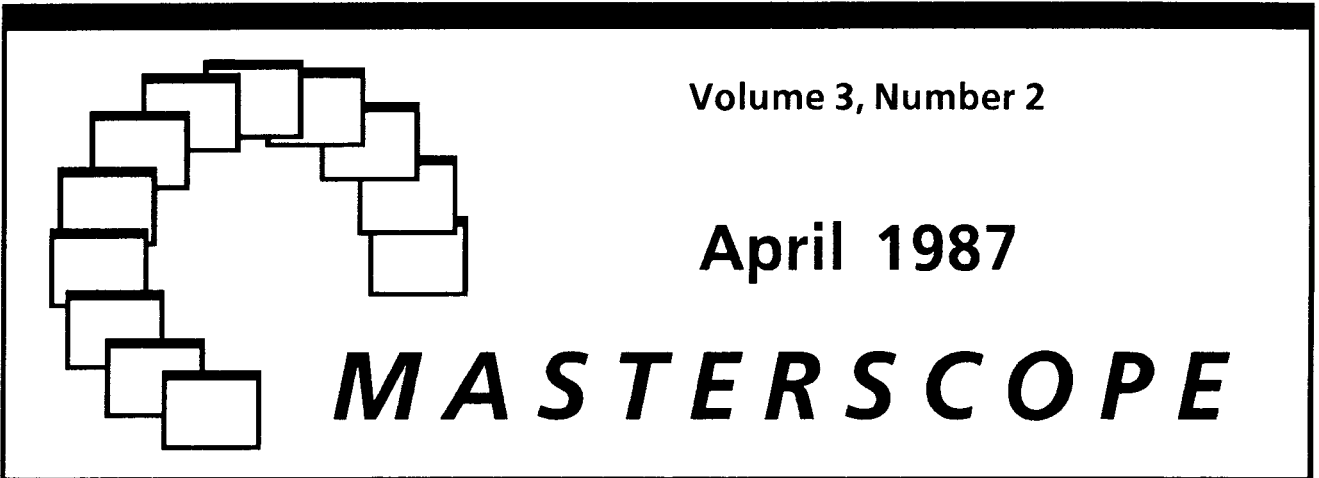




Letter from the Editor

Welcome to another issue of *Masterscope*, the magazine for Xerox AI Systems users. I hope you'll enjoy this issue as much as the last one.

I'd like to thank everyone for the support we're getting in producing it. Keep those letters and articles coming folks, that's what keeps *Masterscope* going. It wouldn't exist without the user community, and we would like to hear from you.

One feature I would like to start is a Users Wish List. Though there's no guarantee that they'll implement any of it, the folks at development in Xerox would like to know what we would like to see in the system, and it would probably help them to prioritize bug fixes and enhancements. Send in any suggestions you would like; and I will compile a list for the next issue. Would you love to see the 1186 handle 3.5 inch floppies? Do you think the documentation is obscure in some parts? Let me know, and we can forward that information in a constructive manner.

Another thing we are going to have in the next issue is an "Ask Xerox" column. If you have questions you would like put to Xerox management, (well, it might not get all the way to the top, but hopefully someone could answer it), please send them in.

The next issue is the big one, the famed "AAAI issue" which for some years was the only issue. This issue will probably get the largest circulation. So..... Become Famous! See your name in print! Show other delegates your article!!! Seriously, folks, I'd like to encourage you to write and let us know what you're doing with the machines, if you've got software to share, or what your local user groups have been up to. It's a lot of fun putting the magazine together, but I still need stuff to put in it.

Arun Welch
Masterscope Technical Editor

Table of Contents

Letter From the Editor	1
Xerox Lisp Articles	
The Lyric Release, By John Sybalsky	3
AI and Education: An Intelligent Tutor for the Space Domain	
Brooks Air Force Base	5
Wizard's Shelf	7
The University Setting	15
Xerox Loops	
An Overview of Xerox Loops	17
News and Announcements From Xerox AIS	
1100 and 1108 Trade-in Program	20
Users' Group Announcement	21
Educational Services Class Schedule	21
Submission Information	22

Xerox Lisp

The Lyric Release

By John Sybalsky
Xerox AIS

We've been working on Lyric for a year. It's in beta-test now and will be delivered to customers early in the third quarter. Following is an overview of major changes and differences between Koto and Lyric and what you can expect to find.

What's new in this release?

First off, this release includes Xerox' implementation of Common Lisp. That, along with fixing the highest-priority problems, has been our main focus for the past year. What we've got is a full, conforming, implementation of the language, along with integration into the programming tools that you're already familiar with. In addition, we've implemented Kent Pitman's proposed standard Common Lisp Error System which provides a hierarchy of error types, error handling, and a consistent way of providing user error recovery.

One outgrowth of Common Lisp is that we've finally bitten the multiple-streams-per-file bullet: effective with this release, you won't be able to use litatom "file names" when you call READ, BIN, and the like—you'll have to traffic in streams. On the other hand, you'll be able to read from the same file in two places at once. This will remove much of the strain of coordinating who's allowed to read what file when in applications.

There's also a new editor for Lisp code. Known as SEdit, it's much more interactive than DEdit: To enter new code, you click with the mouse where you want it, then type away! To delete things, you select them with the mouse and hit the DELETE key; copying and moving also work as you'd expect. For other commands, like FIND, you have the choice of command keys or a menu. When we installed the first prototype of SEdit within the development group, DEdit was out of use within the week (and that was the buggy prototype!). We think people will find this a much more pleasant editor for code and list structures.

To get some commonality between Interlisp-D and Common Lisp at the top level, there is a new EXEC system. It includes a new way of defining EXEC commands and a way to customize each executive to your preferred operating environment. You can have several EXEC windows open at once, perhaps one for Interlisp-D work and one for Common Lisp work; you can switch back and forth, and each will automatically respect the correct syntax.

In the past, RS-232, FX-80 and 4045 printer support have been the source of much trouble. With this release we have fixed RS-232 and have completely new, tested implementations of FX-80 and 4045 support. In RS-232 specifically, the XOn-XOff problems have been resolved and 7-bit character and parity

support have been made to work. The FX-80 driver subsumes all of the old FX-80 library packages, and offers full integration with the right-button Hardcopy menu. While the 4045 driver still only supports a single font, it does work with all manner of line and bitmap graphics now.

How does Common Lisp fit into the environment?

As you can imagine, there is a bit of culture clash between Interlisp-D and Common Lisp! Even so, they seem to coexist well in the single programming environment—due mainly to packages. Because of the isolation the packages provide, existing Interlisp-D applications can continue to run pretty much unmolested, as can pure Common Lisp systems. It's the cross-over that's interesting:

- You can call Interlisp-D functions from Common Lisp & vice versa. So, for example, you can write Common Lisp code that uses windows.
- Most data types can be passed back and forth: numbers, symbols, strings, hash tables, lists, and records/structures. You don't need to worry about whether this is an Interlisp-D string or a Common Lisp string—just pass it around.
- Interlisp-D and Common Lisp use a single error and unwind-protect mechanism, so that each language's error handling constructs will deal usefully with all errors.
- Extensions to Common Lisp, such as the error system, reside in the XCL package so that they're available but don't intrude.
- The programming environment supports Common Lisp.

What's missing in this release?

The basic Lyric sysout is about 500K larger than the basic Koto sysout. To help alleviate the size change, four things were removed from the system and placed in the Lisp Library: MASTERSCOPE, PRESS, MATCH (the pattern-match compiler), and DEDIT. If you use them, you'll need to load them first.

In the course of implementing the unified error handling system, ERRORTYPELST had to be removed. Any code you have that uses ERRORTYPELST will have to be reimplemented using the new error system—you'll have to provide default handlers for any errors you care about.

A number of less-used Library Packages have been demoted to LispUsers: C-150, BIG, BITMAPFNS, BUSMASTER, CHECKSET, CIRCLPRINT, CML, CMLARRAY, COLOR, COMPILEBANG, DECL, IRIS, LAMBDATRAN, PCALLSTATS, and READAIS. This will allow us to concentrate on supporting the parts of the system that people use the most.

AI and Education: An Intelligent Tutor for the Space Domain

Dr. Kathleen M. Swigger

Harry Loveland

Intelligent Systems Branch

Human Resources Lab

Brooks Air Force Base

San Antonio, Texas

In the Intelligent Training Worlds project, the Intelligent Systems Branch (IDI), Human Resources Lab, Brooks Air Force Base, is developing an intelligent tutoring system. This Intelligent Tutor is designed to teach students how to "deduce" a satellite's orbital elements by looking at a graphic display of its ground track. The system also teaches problem-solving skills by monitoring a student's ability to collect, organize, and understand data.

In order to monitor and plan for satellite missions, an Air Force crew must be able to read and understand ground tracks. Ground tracks are two dimensional displays that show the portion of the earth that a satellite covers in one orbit. The "shape" of the ground track is a direct function of orbital elements, so proper understanding of these variables and of the interactions between these variables is critical for anyone interested in satellite operations.

One way to teach students how to deduce orbital elements from a satellite's ground track is to present the various mathematical formulas that are used to compute the orbital elements. In contrast to this approach, IDI's in-house researchers have provided students with a microworld, or simulated environment, that allows them to manipulate various orbital elements and observe how each of the parameters affects the shape of the ground track. If the student is not making satisfactory progress toward understanding the lesson's goal, then the computer intervenes and offers advice, displays examples, or prescribes tasks.

There are several "online tools" that the student can use to explore the ground track world. These include a History tool that allows the student to overlay previously generated ground tracks and note relationships between parameters; an Orbit Window (Figure 1) that displays a two-dimensional representation of the orbit; and a Dictionary that allows students to obtain information concerning terms, etc. The student may select these tools whenever he/she desires and is encouraged to use specific tools whenever the Tutor detects a misconception.

As indicated above, the Tutor is organized to provide a range of guidance for the student. The system is designed to keep track of the student's progress toward, a) the mastery of the relationship between orbital parameters and ground tracks, and b) the demonstration of appropriate investigative behaviors. The system intervenes whenever the student is making erroneous predictions or when he/she is not using the tools effectively. A series of "critics" or rules reside in the system to monitor the student's behavior in these two areas.

The initial prototype is now complete and IDI will be testing the tutor in March 1987 with students from the Space School at Lowry AFB and from the Air Force Academy. This data will then be used to improve the diagnostic portion of the Tutor. Project completion is scheduled for Summer 1987.

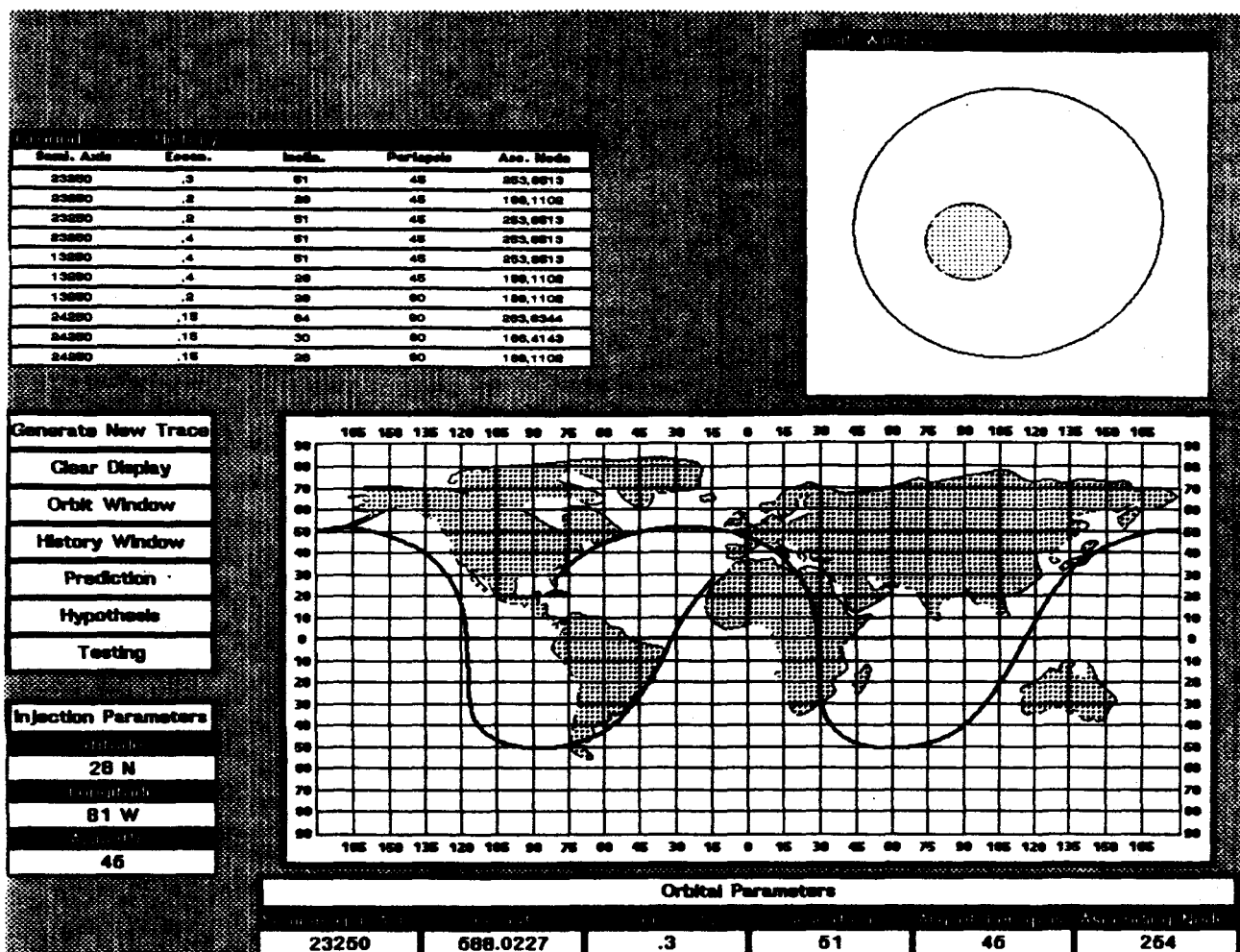


Figure 2: Overview of Ground Trace Microworld

WIZARD'S SHELF

Tips, Tweaks and Miscellaneous Information

This section is devoted to software questions and answers, interesting pieces of code, patches to bugs, and requests for the like. Due to the imminent release of Lyric, I was unsure how to handle this section. Most of the submissions will probably work with Lyric, and I've tried some of them to prove this, but it's left as an exercise to the reader to verify this (don't you love when someone does that?). Some of the tips might be unorthodox or cause your system to crash.

It's also been suggested that we have a section devoted to hardware tweaks and twiddles. This sounds like a good thing. There are probably people out there mucking with the hardware, so The Wizard's Shelf is going to be separated into 2 sections, one for software and one for hardware.

The "rules" for this section are as follows:

1. If you submit software or hardware patches, don't submit things that are copyrighted, untested, or that you know will cause systems to behave strangely.
2. If you are able to answer any of the questions that someone has, then feel free to send me the answer, and I'll feed it to the requester, and include it in the next issue.
3. Neither Xerox, nor I, nor the original authors take any responsibility for what these bits of code do to your system (I have to put this in, obviously). The understanding, however, is that people who submit things for this section will make an attempt to debug them, since they would like others to do the same.
4. If you have got questions you'd like included in the next issue, send them to me. See instructions at the end of the newsletter on how to do this. If you have software that you'd like to make available to other people, this is also the place for it. Send a short description of the software, along with instructions for obtaining it. Note that this isn't a form of free advertising, but rather a way of distributing knowledge among users, so I'm going to have to restrict this to public domain software (software that you only charge for distribution costs, not for software costs).

Software

- ❑ A function for dealing with windows
From Greg Wexler and Rick Martin, Xerox AIS

To have a window confirm with the mouse when it's full rather than with the usual "Press shift" etc., use
(WINDOWPROP window 'PAGEFULLFN 'MOUSECONFIRM)

The PAGEFULLFN property of windows is not one seen in the window record and is stored in the USERDATA field but it is used by the system!

❑ **A function to close all Inspect windows**
From Kent Jones, Allen Corp of America

```
(DEFINEQ (INSPECTICIDE (LAMBDA NIL
  (for window in (OPENWINDOWS) do
    (if (EQ (WINDOWPROP window 'REPAINTFN)
      'INSPECTW.REPAINTFN)
      then (CLOSEW window))))))
```

I add this function to close all Masterscope windows to my background menu.

```
(DEFINEQ (DOWNSCOPE (LAMBDA NIL
  (for window in (OPENWINDOWS) do
    (if (EQ (WINDOWPROP window 'REPAINTFN) 'REDISPLAYGRAPH)
      then (CLOSEW window))))))
```

❑ **From Greg Wexler, Xerox AIS**

1. For users of the function PROMPTFORWORD. If you would like to enable the time out feature using PROMPTFORWORD's URGENCY.OPTION as well as allow the TTY process to exist immediately upon calling PROMPTFORWORD then use the following code:

Within the function using PROMPTFORWORD put

```
(TTY.PROCESS (THIS.PROCESS))
```

and then,

```
(PROMPTFORWORD "Enter some data: " NIL NIL PROMPTWINDOW NIL 20
  (CHARCODE (CR SPACE TAB)) NIL]
```

This seems to work for my applications although I have heard that this doesn't work all the time. I guess it depends on the application and who's and/or how often the mouse process gets changed around. Note also that the cursor does not change to the cursor with the question mark in it.

2. How do you know if someone has buttoned in the titlebar of a window? It's easy, simply put the following code in the window's buttoneventfn.

```
(NULL (INSIDEP (DSPCLIPPINGREGION NIL W)
  (LASTMOUSEX W)
  (LASTMOUSEY W])
```

where W is a window.

Basically, when you button anywhere in the window, the buttoneventfn gets called. If you button on the titlebar, it gets called as well.

Using this fact, the above code will only return T when it gets called AND your mouse is not in the DSPCLIPPINGREGION of the window; so the only other place where buttoneventfn gets called at this stage is in the titlebar.

■ From Christopher Lane, Stanford University

1. The listing of the LOOKUP.NS.SERVER patch that was in the last issue of Masterscope contained a bug. Here's the updated copy:

```
(FILECREATED "11-Feb-87 19:09:52"
  {XDE:MCS:STANFORD}<LISPUSERS>KOTO>LOOKUPSERVERPATCH.;3)
  (PRETTYCOMPRINT LOOKUPSERVERPATCHCOMS)
    (RPAQQ LOOKUPSERVERPATCHCOMS ((ADVISE LOOKUP.NS.SERVER)
      (P (UNMARKASCHANGED 'LOOKUP.NS.SERVER))))
  (PUTPROPS LOOKUP.NS.SERVER READVICE
    (NIL (AFTER NIL (if (EQ 'NSADDRESS (TYPENAME IVALUE))
      then (\PUTBASE IVALUE 5 0)))))
    (READVICE LOOKUP.NS.SERVER)
      (UNMARKASCHANGED 'LOOKUP.NS.SERVER)
```

STOP

2. To print your bitmap in the lower left hand corner (or anywhere on the page) of an Interpress printer, you might try the following:

```
(SETQ STREAM (OPENIMAGESTREAM '{CORE}BITMAP.IP 'INTERPRESS))
  (BITBLT
    (WHICHW) NIL NIL STREAM) (* Or any BITMAP ... )
  (SEND.FILE.TO.PRINTER
    (CLOSEF STREAM) printer)
```

This works on Interpress and Impress printers and should work for Press printers.

3. The Xerox NS file servers store attributes that the Interlisp-D DIRECTORY [DIR] function doesn't know about, one of which is the last reader of a file. The following short patch file makes it possible to know whose been reading your files by doing:

DIR {MCS:}<LANE> READER

Warning: You may not have been fretting about who has been reading your files but once you have the ability to find out you may start; use at your own risk!

```
(FILECREATED "18-Jun-86 08:46:33"
  {XDE:MCS:STANFORD}<LISPUSERS>KOTO>NSREADERPATCH.;3)
  (PRETTYCOMPRINT NSREADERPATCHCOMS)
    (RPAQQ NSREADERPATCHCOMS [(ADDVARS (FILEINFOTYPES (READER
11))
      (\LISP.TO.NSFILING.ATTRIBUTES (READER READ.BY))
    (ADDTOVAR FILEINFOTYPES (READER 11))
    (ADDTOVAR \LISP.TO.NSFILING.ATTRIBUTES (READER READ.BY))
  STOP
```

Note: The '11' part of (READER 11) is the field width that DIR uses when printing this item which can be changed. I used the same width as the AUTHOR property.

4. If you have the full Koto release, then you should already have Greek fonts for your display (and probably sufficient characters in the European set to make a German set).

To test this, make sure you have all of the (for now at least Classic) display fonts with names like CLASSIC*-C46.DISPLAYFONT on a directory that is on your DISPLAYFONTDIRECTORIES list. These font files were on the floppies provided with Koto. Then do the following to see if the fonts are working:

```
(SETQ STREAM (CREATEW))
(DSPFONT (FONTCREATE 'CLASSIC 10) STREAM)
(PRINTCCODE (CHARCODE GREEK, 161) STREAM)
```

Though it seems they should be easier to use, the only intelligible way to make use of the alternate fonts is to use the VIRTUALKEYBOARDS and KEYBOARDEDITOR LispLibrary packages.

The matching printer fonts should already be built into your printer's Classic and Modern fonts. Make sure you have the matching CLASSIC*-C46.WD files on a directory on your INTERPRESSFONTDIRECTORIES list.

The method I use to get random special characters like <= & => (with or without the VIRTUALKEYBOARDS package) is to dump all the characters in a given character set into a scratch TEdit window and then shift select out the ones I want into my document, sketch or whatever.

The following function, given a character set number (like 33 which is where <= and => are found) and a font (like Classic 10 which is fairly complete) will prompt for a region and print out the characters in the particular character set to a scratch TEdit window (after some delay if DSPFONT has to locate and load the character set):

```
(DEFINEQ (SHOWCS
  (LAMBDA (CS FONT)
    (LET ((STREAM (OPENTEXTSTREAM NIL (CREATEW))))
      (DSPFONT FONT STREAM)
      (from 0 to 255 as BYTE from (LSH CS 8) do (BOUT STREAM
        BYTE))))))
```

Of course to do this you must have the font files for the character set in question available via the DISPLAYFONTDIRECTORIES list.

5. This function gets the time off a TCP/IP-only net based on RFC 868.0 Note that the RFC specifies that you read the time from the host as a 32 bit binary number but the actual implementation that was on a VMS VAX supplied the time as an ASCII number, which was perfect for Lisp. However, if you have an RFC 868 time server that returns a binary number, you'll have to change the READ call to something appropriate.

Although I believe it should work, you may have to fudge the 'magic constant' (which corrects the difference between the start of Lisp's time and the protocol's) to correct for your time zone and/or DST.

For the machine I was installing, I tried smashing the normal NS time function with this one to get the machine to find the time on start up, THIS DOES NOT WORK! Unfortunately, TCP hadn't reinitialized by the time the call was made. In fact, PRE/POSTGREETFORMS occurred before TCP's initialization so I ended up advising GREET (and/or LOGOUT) AFTER in order to get the time set automatically.

TCP.TIME.HOSTS should be set to a list of host addresses/names (whatever is acceptable to TCP.OPEN). If you make changes/corrections/additions to the code please let me know so I can make them available to others.

```
(FILECREATED "11-Aug-86 15:31:26" {MCS:MCS:STANFORD}<LANE>TCPTIME.LSP;1)
(PRETTYCOMPRINT TCPTIMECOMS)
(RPAQQ TCPTIMECOMS ((FNS \TCP.SETTIME)
                    (INITVARS TCP.TIME.HOSTS (TCP.TIME.PORT 37))
                    (GLOBALVARS TCP.TIME.HOSTS TCP.TIME.PORT)))

(DEFINEQ
 \TCP.SETTIME
  (LAMBDA (RETFLG)                                (* cd1 "11-Aug-86 15:31")
    (LET (TIME STRING)
      (bind STREAM declare: (SPECVARS STREAM HOST)
        for HOST in TCP.TIME.HOSTS
        when (SETQ STREAM (RESETVAR \TCP.DEFAULT.USER.TIMEOUT
          10000 (TCP.OPEN HOST TCP.TIME.PORT NIL 'ACTIVE 'INPUT
            T))))
      do [SETQ TIME (CAR (NLSETQ (READ STREAM)
        (CLOSEF STREAM)
        (RETURN))
      (if TIME then
        (* * Fudged constant was 2179019800, calculated constant below
        adjusts Jan 1, 1901 (less one year (in seconds) since lisp will not accept
        Jan 1, 1900) to lisp's starting point of "18-Jan-69 19:14:08")
        [SETTIME (SETQ STRING
          (GDATE (SETQ TIME
            (DIFFERENCE TIME
              (CONSTANT (DIFFERENCE (TIMES 365 24 60 60)
                (IDATE " 1-Jan-01 00:00:00 GMT"))
            (if RETFLG then TIME
              else (PROMPTPRINT "[Time reset to " STRING "]")
            T]))
          (RPAQ? TCP.TIME.HOSTS NIL)
          (RPAQ? TCP.TIME.PORT 37)
          (DECLARE: DOEVAL@COMPILE DONTCOPY (GLOBALVARS TCP.TIME.HOSTS
            TCP.TIME.PORT))

      STOP
```

□ From William Pickles, AI Ltd.

For anyone who is interested, the following code will support the use of the abbreviation FILE in a TEdit document. I like to put a footer in a document along the lines of:

"this file is {DSK}<W-FILES>COURSE.TEDIT;2"

(FILECREATED "27-Jan-87 15:05:08" {DSK}<W-FILES>FUTURES>TEDITEXPANDFNAME.;1
836

```

changes to: (VARS TEDITEXPANDFNAMECOMS TEDIT.ABBREVS)
              (FNS \EXPAND.FILE.NAME))
(PRETTYCOMPRINT TEDITEXPANDFNAMECOMS)
(RPAQQ TEDITEXPANDFNAMECOMS ((ADDVARS (TEDIT.ABBREVS
              ("FILE" . \EXPAND.FILE.NAME)))
              (FNS \EXPAND.FILE.NAME)
              (DECLARE: DONTCOPY EVAL@COMPILE
              (FILES (LOADCOMP) SYSEDIT))))
(ADDTOVAR TEDIT.ABBREVS ("FILE" . \EXPAND.FILE.NAME)) (DEFINEQ
(\EXPAND.FILE.NAME
  [LAMBDA (A B C D)          (* "WILL" "27-Jan-87 14:56")
    (MKSTRING (fetch (STREAM FULLFILENAME) of
      (fetch (TEXTOBJ TXTFILE) of (TEXTOBJ A)))
    (DECLARE: DONTCOPY EVAL@COMPILE
      (FILESLOAD (LOADCOMP)
        SYSEDIT))
    (DECLARE: DONTCOPY
      (FILEMAP (NIL (490 744 (\EXPAND.FILE.NAME 500 . 742))))))
  STOP

```

■ From Henning Christiansen, Roskilde University Centre, Denmark

Window and Screen Coordinates - Back and Forth

For some reason, the Interlisp-D system has no clean way to get back and forth between the coordinate systems of the screen and specific windows. The following nice and tricky functions will do the job.

Imperial tests - fast mouse movements and clicks etc. - have proved that the functions are secure despite their inherent time-dependency

```

(DEFINEQ
  (SCREEN.TO.WINDOW.COORDINATES (LAMBDA (POINT WINDOW)
    (* * Transform screen coordinates to window dittos)
    (create POSITION
      XCOORD <-(IDIFFERENCE (fetch XCOORD of POINT)
        (IDIFFERENCE LASTMOUSEX (LASTMOUSEX
          WINDOW))))
      YCOORD <-(IDIFFERENCE (fetch YCOORD of POINT)
        (IDIFFERENCE LASTMOUSEY (LASTMOUSEY
          WINDOW))))))
(DEFINEQ
  (WINDOW.TO.SCREEN.COORDINATES (LAMBDA (POINT WINDOW)
    (* * Transform window coordinates to screen dittos)
    (create POSITION

```

```

XCOORD ←(IPLUS (fetch XCOORD of POINT)
              (IDIFFERENCE LASTMOUSEX (LASTMOUSEX
                                         WINDOW)))
YCOORD ←(IPLUS (fetch YCOORD of POINT)
              (IDIFFERENCE LASTMOUSEY (LASTMOUSEY
                                         WINDOW))))))

```

❑ From Roger Beeman, Boeing

A couple of things dealing with Interpress printing:

1. The "Window Image" at the top of the page can be eradicated with

```
(ADVISE 'INTERPRESSBITMAP 'BEFORE
      (if (STREQUAL TITLE "Window Image") then (SETQ TITLE NIL)))
```
2. Specifying SCALE in HARDCOPYW as a rational can keep your PrintService from interpolating your bitmap, e.g. '(6 . 25) for 300 dpi, '(12 . 25) for 150 dpi, '(18 . 25) for 100 dpi, and '(24 . 25) for 75 dpi.

[Note: This will change in Lyric]

❑ From Johan Lilius, University of Helsinki

I have set up a redistribution site for BITNET users of the info-1100 mailing list.

It is mainly intended for Scandinavian and European users, but as long as there are only a few North-American users, I suppose they could also subscribe from here.

You have to issue the commands :

- on VMS: send listserv@finhutc signup info1100 "your name"
- on IBM: tell listserv at finhutc signup info1100 "your name"

When you are satisfied that everything works you can mail to Chris Schmidt and request to be removed from his mailing list.

You can also receive more info on the networking facilities of FINHUTC by issuing the commands info and help to SILMARIL@FINHUTC and LISTSERV@FINHUTC.

❑ From Jordan Pollack, NMSU

Lest anyone believe that Larry Masinter is the only creator of nifty IDLE functions, enclosed is a more aesthetic one than MELTDOWN:

```

(DEFINEQ
(ACEBOUNCE
  [LAMBDA (W)
    (LET ([MX (DIFFERENCE (WINDOWPROP W (QUOTE WIDTH))
                          (fetch BITMAPWIDTH
                                of (CAAADR (CAR ACE.CURRENT.SEQUENCE))

```

```

[MY (DIFFERENCE (WINDOWPROP W (QUOTE HEIGHT))
  (fetch BITMAPHEIGHT
    of (CAAADR (CAR ACE.CURRENT.SEQUENCE]
  X Y DX DY)
  (SETQ X (RSH MX 1))
  (SETQ Y (RSH MY 1))
  (WHILE T
    do (SETQ DX (RAND -25 25))
      (SETQ DY (RAND -25 25))
      (for FRAME in ACE.CURRENT.SEQUENCE
        do [SETQ X (MIN MX (MAX 0 (PLUS X DX)
          [SETQ Y (MIN MY (MAX 0 (PLUS Y DY)
            (for BM.OFF in (CADR FRAME)
              do (BITBLT (CAR BM.OFF)
                NIL NIL W
                (PLUS X (CADR BM.OFF))
                (PLUS Y (CDDR BM.OFF))
                NIL NIL (QUOTE INPUT)
                (QUOTE PAINT)))
          (BLOCK (CAR FRAME))
          (for BM.OFF in (CADR FRAME)
            do (BITBLT (CAR BM.OFF)
              NIL NIL W
              (PLUS X (CADR BM.OFF))
              (PLUS Y (CDDR BM.OFF))
              NIL NIL (QUOTE INPUT)
              (QUOTE ERASE]))

```

Then load the file ACE-FOUETTE from LISPUSERS and
 [PUSH IDLE.FUNCTIONS (QUOTE (DANCER (QUOTE ACEBOUNCE)]
 to add the function to your choice of IDLE functions.

Hardware

For obvious reasons, doing strange things to the hardware may void your warranty or service contract. I would suggest that you make sure that you only do these things with the approval of your service rep. Sometimes the reps are interested in this sort of thing too, and might be willing to help or offer advice.

□ From Anonymous...

The 1186 5-1/4" floppies can be replaced with Toshiba 3-1/2" floppies with no changes to the hardware. The software, unfortunately, cannot address more than it can on the 5-1/4's, but with a little tweaking of the device driver should be able to. The even more interesting implication of this is that since Toshiba has announced the release of a 4 Mb 3-1/2" drive in August, a sysout will someday be able to fit on 2 floppies !! Yup, that's 2 floppies. Little ones. This really thrills me, since the Lyric sysout is currently 14 floppies, and it's a pain to load.

THE UNIVERSITY SETTING

Because of the number of educational institutions using the Xerox D-machines, it has been suggested that we start a column describing how, and for what projects, the machines are being used. Universities tend to have a rather unique environment in that the users are generally on the machines for a relatively short period of time and the number of users sharing a machine is fairly high. On the other hand, some of the constraints are relaxed because the machines aren't being used for a specific project.

Following is a description of the environment at the University of Wisconsin.

David Neves
Computer Sciences Department
University of Wisconsin at Madison
Madison, WI 53706
Usenet: {allegro, heurikon, ihnp4, seismo}!uwvax!neves
Arpanet: neves@rsch.wisc.edu

Hopefully the following description of the software we have developed for our instructional use will help you create something better for yourself. Much of our software was hacked together by an undergraduate and contains specific things for our environment. If you would like to have the software, send me an 8-inch Xerox floppy disk and a stamped, self-addressed floppy envelope.

We load several files into a naked Koto sysout to create our student sysout. These files are Newlogin, Init, Help, and Reserve. The first three files are somewhat specific to having a Clearinghouse.

- LOGIN

A student logs in by calling up the background menu and selecting LOGIN. LOGIN is installed by loading file NEWLOGIN (which is already in the student sysout). The function NLOGIN3 (in file INIT) is called and a system patch file (called PATCH.DCOM) is loaded. This way we can make small changes to everyone's sysout or we can issue a common message for all classes. In order to load the patch file, we have to temporarily log in so that we can access the files server. The function SECURELOAD does this. SECURELOAD does an unbreak and traps interrupts so that the student can't abort out early and wreck the directory.

After the student selects LOGIN, a new menu comes up with a list of classes. This list is generated (in function classlogin) by going out to the Clearinghouse and looking for all groups that start with "CS." We have five classes using the machines this semester.

After the student selects the class, a new menu with names of students in the class comes up. This list is generated by going to the Clearinghouse and asking for all members of that class. Since we use the full student name as a login name and some student names can be longer than 20 characters, it is important to provide a menu facility for LOGIN.

When the student selects his/her name, we then prompt for their password in the Promptwindow and verify it with the function CHECKPASSWORD. We allow up to three tries to get the password typed correctly.

The software then computes the location of the student directory by taking various parts of the student name and combining with the name of 1 of 2 file servers. e.g. it might try {uwinst:}<david neves> or {uwinst:}<neves> or {uwinst:}<david> or {uwresearch:}<same as with uwinst:>. We make sure when we install students that their directory name looks something like their real name.

Once the student directory has been determined, we connect them to it. Then the software computes the location of the class directory (which is always ' "{uwinst:}<c" classnumber ">". We use the class directory to hold a MOTD (message of the day) file that is read in whenever a student logs in. The MOTD file can print out a message for the student in the login/help window (by having a call to PRINTOUT in the P part of the COMS file variable) or can define library functions for the latest programming assignment.

By doing all of the above, we achieve the functionality of logging into a Unix system where the system checks the password, the student gets automatically connected to his/her directory and a message is printed out upon login. In addition, we can load code into the student's environment.

- LOGOUT

A student logs out by going to the background menu and selecting LOGOUT. Logging out does two things. First a MAKEFILES is executed to make sure that all files or functions will be saved out. Second, we execute (LOGOUT T). By using (VMEM.PURE.STATE T), which is set in our student sysout, and (LOGOUT T) we insure that each student booting a machine will get a virgin sysout. We had all sorts of problems when we required that students use COPYVM of systems tools to copy new partitions onto dirty ones. Many students forgot to use COPYVM or did it incorrectly by copying the dirty partition onto the clean one.

- HELP file

To get a bit of help, the student can select HELP/MISC from the background menu. The help isn't that useful but another menu comes up that allows the student to change his/her password or to connect to another directory on the file server. The directory function knows where the library, class, and student directories are located. I think there might be something a little wrong with our password changer but Xerox doesn't want to show us their code to do it correctly.

- RESERVE

We found that we needed an online reservation system because students would cheat with a paper one. Students call this up by selecting RESERVE from the background menu. After RESERVE is selected, there

appears a screen image of a reservation sheet with times along the y axis and terminal numbers along the x axis. Only a single day can be shown. The student can select a different day's sheet by selecting the name of the day from the menu. In addition, the student can select "make reservation" or "remove reservation."

If you use our software you will need to make the following changes in the function NRESERVE:

- The number of terminals from 13 to whatever you have.
- Change the Boundp check on CURRECTSTUDENT to a check on USERNAME.
- Change the file prefix from "{uwresearch:}<reserve>aa" to wherever you are going to store the reservation files. A small bug/problem: this directory must be read/write accessible by all students.

When the student QUITs out of the program a deletion is done of old reservation files.

If you have any questions, feel free to ask. If you look at our software and have any suggestions or bug fixes, please let me know about them.

Xerox LOOPS

Xerox Loops will be available in the second quarter of this year as a supported product. Release 1.0 of Xerox Loops is the first release of Loops backed by the full range of Xerox Customer Support services. The list price of \$6,500 includes a full year of Loops Software Maintenance for those customers with valid Lisp Software Maintenance contracts. We are currently offering Xerox Loops at promotional prices; ask your Xerox AIS Sales Representative for details.

Release 1.0 will be fully compatible with the Koto release of Xerox Lisp. Release 1.0L, which will be available soon after the introduction of the Lyric release of Xerox Lisp, will be compatible with Lyric. Release 1.0L will be provided without extra charge to customers purchasing Release 1.0.

The following is a brief description of Xerox Loops.

LOOPS OVERVIEW

Loops is a multiple paradigm programming system designed to run on the Xerox series of Artificial Intelligence Workstations. Loops extends the function oriented programming style of Interlisp-D with additional paradigms for Object and Access oriented programming.

Loops is smoothly integrated within the Xerox Artificial Intelligence Environment, and along with Interlisp, Common Lisp, and Prolog, is one of the basic set of programming facilities available to users of our workstations.

The major components of Loops and a description of each follows.

OBJECTS

Object oriented programming is the primary style of programming within Loops. In contrast to Functional oriented programming, where programs are partitioned primarily into procedures and separate data, an object oriented program is organized around entities called objects that have aspects of both procedures and data. Objects have local procedures (called methods) and local data (or variables). All of the action in this paradigm comes from sending messages between objects. Objects provide local interpretation of the message form.

In object oriented programming, behavior is determined by responses of instances of classes to messages sent between these objects with no direct access to the internal structure of an object. This approach makes it convenient to define program interfaces in terms of message protocols. LOOPS provides:

- inheritance* of instance behavior and structure from multiple super classes.
- user extendible property list* descriptions of classes, their variables, and their methods.

Additionally, Loops supports the concept of *Virtual Copy*. Virtual copy provides a way to use a network of interconnected objects as a prototype by making virtual copies of it. The virtual copy is created incrementally. The copy is virtual in that the contents of each instance are not completely copied at creation time. It inherits default values from the original, thus continuing to share the parts not modified in the copy. The copy is also virtual in the sense that only those instances needed in the processing are copied. This facility permits several solutions for a particular problem to be quickly designed and compared, while still maintaining an unchanged original

ANNOTATED AND ACTIVE VALUES

Annotated Values are a way of associating behavior and annotations with data. In keeping with the object oriented programming style of Loops, these annotations are objects. Annotation objects are called *Active Values*. When a variable containing an annotated value is accessed, a message is sent to the active value. This mechanism is dual to the notions of messages: messages are a way of telling objects to perform operations which can change their variables as a side effect; active values are a way of accessing variables which can send messages as a side effect.

GAUGES

Loops *Gauges* are a graphical representation of a series of standard display devices (such as dials, meters, LCDs, vertical and horizontal scales) used as tools to monitor the values of variables. They can be thought of as probes inserted onto the variables of an arbitrary Loops program. Gauges are defined in Loops as classes and are driven by active values.

Gauges are an important part of Loops and serve as both a user interface to a Loops program, and as a debugging tool for the programmer. Gauges assist programmers to understand the dynamic nature of the programs they are writing. As a user interface tool, Gauges can be used to display, in either analog or digital form, various data that may be rapidly or slowly changing. An alarm system is implemented in order to notify the user when a particular gauge reaches a predetermined level. Additionally, Gauges can be used as a control device that gives the user a convenient method for setting values or for inputting information into a system.

USER INTERFACE

Since Loops is an integrated part of the Interlisp-D system, the user interface of Loops is based on the same graphics intensive, high bandwidth style of user interface, and the Interlisp-D facilities will be available to the Loops user.

Within Loops, the major interface tools provided are *Browsers*. Browsers are two dimensional graphical structures that display the hierarchical relationship or inheritance among various pieces of the system. Browsers are interactive programs that permit users to further examine and access information by selecting specific nodes within a browser.

There are a number of different browsers built into Loops. The most general class is the LatticeBrowser which defines the overall structure of the Loops Browsers and is tied to Interlisp via the Grapher package. Specific browsers exist that let the user examine the relationships between Classes, Instances, Paths, Files, Supers, and Metas. Additionally, browsers are user extensible and can be modified to examine other relationships or links within the system.

INTERLISP INTEGRATION

Loops is smoothly integrated within the Interlisp-D environment and, therefore, has access to the standard system capabilities. Loops relies on a number of Interlisp facilities including Tedit, Dedit, Masterscope, Inspect Package, File Package, Break Package, Windows, and the Programmers Assistant.

PERFORMANCE

Extensive performance measuring and tuning has been done to Xerox Loops. Based on our measurements, both message sending and variable access are substantially faster in Xerox Loops than in the earlier, unsupported Buttress Loops. We were able to shrink the amount of space that Xerox Loops objects consume. Garbage collection performance has also been improved by a number of alterations to the low levels of Xerox Loops. As a result, Xerox Loops is both faster and smaller than Buttress Loops.

NEWS AND ANNOUNCEMENTS

FROM XEROX AIS

TRADE-IN PROGRAM FOR YOUR 1100 AND 1108

AIS is offering an outstanding, new 1100/1108 trade-in program. To take advantage of this offer you must order a minimum configuration of an 1186-103 with 3.7 Mb of RAM. The order may include additional items such as PC Emulation, transceivers, drop cables, and other software though these items are not a condition of the trade-in offer. If the currently owned 1100/1108 is covered by a Software Support Maintenance Agreement, the agreement contract will be transferred to the newly purchased 1186. Full Service Maintenance Agreements (hardware maintenance) must be cancelled on the 1100/1108 being returned and the remaining funds on the contract will be refunded to the customer. A new FSMA must then be ordered to cover the 1186. A maximum of one trade-in unit may be applied against each unit purchased. This offer will be in effect until further notice.

The price of the 1186 ordered will be at the current prevailing rate with normal discounting practices in effect (i.e., negotiated customer/quantity discount or Educational Institution discount). The base amount of the 1100/1108 trade-in credit is \$8,000. The \$8,000. credit will be applied before any discounts are applied.

Contact your local Xerox AIS Sales Representative to submit a purchase order. When the purchase order is received, AIS will ship and invoice you for the equipment on the purchase order. At your convenience, ship your trade-in unit to Pasadena. When the unit has been received in Pasadena, a check will be prepared and sent to you. At that point, AIS will transfer any outstanding software contracts to the 1186.

Ask your Sales Representative about financing through Xerox Corporation. Financing is available to all qualified customers and can be used to purchase the 1186 on this trade-in offer.

1108S FOR EDUCATIONAL INSTITUTIONS

1108-105s and 1108-110s are now being offered to Educational Institutions at a greatly discounted price. This offer will be limited to the 1108-105 (42MB disk drive and 1.5 memory) and 1108-110 (42MB disk drive and 3.5 memory) configurations only. The 1108s are totally reconditioned and carry our regular 90-day warranty. Check with your Local Xerox AIS Sales Representative for more details.

USERS GROUP

As promised at our User's Group (Birds-of-a-Feather) meeting at AAAI in Philadelphia, Xerox Artificial Intelligence Systems is starting to transition the User's Group to be an independent entity. The first move in this effort is to have a meeting separate from AAAI. This first meeting will be held in Orlando, Florida on June 1st under the auspices of EDGE International.

EDGE is an independent association of Xerox systems users. It functions as an innovative, informative group whose members, working in concert with Xerox management, experience a quantum leap in their ability to use our systems productively.

EDGE conferences are held twice each year. The three-day Spring and Fall events are focal points for all EDGE activities. The programs include panel discussions, software and hardware release discussions, innovative uses of the 8000 system devices, demonstrations introducing new products and techniques and Special Interests and Knowledge Exchange Groups. This Spring, the ninth EDGE conference will be held in Orlando, Florida, May 31 - June 3. The registration fee for the entire conference is \$350.00.

We plan to meet on Monday, June 1, at 1:45 p.m. for a half-day working session: the goal is to set the stage for an organized, functional, fully participative, mutually supportive AI National Users group at the tenth EDGE conference. Input from each of you is strongly encouraged. If you plan to attend, if you are considering attending or if you need more information, please contact Paula Burke at (818)351-2351 or at the 800 numbers.

We hope to see you in Orlando.

EDUCATIONAL SERVICES

Educational Services has designed a 3-1/2 day Conversion Seminar to help our users convert to Xerox Common Lisp (XCL). The Seminar will cover XCL and the new aspects of the programming environment available in Lyric. Current capabilities of the system regarding code portability i.e., pure Common Lisp to XCL, pure XCL to the outside world and Koto Interlisp to Lyric Xerox Lisp will be covered. In general the Seminar will direct you in the entire transition process to Xerox Common Lisp.

We are also offering a 2-day lecture only version of the Conversion Seminar. The lecture-only Conversion Seminar will be given in cities throughout the country or as an on-site class. The workbook for this special class will include a floppy disk with source code for the class exercises and answers. Contact your Xerox AIS Sales Representative or the Xerox Educational Services Coordinator at extension 2676 at the toll free numbers for exact dates and locations of the Conversion Seminar and Conversion Lectures.

Schedule of Xerox AIS Educational Services classes through August.

DATE (week of)	CLASS	LOCATION
27 April	Conversion Seminar	Pasadena, CA
4 May	LOOPS I: Programming in Xerox LOOPS	Pasadena, CA
11 May	Prolog I: Introduction to Xerox Quintus Prolog	Pasadena, CA
1 June	LISP I: Introduction to Xerox Lisp	Pasadena, CA
8 June	LISP IB: Intermediate Xerox Lisp Programming	Pasadena, CA
15 June	Conversion Seminar	Pasadena, CA
27 July	LISP III: Advanced Xerox Lisp Programming	Pasadena, CA
3 August	LISP I: Introduction to Xerox Lisp	Pasadena, CA
10 August	LISP IB: Intermediate Xerox Lisp Programming	Pasadena, CA
17 August	LOOPS I: Programming in Xerox LOOPS	Pasadena, CA

SUBMISSION TO NEWSLETTER

Masterscope is published quarterly by Xerox Artificial Intelligence Systems, 250 North Halstead Street, Pasadena, CA 91109. Opinions expressed herein are those of the authors and do not necessarily reflect Xerox Company Policy.

Masterscope is intended as a communications forum, providing interaction between customers/users of Xerox AI products and between the users and Xerox AIS Management. Please send comments and submissions for future *Masterscope* issue to:

Via network:

Welch@Ohio-State.ARPA
 Welch@Ohio-State.CSNet
 {ihnp4,cbosgd}!osu-eddie!welch

or:

AINewsletter ↑ .pasa@Xerox

Via US Mail:

Arun Welch
 Laboratory for AI Research
 Computer Information and Science Department
 Ohio State University
 2036 Neil Avenue Mall
 Columbus, Ohio 43210-1277

MAILING LIST UP-DATE

It's time to up-date the Masterscope mailing list. If the name and/or address on your copy are incorrect or if there are others in your company that would like to receive Masterscope, please complete the following form and return to:

Paula Burke
Masterscope
Xerox AIS
250 North Halstead Street
P.O. Box 7018
Pasadena, CA. 91109
or
Burke.Pasa@Xerox

Name _____
Company _____
Address _____

Phone # _____
Electronic Address _____
(please indicate full net address)

Xerox AIS Software Support
Toll Free Numbers
(800)228-5325 Outside California
(800)824-6449 Inside California

Xerox Artificial Intelligence Systems
250 North Halstead Street
P.O. Box 7018
Pasadena, Ca. 91109-7018