

UNSAFE.TO.MODIFY.FNS [Variable]

Value is a list of functions that should not be redefined, because doing so may cause unusual bugs (or crash the system). If the user tries to modify a function on this list (using DEFINEQ, TRACE, etc), the system will print "Warning: XXX may be unsafe to modify -- continue?" If the user types "Yes", the function is modified, otherwise an error occurs. This provides a measure of safety for novices who may accidentally redefine important system functions. Users can add their own functions to this list.

Note: By convention, all functions starting with the backslash character (\ ) are system-internal functions, which should never be redefined or modified by the user. Backslash functions are not on UNSAFE.TO.MODIFY.FNS, so trying to redefine them will not cause a warning.

File Package

New variable MARKASCHANGEDFNS provides multiple definitions of MARKASCHANGED.

The variable MARKASCHANGEDFNS is a list of functions that MARKASCHANGED calls (with arguments NAME, TYPE, and REASON). Functions can be added to this list to "advise" MARKASCHANGED to do additional work for all types of objects. The WHENCHANGED file package type property can be used to specify additional actions when MARKASCHANGED gets called on specific types of objects. Masterscope and DEdit use the new mechanism rather than the old "built-in" way.

LOAD? is more careful about reloading files.

The function LOAD? has been changed to be more careful when deciding whether to load a file. (LOAD? FILE) loads FILE except when the same version of the file has been loaded (either from the same place, or from a copy of it from a different place). This means that if an old version of FILE is already loaded, but a new version now exists, LOAD? will load a different version of FILE. Specifically, LOAD? considers that FILE has already been loaded if the full name of FILE is on the variable LOADEDFILELST or the date stored on the FILEDATES property of the root file name of FILE is the same as the FILECREATED expression on FILE.

Note that this is different from the behavior of the FILES File Package command (and the function FILESLOAD), which does

not load a file if *any* version of that file is already loaded and the file was created using the file package.

Function ADDFILE can be used for explicitly recognizing files.

The user can explicitly tell the file package to notice a newly-created file by defining the filecoms for the file, and calling ADDFILE:

(ADDFILE *FILE* — — —)

Tells the file package that *FILE* should be recognized as a file; it adds *FILE* to FILELST, and also sets up its FILE property to reflect the current set of changes which are "registered against" *FILE*.

New file package type properties have been added: NULLDEF, CANFILEDEF.

The function FILEPKGTYPE recognizes two new file package type properties, NULLDEF and CANFILEDEF.

NULLDEF: The value of this property is returned by GETDEF when there is no definition and the NOERROR option is supplied. For example, the NULLDEF of VARS is NOBIND.

CANFILEDEF: If the value of this property is non-NIL, this indicates that definitions of this file package type are not loaded when a file is loaded with LOADFROM. The default is NIL. Initially, only FNS has this property set to non-NIL.

MAKEFILE allows continuing without loading (DECLARE: -- DONTCOPY --) expressions.

When a remake is specified, MAKEFILE checks to see how the file was originally loaded. If the file was originally loaded as a compiled file, MAKEFILE will call LOADVARS to obtain those DECLARE: expressions that are contained on the symbolic file, but not the compiled file, and hence have not been loaded. If the file was loaded by LOADFNS (but not LOADFROM), then LOADVARS is called to obtain any non-DEFINEQ expressions. Before calling LOADVARS to re-load definitions, MAKEFILE asks the user, e.g. "Only the compiled version of FOO was loaded, do you want to LOADVARS the (DECLARE: .. DONTCOPY ..) expressions from {DSK}<MYDIR>FOO.;3?". Respond Yes to execute the LOADVARS and continue the MAKEFILE, No to proceed with the MAKEFILE without performing the LOADVARS, or Abort to abort the MAKEFILE. You may wish to skip the LOADVARS if you circumvented the file package in some way, and loading the old definitions would overwrite new ones.

File package handling of function definitions has been regularized.

The internal structure of the file package and the way that it deals with function definitions has been unified. Masterscope, the editor, and other parts of the system all go through and use the standard GETDEF interface. Most of the changes are invisible to users.

EDITF, EDITV have changed.

The internal functioning of the editor entries EDITF, EDITV have been regularized; the result, while backward compatible, looks slightly different.

EDITF tries the following algorithm when asked to edit a function: it defaults to LASTWORD if no function is supplied. If a function is supplied, EDITDEF is invoked, with TYPE = FNS. EDITDEF then calls the FNS edit-method, which has similar behavior as before, with the exception that, if no definition is found, it calls EDITFERROR.

If you say EDITF(FOO) and FOO has no function definition but *does* have a macro, the system will put you in an editor for the macro instead.

New general-purpose function for calling editor has been added: EDIT.

(EDIT *NAME* —) [Function]

This function figures out what type of definition *NAME* has (function, variable, macro, etc.), and calls the editor to edit it. If *NAME* has more than one definition of different types, the user is prompted for the type of definition to edit. EDIT attempts to capture the previous behavior of the inspector and DEdit when asked to edit a given name.

New GETDEF option has been added: EDIT.

If the OPTIONS argument to GETDEF is (or contains) EDIT, GETDEF returns a copy of the definition unless it is possible to edit the definition "in place." With some file package types, such as functions, it is meaningful (and efficient) to edit the definition by destructively modifying the list structure, without calling PUTDEF. However, some file package types (like records) need to be "installed" with PUTDEF after they are edited. The default EDITDEF calls GETDEF with OPTIONS of (EDIT NOCOPY), so it doesn't use a copy unless it has to, and only calls PUTDEF if the result of editing is not EQUAL to the old definition.

Masterscope

Masterscope no longer automatically analyzes functions not explicitly mentioned.

Previously, if you typed ". ANALYZE FOO", and FOO called FUM, and FUM had an expr definition, Masterscope would analyze FUM too. This is no longer the case. It remains true that if you reference a function in a question such as ". WHO DOES FOO CALL," you imply that the function FOO should be analyzed.

Record Package

CLisp characters are allowed in record field names.

The record package name parser has been changed to allow the declaration of records with CLISP characters in field names, including "-". Such field names can only be used with fetch and replace, not in the "infix" notation, e.g., if you say (RECORD A (B-C D)) then (fetch B-C of X) will work, but X:B-C will not.

Code Editor

Edit macro IFY translates COND statements to IF statements.

If the current expression is a COND statement, the IFY edit macro replaces it with an eqivalent IF statement. This macro can be easily used by typing CONTROL-Z to DEdit.

1108 Local File

1108's local file system is faster, more robust.

The local file system has been revised so that it reads or writes multiple disk pages in a single operation, which makes disk accesses much faster. Also, the file system now does more error checking (at no loss in speed), so that errors are caught earlier, before information on the disk can be damaged.

Note: If a HARD DISK ERROR occurs, do not simply continue from the break. The problem will not be automatically fixed, although you may be able to run successfully for a while.

If such an error occurs, LOGOUT of Interlisp, run the Pilot scavenger on the logical volume in question (to straighten out any Pilot structures), restart Interlisp, call SCAVENGEDSKDIRECTORY (to straighten out the Lisp directory), and then continue.

You can now set initial position of DSKDISPLAY window.

The global variable DSKDISPLAY.POSITION contains a POSITION record which designates the screen position of the volume display window. If the DSKDISPLAY window is opened, it will appear at this position. If it is moved, this variable is reset to the new position.

DSKDISPLAY window now displays default volume.

The DSKDISPLAY window now includes a line displaying the default volume (the next volume with a Lisp files directory after the one you are running in). This is the volume that will be accessed by using the device {DSK} without any volume name.

(DISKFREEPAGES) uses default lisp files volume.

If given no argument, DISKFREEPAGES assumes the default volume (the next volume with a Lisp directory after the one you're running in.) If given an argument, it tells the number of free pages on the volume whose name matches the argument.

Communications

Incompatible change: **SPP.OPEN argument WHENCLOSEDFN no longer exists.**

The last argument to SPP.OPEN has been changed from WHENCLOSEDFN to PROPS, an optional property list, used to set the properties that determine the behavior of the SPP stream when certain events occur. The following properties can be specified:

CLOSEFN – a function or list of functions called (with the stream as argument) when an SPP connection is closed.

ATTENTIONFN – a function called (with the stream as argument) when an ATTENTION packet is received on the SPP connection.

ERRORHANDLER – a function called (with the stream as argument) when an error (such as end-of-stream) occurs on the SPP connection.

OTHERXIPHANDLER – a function called (with the stream as argument) when a non-SPP, non-error packet is received on the socket associated with the SPP connection.

EOM.ON.FORCEOUTPUT – a property whose value should be either T or NIL (the default). If T, then the end-of-message bit is set when the current collection of bytes buffered for transmission is forcibly sent (e.g. by FORCEOUTPUT).

SERVER.FUNCTION – a property that can be used for creating SPP servers. Normally, when a connection is opened with the HOST argument set to NIL, a passive "listener" connection is created. SPP.OPEN will not return until some other host attempts to connect to the socket specified in the SPP.OPEN call.

If the SERVER.FUNCTION property is specified, a new listener (and listener process) is created. SPP.OPEN will return immediately. Whenever another host attempts to connect to the specified socket, a new process and unique SPP connection are created. The function specified by the SERVER.FUNCTION property is run in the top level of the new process. The server function should be a function of two arguments: the first argument is the SPP input stream associated with the connection; the second argument is the SPP output stream associated with the connection.

COURIER.OPEN takes new argument OTHERPROPS.

If OTHERPROPS is non-NIL, it should be a property list of SPP stream properties, as accepted by SPP.OPEN. Any CLOSEFN property on this list is overridden by the value of the WHENCLOSEDFN argument to COURIER.OPEN.

Incompatible change: **SPP.EOMP has been removed.**

The function SPP.EOMP has been removed. The appropriate way to determine whether an SPP stream is open, or whether an End-of-Message or Attention indication has been reached (for input streams) is to use the EOFP function. When EOFP is applied to an SPP stream, it returns one of the following values:

NIL: The connection is open and readable or writable.

T: The connection is closed.

EOM: (Input streams only) The End-of-Message bit was set in the last packet received, and all bytes from the packet have been read. The function SPP.CLEAREOM must be called to clear this condition.

ATTENTION: (Input streams only) An attention packet is waiting. SPP.CLEARATTENTION (below) must be called before the single byte of data associated with the attention packet can be read.

EOFP may wait if the input stream is not currently at EOM or ATTENTION, but no data exists to be read (since the next packet

could be an Attention packet, for example). The function READP in this case returns NIL immediately.

SPP.CLEARATTENTION has been added.

(SPP.CLEARATTENTION *STREAM NOERRORFLG*) [Function]

Clears the Attention packet indication on *STREAM*. This must be called before the single byte of data associated with the attention packet can be read. Causes an error if the stream does not have an attention packet waiting, unless *NOERRORFLG* is non-NIL.

Virtual Memory

VMEM.PURE.STATE permits "freezing" a given virtual memory state.

(VMEM.PURE.STATE *FLG*) [Function]

If *FLG* is true, VMEM.PURE.STATE modifies the swapper's page replacement algorithm so that dirty pages are only written at the end of the virtual memory backing file, leaving the original virtual memory pure. This "freezes" a given virtual memory state, so that Interlisp will come up in that state whenever it is restarted. This can be used to set up a "clean" environment on a pool machine, allowing each user to initialize the system simply by rebooting the computer.

The way to use VMEM.PURE.STATE is to set up the environment as you wish it to be "frozen," evaluate (VMEM.PURE.STATE T), and then call any function that saves the virtual memory state (LOGOUT, SAVEVM, SYSOUT, or MAKESYS). From that point on, whenever the system is restarted, it will return to the state as of the saving operation. Future LOGOUT, SAVEVM, etc., operations are not permitted while in this state.

Note: When the system is running in "pure state" mode, it uses a significant amount of the virtual memory backing file to save the "frozen" memory image, so this will reduce the amount of virtual memory space available for use.

(VMEM.PURE.STATE NIL) returns the swapper to its normal state, so that LOGOUT is again permitted. However, the additional virtual memory backing file space consumed is *not* reclaimed.

(VMEM.PURE.STATE) returns T if the system is running in "pure state" mode, NIL otherwise.

Miscellaneous

DATEFORMAT accepts new keyword: DAY.OF.WEEK .

The DATEFORMAT keyword DAY.OF.WEEK is now implemented It causes the date to look like "date time (day of week)", e.g., "28-Jun-85 12:32 (Friday)". When DAY.OF.WEEK is specified, you can also specify the keyword DAY.SHORT, which results in a 3-letter day.

IDATE ignores the parenthesized day of the week when parsing a date. IDATE also now correctly handles time zone specifications, at least for those time zones registered in the list TIME.ZONES.

Middle button scrolling has been implemented in EDITBM, bitmap editor.

Horizontal and vertical middle button scrolling and display of EXTENT bars in scroll windows has been implemented.

New functions have been added to provide information about the system and loaded patch files.

(PRINT-LISP-INFORMATION) [Function]

Prints out a summary of the lisp version and patch files, for example:

> Interlisp-D version KOTO of 7-Nov-85 00:00:10 on 1108, microcode 5658, machine 222#0.125000.16750#0
>
> Patch files: NIL

In order to implement this, the Common Lisp "Other Environment Inquiries" functions were included. See Steele, p 447, for description of these functions.

(LISP-IMPLEMENTATION-TYPE)
Returns "Interlisp-D"

(LISP-IMPLEMENTATION-VERSION)
Returns the system name and date, e.g.
"KOTO of 07-Sep-85 04:37:12"

(MACHINE-TYPE)
Returns one of "1108", "1132", "1186" etc.

(MACHINE-VERSION)
Currently returns the microcode version and real memory size.

(MACHINE-INSTANCE)
Returns the machine's NS address.

(SOFTWARE-TYPE)
Returns "Interlisp-D"

(SOFTWARE-VERSION)

Returns the time of the Interlisp kernel creation. Unlike the value of (LISP-IMPLEMENTATION-VERSION), this does not change when a MAKESYS is performed.

(SHORT-SITE-NAME)

Returns (ETHERHOSTNAME) or "unknown"

(LONG-SITE-NAME)

Returns the same as (SHORT-SITE-NAME).

New value exists for (MACHINETYPE): DOVE.

On the 1185/6 workstation, the function MACHINETYPE returns the value DOVE.

FIXP, NUMBERP, LITATOM performance has been improved.

Minor changes were made to the data type predicates to increase the speed of these functions. Recompiling old code is recommended but not required.

[This page intentionally left blank.]

3. BUG FIXES

A significant number of bugs have been corrected for the Koto release. Some of these are described below, and affect the following areas:

- TTYIN
- Printing
- Fonts
- Window System
- Stack and Interpreter
- Microcode
- File System
- 1108 Local Disk
- Floppy
- Communications

TTYIN

TTYIN no longer misgauges line height.

TTYIN no longer misgauges the line height when it reaches the bottom of a window that is not an integral number of lines high. Previously, it might have placed the cursor too high, a problem most noticeable following the ? = command.

TTYIN ? = command fixes spelling in user type-in.

TTYIN's ? = command, when the function being asked about was correctably misspelled, now fixes the spelling in the user's type-in as well, not just when retrieving the function's argument list.

CTRL-W backs up over carriage return in TTYIN input.

CTRL-W in TTYIN now is able to back up over a carriage return to delete words on the previous line. Backspace and CTRL-W at the beginning of a non-blank line are also able to back up to the previous line, moving the text in front of them along.

TTYIN command Meta-P works correctly; can reshape window.

The Meta-P command in TTYIN now correctly prettyprints the current type-in, rather than erasing some or all of it. This also means that a reshaped TTYIN input window will redisplay correctly.

Printing

Non-Print Service error message has been modified.

When attempting to print to an NS server that is not running Print Service (including a Print Server that has just been booted), the error message is now "no Such Service" rather than "no Such Program Number."

Direct calls to NSPRINT show the file name in the prompt window.

Direct calls to the function NSPRINT that do not include a DOCUMENT.NAME property in their OPTIONS argument now default the name to the name of the file or stream passed to NSPRINT, rather than NOBIND, for purposes of the printer notification printed in the prompt window.

SEND.FILE.TO.PRINTER creates scratch file for each call.

The function SEND.FILE.TO.PRINTER is now careful to create — and keep track of — a unique scratch file for each invocation. This lets you run more than one hardcopy at once, without one walking on another.

Hardcopy printer menu has been corrected.

A bug that caused entries of type (type host) (e.g. (FULLPRESS JEDI) on DEFAULTPRINTINGHOST to be displayed incorrectly on the Hardcopy printer menu has been fixed.

Fonts

Error in FONTCLASS documentation in Harmony release notes has been corrected.

The description of the FONTCLASS function in the Harmony release notes is incorrect. It stated that the FONTLIST argument should be a list of the form

(<Display Font> <press font> <interpress font> ...).

In fact, the CAR of the FONTLIST argument should be the font number used by PRETTYPRINT to change to the font. The correct form of the FONTLIST argument is therefore

(<font#> <Display Font> <press font> <interpress font> ...).

Window System

CHANGEBACKGROUND

CHANGEBACKROUND has been fixed to return the previous background shade, as documented.

(OPENIMAGESTREAM NIL 'DISPLAY) now works.

It is no longer the case that calling (OPENIMAGESTREAM NIL 'DISPLAY) causes an error. Instead, it creates a stream that points at an untitled window.

DRAWCURVE now handles over-sized brushes the same as DRAWLINE.

DRAWCURVE was changed so that in the case where its brush was an even number, the extra bit is up and/or to the left. It previously had been down and/or to the left. The change makes it compatible with DRAWLINE.

Stack and Interpreter

Stack Overflow

A microcode bug was fixed that had previously caused Lisp to die when it encountered a hard stack overflow (a stack overflow that occurs when you proceed after the "soft" stack overflow break and completely exhaust the stack). Hard stack overflow now correctly results in MP 9319, from which you can run TeleRaid or reset back to top level (see description of MP 9319).

Microcode

Fatal microcode bug for 1132 has been fixed.

A serious bug in 1132 local disk which caused a fatal machine crash (blank screen) when doing a directory listing has been found and fixed.

File System

SAMEDIR

SAMEDIR now knows about device names.

1108 Local Disk

CNDIR returns the correct name when connecting to a subdirectory on the local disk.

Previously, there was a bug that caused (CNDIR '{DSK}<LISPFILES>FOO>) to return {DSK}<LISPFILES>FOO (dropping the final closing bracket) instead of {DSK}<LISPFILES>FOO>. This has been fixed.

You can now continue from "File System Resources Exceeded" errors.

If a "File System Resources Exceeded" break occurs because the 1108's local file system is full, it is possible to either (a) delete some files to free up some space, and continue with OK, or (b) exit the break, in which case (depending on circumstances) the file will either be deleted, or will remain incomplete but deletable.

Floppy

COPYFILES to FLOPPY using * only copies highest version.

The use of * as a wildcard in COPYFILES to or from {FLOPPY} with multiple version files will copy only the version of the file with the highest version number.

FLOPPY.FORMAT in CPM mode has been fixed.

Problems with formatting a single-sided, single density floppy in CPM mode have been resolved.

FLOPPY in CPM mode has changed.

When FLOPPY.MODE was set to CPM, files would be written to the floppy under the convention that a simple <CR> would determine the end of a line. However, on a CPM system (i.e. the Xerox 820-11), the <CR><LF> convention is used instead.

DELFILE in FLOPPY.MODE CPM now updates directory.

In CPM mode, if you use DELFILE on a floppy file, DIR shows the correct directory after the deletion.

CTRL-Z indicates EOF in FLOPPY.MODE CPM.

CTRL-Z works as an end of file specifier.

The space from deleted files is now reclaimed in FLOPPY.MODE CPM.

(SYSOUT '{FLOPPY}) no longer formats bad floppy forever.

Previously, if an unformattable floppy was inserted while trying to transfer a sysout to floppies, the system would try formatting the floppy forever. Now, if (SYSOUT '{FLOPPY}) is given a bad floppy, it will return a message "Bad floppy" and will wait for another floppy to be inserted into the disk drive.

COPYFILE to FLOPPY correctly overwrites pre-existing file.

Using COPYFILE to copy a file to {FLOPPY} with an explicit version number will overwrite a pre-existing file with that same name and version number if it already exists on the floppy.

Communications

SPP.OPEN

SPP.OPEN has been fixed so as not to crash with an MP error code of 9305 when the HOST argument is neither an NS host name nor an NS host address constant.

The use of * as a wild card in COPYFILES to or from {FLOPPY} with multiple version files will copy only the version of the file with the highest version number.

[This page intentionally left blank.]

This section describes new versions of the following:

- Chat
- CMLARRAY
- File Browser
- RS232
- HASH
- SINGLEFILEINDEX
- TCP/IP
- TEdit

Chat

Chat has been moved into the Lisp Library. The following Chat files are of interest:

CHAT.DCOM – the basic Chat kernel.

CHATTERMINAL.DCOM – the basic terminal emulator support package.

PUPCHAT.DCOM – implements Pup-based Chat (use in chatting to Stanford Unix, Xerox DEI, and Stanford Tops-20 Pup implementations).

NSCHAT.DCOM – implements the NS Chat protocol (use in chatting to Xerox file and print servers, and Berkeley 4.3 Unix)

TCPCHAT.DCOM – implements TCPTELNET.

RS232CHAT.DCOM – provides terminal emulation capability using the RS232 port on 1108/1185/1186.

TTYCHAT.DCOM – provides terminal emulation capability using the TTY port on 1108/1185/1186. See the RS232 documentation for caveats on this package.

DMCHAT.DCOM – provides terminal emulator for Datamedia 2500 terminals.

VTCHAT – provides terminal emulator for Digital Equipment Corporation's VT100 terminals.

TEK4010CHAT.DCOM – provides terminal emulators for Tektronix 4010 terminals.

TEDITCHAT.DCOM – provides a simple terminal interface which is backed up by a TEdit stream. The TEdit terminal emulator lets the user scroll the Chat window back and forth, as well as copy-select text out of the Chat window.

To use Chat, you must load: CHAT and CHATTERMINAL, one of PUPCHAT, NSCHAT, TCPCHAT, RS232CHAT, TTYCHAT, and one of DMCHAT, VTCHAT, TEK4010CHAT, TEDITCHAT.

Changes to Chat

If you shrink an active Chat window, it becomes an icon depicting a terminal. If the Chat connection is terminated while the window is shrunk, the icon greys out. At this point, clicking middle button in the icon window offers the same "Reconnect" menu as you get in an inactive Chat window.

Two new variables control placement of Chat windows.

CHAT.WINDOW.REGION [Variable]

If non-NIL, should be a region specifying where the first ("primary") Chat window should be placed.

CHAT.WINDOW.SIZE [Variable]

If non-NIL, is a size, i.e., a dotted pair (width . height) specifying the desired dimensions of any new Chat window (not counting the window, if any, specified by CHAT.WINDOW.REGION). In this case, Chat prompts you to place a box of the specified dimensions, rather than requiring you to shape out a region.

Chat Dribble and New now pre-empt tty.

The Chat Dribble and New commands now correctly switch the tty to the prompting process, so that you don't have to click the prompt window in order to type the requested input.

Handling of string argument passed to INITSTREAM modified.

Chat no longer smashes (destructively consumes) the string that you pass as the INITSTREAM argument.

The "Clear" entry in the middle button Chat menu now works.

CMLARRAY

The CMLARRAY package has been almost completely rewritten. Most known bugs have been fixed (including a number with the function ADJUST-ARRAY). The documentation has also been revised.

Changes to CMLARRAY

Displaced arrays are now required to share only with others of same type.

MAKE-ARRAY now allows displacement to floating point arrays.

The primary array reference function, AREF, now has a SETFN property.

Interlisp type specifiers (such as BYTE) are no longer supported as :ELEMENT-TYPE arguments to MAKE-ARRAY. The equivalent Commonlisp type specifiers should be used instead.

The functions LISTARRAY and FILLARRAY, the nibble functions 4AREF and 4ASET, and the non-error-checking versions of the "fast" accessing functions (\PASET, \PAREF, \8AREF, etc.) have all been removed.

The CMLARRAY file package command no longer exists.

Known Problems:

It is not possible to create a displaced array with MAKE-ARRAY whose storage is in an array of datatype ARRAYP, STRINGP, or BITMAP.

In order to create an array which will later be given as a :DISPLACED-TO argument to MAKE-ARRAY, it is necessary to call MAKE-ARRAY specifying :ADJUSTABLE as T.

Arrays currently have a maximum total size of 64k - 2 16-bit words (32K-1 pointers.)

Future Changes:

In a future release, the "fast" accessing functions (PAREF, PASET, NAREF, NASET, LAREF, LASET, 16AREF, 16ASET, 8AREF, 8ASET, 1AREF, 1ASET) will be removed, when specialization of array access is supported via declarations.

In a future release, the function ASET will be removed, in favor of SETF

File Browser

This release has a completely new File Browser. Some of the interesting changes and bug fixes are as follows:

Changes to File Browser

Window layout is now more compact.

The window layout is completely rearranged and is more compact. The region you are prompted for is the whole region, not just the region for one of the subwindows. The prompt window is independently closeable. The menu of information to display comes up only on request, and is independently closeable. Subcommands appear on roll-out submenus instead of being hidden behind middle-button clicks.

File selection has been improved.

File selection is indicated by a little marker in the left margin, rather than by inverting the entire line. The rules for selection have changed slightly: The middle button always adds a file to the current selection. To remove a file from the current selection, select it with middle while holding down CTRL. Extending a selection with the right button does not include deleted files, unless you also hold down CTRL.

Copy-selection has been improved.

When you hold down a copy key, FileBrowser now correctly follows the mouse as you move it inside the window. The file, if any, underlined when you let up on the mouse is the one whose name is copy-selected.

Options have been added to the Copy and Rename commands.

When the Filebrowser Copy and Rename commands are issued for a single file, the destination can either be given as a regular file name specification, or a directory specification; in the latter case, the file is copied/renamed to the same named file on the specified directory, just as if the command had been given with multiple files selected.

Gratuitous Updates have been eliminated.

The Expunge and Rename commands simply remove files from the display. The Update command itself is now called Recompute, to make its meaning clearer. It has two roll-out sub-commands, "New Pattern" and "New Info", which are used to change the pattern and the information displayed, respectively.

Copy and Rename notice when the new file belongs in the browser, and insert it accordingly.

A Delver (Delete Old Versions) command has been added to the Delete rollout submenu.

New versions of the See command are now available.

The See command by default is a "Fast" See, similar to the function SEE, which does not have the capability of scrolling backwards, and does not require that TEdit be loaded. The "old" See, which laboriously formats Lisp source files and presents them to TEdit, is available on a rollout. There is also an unformatted See for looking at files that might be binary. The Fast versions of See reuse the same display window. If you click See when several files are selected, you get to see them serially, with the option of aborting each file after partially seeing it.

A new mechanism for displaying file count has been added.

The small window that used to cryptically keep track of the number of files total, selected and deleted now is slightly more verbose, keeping track of total number of files and of deleted files, plus, if the displayed info includes a size attribute, the number of pages in the files displayed and deleted.

FB command can request attributes.

You can follow the FB command with the set of attributes you want to see displayed, much as with the DIR command. E.g., FB <foo> SIZE READDATE. The set of default attributes is stored in FB.DEFAULT.INFO, initially (SIZE CREATIONDATE AUTHOR).

Subdirectories are displayed on separate heading lines, not as part of the file name (much as with the DIR command).

Files are sorted by decreasing version.

The newest version of a file appears first. This provides some uniformity across devices that do not normally agree on the order in which to enumerate versions.

For more details, see the File Browser documentation in the Lisp Library Packages manual.

RS232

RS232 has been reimplemented in its entirety. Refer to the new RS232 documentation in the Lisp Library packages. The Koto version of RS232 (for 1108/1185/1186 machines) is called DLRS232C. Subsidiary Lisp Library packages of interest are DLTTY, RS232CHAT, RS232CMENU, TTYCHAT, KERMIT, and KERMITMENU.

DLRS232C and DLTTY implement stream-oriented interfaces to the RS232C, and TTY ports of the 1108 and 1185/1186 processors. (Users of the TTY port need not load DLRS232C.) The TTY port is now supported primarily as an output-only device. No extreme measures are taken to capture all characters which enter the port. The RS232C port (which requires the E30 option on an 1108) performs character buffering and flow control, and should pose no problems to users with high speed connections to other computers.

Programs which use the RS232 stream interface should continue to run with only minor changes. Programs which used the RS232READBYTE/RS232WRITEBYTE interface will have to be rewritten to use the standard stream interface.

RS232CHAT is a stub of the Interlisp-D generic Chat facility. As a result, the DM2500, BT100, TEK4010, and TEDIT chat terminal "emulators" all work with the RS232 port.

TTYCHAT implements a similar stub for the TTY port. It is meant primarily as a means of testing low-speed RS232 devices, such as printers. It will not support a reliable connection to 1200 baud (or faster) modems.

RS232CMENU is a FREEMENU-based interface for controlling RS232 and TTY port parameters. It is available by choosing the "Options" entry of the middle-button Chat menu.

KERMIT implements both Kermit and Modem file transfer programs on top of Chat. While this package will be of primary interest to RS232 Chat users, users of other Chat protocols (such as Pup, TCP, and NS) can also use Kermit/Modem. Documentation is provided in the Lisp Library packages.

KERMITMENU is a FREEMENU-based interface for KERMIT. See the Kermit documentation in the Lisp Library packages.

HASH

GETHASHFILE will now automatically reopen a closed hash file.

Incompatible Change: Koto hash files will are not readable in Intermezzo and vice versa.

SINGLEFILEINDEX

SINGLEFILEINDEX inserts a blank page before the index if necessary to ensure that the index starts on an odd-numbered page, for the benefit of people who print on two-sided printers.

TCP/IP

TCP/IP now supports RFC940-style subnet routing.

If your site employs subnet routing, please refer to the installation instructions at the end of the TCP/IP documentation section in the Lisp Library packages. You will need to perform the TCP configuration step to add subnet routing to your IP configuration file. If you are unfamiliar with subnet routing, refer to the section titled "A Primer on IP Networks," in the TCP/IP document.

The file TCP/IP can now be part of a user-created sysout.

Previously, saving TCP/IP in a sysout while the package was enabled caused problems when the saved sysout was started.

New variable can now identify file type.

TCPFTP.DEFAULT.FILETYPES [Variable]

This variable is an association list, keyed by common file name extensions (e.g. DCOM, BIN, etc.). The CDR of each entry in the list id BINARY or TEXT. The initial value of TCPFTP.DEFAULT.FILETYPES is:

((DCOM . BINARY) (BIN . BINARY) (NIL . TEXT))

This variable is used when TCPFTP is trying to open a remote file for input, and needs to know its type. On operating systems such as Unix and Tops-20, it is imperative to open files in the proper mode. Users may set this variable according to their own needs in their personal or site initialization files.

The file TCPNAMES is now documented.

This file contains routines for translating between file name syntax of various operating systems. Instructions are provided in the documentation for extending the routines to handle new operating system types.

The function TFTP.SERVER creates a "trivial file transfer protocol" server, which can be used to copy files between machines supporting TFTP.

See the TCP/IP documentation for details.

UDP.OPEN now successfully returns socket numbers between 1000 and 65535 when given a SKT# argument of NIL.

The case of host names is now ignored in HOSTS.TXT files.

TEdit

Considerable changes and improvements have been made to the TEdit package. These are summarized below.

Changes to TEdit Functions

The relY argument to an IMAGEOBJ's ButtonEventInFn is now measured from the IMAGEOBJ's baseline.

TEDIT.PARALOOKS now lets you specify baseline-to-baseline spacing between the lines of a paragraph.

The BASETOBASE paralook is a distance in points, between the baselines of adjacent lines of text in this paragraph–regardless of things like type size, line leading, and so on. Paragraph leading, though, does affect line spacing. Setting the BASETOBASE paralook to NIL reverts to the old behavior.

TEDIT.FORMATTEDFILEP now has another potential return: NSCHARS, which signifies that the document contains characters in other than NS character set 0.

This is an indication that you must save the file as a TEdit-format file if you want TEdit to correctly read it back in at a later time. However, it is *not* necessary to save TEdit formatting information for the output file to contain text in proper NS-encoded format, readable by other programs (e.g., SEE, LISTFILES) that accept the full NS character set as text input.

Beginning in this release, all properties that can be specified in the PROPS argument can now be changed with TEXTPROP.

Those that were window properties in the past *are no longer.* For example, to change the LOOPFN during an edit session, you must now use TEXTPROP (or the PROPS argument to TEdit) instead of WINDOWPROP.

Old WINDOWPROP	New TEXTPROP/PROPS Argument
TEDIT.CMD.LOOPFN	LOOPFN
TEDIT.CMD.CHARFN	CHARFN
TEDIT.CMD.SELFN	SELFN
TEDIT.CMD.AFTERQUITFN	AFTERQUITFN
TEDIT.CMD.OVERFLOWFN	OVERFLOWFN
TEDIT.CMD.PRESCROLLFN	PRESCROLLFN
TEDIT.CMD.POSTSCROLLFN	POSTSCROLLFN
TEDIT.CMD.QUITFN	QUITFN

A new option for the PROPS argument, COPYBYBKSYSBUF, has been added.

Normally, text is copied from TEdit document to TEdit document internally. For some applications, you may want to have copied text inserted into the keyboard buffer instead, and read as though it had been typed. To do this, there is a new option for the PROPS argument (and TEXTPROP): COPYBYBKSYSBUF. If that property's value is non-NIL, TEdit will copy characters via the keyboard buffer.

There is a new character look, available via TEDIT.LOOKS, "INVERTED".

If set to ON, the text with that look will appear in inverse video on the screen–but not on hardcopy.

There is a new paragraph look, HEADINGKEEP.

If set to ON, the paragraph so marked will be kept with the beginning of the next paragraph–so that, for example, a heading will stay with the text it heads.

There is a new TEXTPROP, the GETFN.

It is analogous to the PUTFN (and is called with the same arguments); having it return DON'T will abort a GET.

OPENTEXTSTREAM no longer ignores the PROPS argument when itsTEXT argument is an existing textstream.

It is possible to SAME- select from a read-only window now.

TEDIT.GETSYNTAX and similar functions will now take a TEXTOBJ/Text Stream in place of the TABLE argument, and will use the table appropriate to the given TEXTOBJ.

The function TEDIT.SUBLOOKS can be used to change all characters of a given font/size/etc. to another font/size/etc.

TEdit now allows a user-supplied PUTFN.

If you supply a PUTFN, TEdit no longer breaks with "file won't open."

The SEL property of the PROPS argument to TEdit will now accept the value of DON'T, meaning that nothing is to be selected initially.

Changes to TEdit Commands

Incompatible Change: The Koto release of TEdit includes an incompatible change in file formats; the latest file format should minimize backward compatibility problems in the future.

Future additions to menu options such as Paragraph Looks will not affect the ability of old TEdits to read a new file--though any new information would perforce be lost in the process.

To help ease the transition, the PUT menu command has a new sub-option, "Old-Format". Selecting this will save the file in the OLD TEdit format, for backwardcompatibility.

It is no longer possible to LOGOUT while doing a PUT (thereby preventing potential inconsistencies).

Expanding the DATE or >>DATE<< abbreviations now gives a date in the form: month day, year

---

When EXPAND fails to expand an unknown abbreviation, the caret is left where it was.

The default TEdit abbreviations now expand to NS characters.

The abbreviations and their expanded NS characters are:

n	en dash	–
m	em dash	—
b	bullet	•
T	thin space (1/5 em)	| |
d	dagger	†

D	double dagger	‡
'	single close quote	'
'	single open quote	'
-	open double quote	''
"	close double quote	''
s	section sign	§
DATE	the current date	October 23, 1985
>>DATE<<	the current date	October 23, 1985
1/2	one half	½
1/4	one quarter	¼
3/4	three quarters	¾

Note that to expand a multi-character abbreviation, you must select the *entire* abbreviation before pressing EXPAND.

Changes to TEdit Menus

Dotted leaders are now available.

To get them, use the "dotted leader" option when you set tabs from the Paragraph Looks menu. This can be turned off and on independently of what kind of tab you are setting.

Changing page layout now marks the document changed, and in need of saving.

TEdit now supports roman numeral page numbers along with arabic numerals.

You may also specify constant text to appear around each page number (e.g., so that page numbers come out looking like "Page xiv"). These options are available on the Page Layout menu.

TEdit's Page Layout menu now allows you to specify up to eight different kinds of running heads and feet on a page.

TEdit can now display paragraphs in "hardcopy" mode.

Characters are displayed as they'll appear on paper (provided you've set the paragraph margins.) This means that line breaks, tab locations, etc., will be accurate.

The Page Layout menu now supports A4 and legal-size paper.

The Page Layout menu lets you choose the size of the paper you want to print on. The function TEDIT.SINGLE.PAGEFORMAT has

had a PAPERSIZE argument added to it, which may take as its value one of the atoms Legal, Letter, A4, or NIL (defaults to letter size).

The "Hardcopy" and "Press File" items have been removed from the TEdit pop-up menu.

Use the right-button menu's "Hardcopy" option.

TEdit's Expanded Character Looks and Paragraph Looks menus now contain a "NEUTRAL" option.

Selecting it resets the menu so that the other options have no effect when you APPLY. After selecting NEUTRAL, you can select one specific option from the menu that you want changed, then APPLY. This is useful, for instance, for changing only the right margin of a series of paragraphs.

When you close one of several Expanded menus, the remaining menus now snuggle up to the Text Editor window.

TEdit now allows one to start a document at a page number other than 1.

The Page Layout menu has a "starting page number" field. If you fill it in (when setting the layout of the first page), the number you supply will be the first page's number, and succeeding pages will be numbered accordingly.

Changes to TEdit Windows

Split-Window Editing is now provided.

There's a band along the right edge of the window (analogous to the line bar at the left). If you're in that "window" bar, you can use the middle mouse button to split the window, and the right mouse button to unsplit it.

Note: You must be in the split-off section to unsplit. You can't be in the "original" window.

In a future release, the left mouse button will let you move the closest split point around to adjust the allocation of space among windows. For now, you must unsplit and resplit to do this.

TEdit now shows indication of unsaved changes.

TEdit now indicates when a document contains unsaved changes. A "dirty" document will have a star to the left of its title, e.g. "*Edit Window for...."

TEdit no longer brings the edit window to the top solely to flash the caret.

Miscellaneous Changes

This version of TEdit supports Xerox's extended characters.

You can type them in (by setting KEYACTIONs appropriately), save them in files, and print them on Interpress printers.

If you TEdit a file which is already open, you now get the entire document in the TEdit window, instead of just the part beyond the current filepointer.

The time that TEdit spends inside its start-up monitor lock has been drastically shortened .

This means that file-not-found errors and the like will not impede other TEdits from starting up.

TEdit's support for invisible characters, both on the display and hardcopy, has been made considerably more reliable.

Copying character looks (using the "SAME" key) now sets the looks for future type-in, even if no existing characters' looks change.

TEdit's post-paragraph leading now works.

Several scrolling bugs have been fixed.

In particular, a bug in TEdit's screen update code that caused an infinite loop, scrolling the window looking for the caret's new location, has been fixed.