To: Lisa Software, Hardware, Pubs & NPR, and Lisa Users
From: Rich Page
Ralph Sahm
Ken Friedenbach
Date: April 11, 1983
Subj: Release 12.0 of the Monitor Pascal Devlopment System

This release contains contributions from Bruce Daniels, Sussanah Lam, Chris Franklin, Al Hoffman, Fred Forsman and David Hough. Due to lack of space on the boot disk we were not able to also include IUFPLIB. This can be obtained separately.

Overview

This release of the Monitor provides a multi-programming environment in which programs and exec files can be executed in either foreground or background. This release includes the Pascal compiler which supports Classses. This release also includes numerous bug fixes. The loader has been enhanced to implement code remembering. The Monitor has been enhanced to remember directories. The user should notice a 2x to 4x improvement in system through put (ie. typical process launch is quite short). This release provides a viable Twiggy based development system.

Installation

This release consists of a complete development system contained on a single Twiggy. Simply copy the contents of the Twiggy onto your root volume (ie. #5: of the working device) and reboot with the new boot disk.

Notes:

The files listed below in the section called Boot Files do not need to be copied to your root volume.

The process above assumes that your root volume is named LISA:. If you have a volume named LISA: which is not your root volume you must have a copy of the Mouse Editor files and Font files on LISA:.

The Intrinsic.LIB on this release has been taken from A5.2.5. The development system libraries (ie. IUPASLIB & OBJIOLIB) are compatible with releases 10.2 and 11.x. If you wish to use a different Intrinsic.LIB remember to Install these files.

If you wish to use this release with a Lisa 1.5 replace the CONFIG.DATA file with the file CONFIG1.5.DATA.

Directory

Boot Files

MON.LOADER	10	13-Jan-83
CONFIG.DATA	1	30-Mar-83

```
BOOTFILES.DATA      1 21-Feb-83
LISABUG.OBJ        63 10-Apr-83
LISABUG2.OBJ       29 27-Feb-83
DRIVERS.OBJ        16 23-Mar-83
MS.OBJ              5 28-Mar-83
UARTDRVR.OBJ        2 23-Apr-82
LOADER.OBJ         64  8-Apr-83
TWGDRVR7.OBJ        4 25-Mar-83
MONITOR.OBJ        38 10-Apr-83
MONITOR.SYMBOLS    12 10-Apr-83
CONFIG1.5.DATA      1  8-Apr-83

Start Up Files
--------------
MON.MISCINFO        1 12-Jun-80
MON.STARTUP         4  8-Oct-81
MONSTART1.OBJ       3  4-Jun-81

System Utilities
----------------
FILER.OBJ          91  6-Apr-83
SYSMGR.OBJ         29  6-Apr-83
MOVESOROC.OBJ       4  7-Jun-81
CHANGEMEM.OBJ       2  1-Apr-83
MOUNT.OBJ           4 24-Sep-81
FORMATTER.OBJ      12 28-Mar-83
SHELL.OBJ          11 10-Apr-83

Library & Link Files
--------------------
CALLS.OBJ           3 18-Nov-82
LOADER.IMAGE        1  1-Jun-81
MPASLIB.OBJ        48 21-Jan-82
NOFPLIB.OBJ        18 21-Jan-82
INTRINSIC.LIB       8  8-Apr-83
OBJIOLIB.OBJ      103  6-Apr-83
IUPASLIB.OBJ       15 14-Feb-82

Development Files
-----------------
COMPILER.OBJ      180  8-Apr-83
PASERRS.ERR        12  7-Apr-83
CODE.OBJ          103  7-Apr-83
IULINKER.OBJ       56 10-Apr-83
LINKER.OBJ         32  8-Apr-83
N68K.OPCODES        8  4-Feb-83
N68K.ERR            6  6-Apr-83
ASSEMBLER.OBJ      80  6-Apr-83

Development Utilities
---------------------
BYTEDIFF.OBJ        4 28-Mar-83
CHANGESEG.OBJ       3 28-Mar-83
CHANGETRAP.OBJ      4  8-Mar-83
CODESIZE.OBJ       13  8-Apr-83
DIFF.OBJ           14 28-Mar-83
DISKCOPY.OBJ       11  8-Mar-83
DUMPHEX.OBJ         5  8-Mar-83
DUMPO.OBJ           8 10-Jun-81
DUMPOBJ.OBJ        35 28-Mar-83
```

```
FILEDIV.OBJ          16 25-Feb-82
FILEJOIN.OBJ         15 25-Feb-82
FINDID.OBJ            4  8-Mar-83
FINDWORD.OBJ          2  8-Mar-83
GXREF.OBJ            12  8-Mar-83
IUMANAGER.OBJ        22 28-Mar-83
OBJDIFF.OBJ          26 20-Aug-81
PACKSEG.OBJ          12 28-Mar-83
PATCH.OBJ             7  8-Mar-83
PRINTSIZES.OBJ        3 18-Mar-83
REUSE.OBJ            19  8-Mar-83
SEGMAP.OBJ            4  8-Mar-83
SETSIZES.OBJ          5 18-Mar-83
SXREF.OMIT.TEXT       4 23-Sep-82
SXREF.OBJ            10 28-Mar-83

Editor Files
------------
EDITOR.OBJ          199  7-Apr-83
EDIT.MENUS.TEXT       6 22-Feb-83
PAPER.TEXT            4 21-Jun-82
UCSDEDITOR.OBJ       71  8-Apr-83

Font Files
----------
CENTURY12R12S.F       3  5-Aug-82
CENTURY12R10S.F       3  5-Aug-82
CENTURY12RPS.F        3  5-Aug-82
FM.MN.HEUR            2 17-Jun-82
SYSCURSORS.F          4  5-Aug-82
SYSTEXTS.F            5 30-Sep-82
TILE7R20S.F           3  5-Aug-82
TILE12RPS.F           3  5-Aug-82
TILE12R12S.F          4  6-Aug-82
TILE7R15S.F           2  5-Aug-82
WMFONTS.F             8  8-Nov-82
```

Monitor Enhancements

The enhancements to the Monitor, boot and system files are as follows:

1) Low memory has been re-arranged so that release 8.5 of the Drivers can be used. This results in a larger type ahead buffer (ie. back to 32 characters).

2) At address $D00-&176 there is a pointer to the start of three 12 byte buffers. These buffers are used as follows by the Twiggy driver:

   a) At ($D00-&176) there is a 12 byte buffer for the header of the last block read by the twiggy driver.

   b) At ($D00-&176)+&12 there is a 12 byte buffer for the header of the next block to be written by the twiggy driver (for blocks 1 to 1701). This buffer is initialized to all zeros.

   c) At ($D00-&176)+&24 there is a 12 byte buffer for the

header of the next block to be written by the twiggy driver (for block 0 only). This buffer is initialized to all A's.

3) When a fatal error is encountered, the process filename and process number are displayed along with the error number. The user has the option to type <space> to continue (ie. back to prompt) or type <esc> to debug.

4) The phantom <cr> problem has been fixed. Some programs left the EOLN flag set, which would cause the next program to appear to see a <cr> before the user had a chance to type any input.

5) The M command now works for logical files.

6) The cursor on the primary screen is a black rectangle. The cursor on the alternate screen (except for lisabug) is a gray rectangle.

Multiple Programming

The Monitor now allows programs and exec files to execute in the Foreground or the Background. These are defined as follows:

Foreground	Background
Sup state Domain 0	User state Domain 1,2,3
X command (from Monitor)	B command (or X from Shell)
Block I/O is Atomic	~~Block I/O is not Atomic~~
Old Physical JT above A5	~~Cannot execute Old Physical~~
New Physical JT above A5	~~Cannot execute New Physical~~
Logical or IU JT above A5	Logical or IU JT above A5
Same screen as Monitor prompt	Defaults to the Alternate Screen

Note: ACDEFGLSU behave the same as the X command.

There are two new commands on the Monitor alternate prompt line. These are B for background and P for Process Mgr.

The user can launch background processes via the B command. The B command prompts the user for three pieces of information as follows:

1) Name of the file to execute (must be logically linked)
2) Input file name.
3) Output file name.

The loader will open the predefined files INPUT & OUTPUT with the file names given by the user except as follows:

If the user types <cr>, CONSOLE: is used as the file name and the I/O is directed to the alternate screen.

If the user types <esc>, CONSOLE: is used as the file name and the I/O is directed to the primary screen.

The number of processes is currently restricted to 4 in addition to the Monitor. Up to three processes can be

launched in the background prior to running a foreground process. Foreground proccesses are launched with the X command or an abbreviation of the X command (ie. A,C,E ...).

The scheduling is round robin on the first call to the Monitor or the Drivers (ie. Trap #5 and #$E) after 100 ms has expired since the last scheduling took place. Most calls to the Monitor are atomic. Exceptions to this include block I/O to any file and unit I/O to units 1 and 2.

The ProcMgr currently consists of only three commands:

1) F(lush

   Allows the user to flush all remembered code. Normally, this should not be needed.

2) L(ist

   Lists the active processes.

3) Q(uit

   Returns to Monitor prompt line.

Note: The debuggers RM command returns to Monitor and kills the current user process. If RM causes a trace display rather then returning to the Monitor then you should type G <cr> and try NMI RM again (or use RB for Reboot).

Launching Exec Files in the Background

To launch an exec file in the Background execute the Shell program with the B command and enter the name of your exec file in response to the Input prompt and Type <cr> to the output prompt.

Note: If you wish to terminate the Shell upon completion of the exec file you must include a Q for Q(uit to exit the shell.

Note: Someday the above will be available as B execfilename.

Error Summary

Loader Errors

0 Unknown segment
1 No room in memory
2 Bad block
3 Can't read code file
4 Jump Table Overflow
5 N.A.
6 N.A.
7 Too many units
8 Bad unit number
9 No Intrinsic.LIB file

```
 10  No unit location table
 11  No segment location table
 12  Can't open intrinsic library file
 13  Can't read file names block
 14  Bad segment number
 15  No unpack translation table
 16  Bad unpack version or unpack failed
 17  Can't load physical in the background
 18  Can't make process out of memory or domains
 19  Can't open Input or Output file
*20  No IULoader
*21  Too many object files
*22  Too many programs
*23  Too many code segments
*24  IULoader bug
*25  Squeeze failed
*26  CALLS1P1 returned
*27  Resolve failed
*28  GetAHeap failed
 29  Size Unpacked > 32767
*30  GetAStack failed
```

Note: Errors marked with * should not happen

Fatal Errors

```
 0  Illegal index to Trap handler
 1  Stack overflow
 2  Program halt
 3  Range value error
 4  Illegal string index
 5  Can't read Root volume
 6  Can't grow the heap (ie. out of memory)
 7  Can't get space for directory
```

DynStack, MaxStack, MinHeap, MaxHeap

The fields Maxstack and Maxheap are not enforced by the Monitors memory manager. Maxstack is currently limited to 128K. MaxHeap is limited by available physical memory. Approximately 826K is available for code and data. If a program needs more or less stack or heap than the linker defaults provide (128K each) use the SetSizes utility (see below).

New Utilities

ChangeMem

Can be used to modify the memory volume:

```
1) create #4:                    Type new size
2) grow the size of #4:          Type new size
3) flush #4:                     Type -1
```

Notes:

1) Shrink is currently not implemented. Use Flush.
2) Memory volume created by ChangeMem is limited to 128K bytes (ie. 256 blocks).

PrintSizes

Allows the user to see the current settings for the fields: DynStack, MaxStack, MinHeap and MaxHeap.

SetSizes

Allows the user to adjust the stack and heap sizes (ie. DynStack, MaxStack, MinHeap and MaxHeap) in the Executable block.

Shell

Monitor shell look alike which can be run in the background to launch other programs in the background.

Compiler, Assembler, Linker, Editor, ... Enhancements

1) The Assembler has been modified to use ObjIOLib for output. This has removed the intermittant bugs for large programs caused by reading and writing .CODE file formats.

2) The Assembler no longer uses any temporary files for patching forward references. The internal code buffer has been opened up to 32 K.

3) A .SEG directive has been added to specify the segment name. With this directive, assembler language modules can be placed to different segments without using the ChangeSeg Utility.

   Example:

```
.SEG    "FooBar"
.PROC   XXX
```

   This will place the PROC XXX (and all succeeding PROCs and FUNCs) in the segment named FooBar.

4) The ChangeSeg utility can now be used to change the segmentation of units without recompiling. The previous constraint (it only worked on assembly files) was relaxed.

5) A PrettyList option has been added for back substituting forward references in the assembly language listing. This replaces the PrettyList utility program.

6) The compiler and code generator now support Classes. If you use Classes or units which contain Classes, then you must link with the file 'CALLS.OBJ'. There are some new compiler errors relating to classes. See the Classcal reference manual for further details.

7) The compiler no longer prompts for the debugger output file.

8) PackSeg was enhanced to support both the O.S. packing strategy (one per system) and the Monitor's packing strategy (one per file). It now asks whether you want to read the pack table (or compute it). And whether you want to write the pack table into each file (or not). PackSeg will then accept a list of files to pack, terminated with a <cr>.

9) The Compiler and IULinker now accept alternate specifications for which Intrinsic.Lib to read as data. These options are invoked by typing +W to the Input prompt of the IULinker or ? to the Input prompt of the Compiler. This is useful when bootstrapping to architectures which are not upwards compatible.

10) The Compiler and Code generator were modified to add "Father" information in the debug information (*$D+ *) so that LisaBug could handle the many duplicate names generated by Classes.

11) The editor has been enhanced significantly as follows:

    a) UNDO and COMMAND-X, COMMAND-C, and COMMAND-V.
    b) Printing is now available.
    c) Grows the heap as needed.
    d) Duplicate is now available.

Debugger Enhancements

a) The display memory commands (ie. DM,DW and DL) have been modified to use word moves. The display bytes command (ie. DB) still uses byte moves and allows use of odd addresses.

b) The set memory commands (SW and SL) have been modified to use word moves. The set memory and set bytes commands (ie. SM and SB) still use byte moves and allow use of odd addresses.

   Note: For both display and set memory, if a word operation is applied to an odd the address then the address is rounded down to the nearest word address.

c) There is a new command CS (clear screen).

d) The process number is now displayed.

e) Break points can be of the form processnumber:address.

f) There are five new commands for the printer.

   PR expr Printer

   The PR command enables or disables printing to the two port card.

   expr = 2 enable printing upper port
   expr = 1 enable printing lower port

expr = 0 disable printing

Note: while printing is enabled all debugger output to the screen is echoed to the printer.

PS expr Print Screen

The PS command dumps the entire upper or lower screen.

expr = 1 print primary screen
expr = 0 print alternate screen

Note: If printing is disabled then nothing will happen.

FF Form Feed

The FF command sends a form feed to the printer if printing is enabled.

PL Print bug report via lower port
PU Print bug report via upper port

The PL and PU commands dump a "bug report" to the printer. The bug report consists of the following:

Dump of the primary screen.
Dump of the alternate screen (OS only).
Form Feed.
Description of exception.
Trace display.
Stack crawl.
Disassemble of 20 lines from PC-$20.
Display words from RA6-$20 for $80 bytes.
Form Feed.

Note: after a PL or PU command, printing is left disabled.

g) There are two commands which allow the user to dump all of memory to the upper or lower Twiggy as follows:

ML Memory dump to the lower Twiggy drive.
MU Memory dump to the upper Twiggy drive.

The ML and MU commands behave as follows:

1) If there is a disk in the drive when the command is issued the disk ejected.

2) The user is prompted to insert a disk in the appropiate drive. For example, the ML command will cause the prompt:

Please Insert Disk in Lower Drive.

3) After the disk is inserted the command proceeds as follows:

a) Clamp the disk.

b) Format the disk
c) Bad block scan the disk
d) Write block zero with debugger label
e) Dump the MMUs for all domains
f) Dump the first 860K bytes of memory.
g) Close the files.
h) Eject the disk.

Note: If everything goes ok the debugger prints the message:

Memory dump complete.

Note: If the process fails at some point the user will get an obscure message something like:

Floppy Disk Error XX

Note: The entire process takes about 3 1/2 minutes.

h) The commands for clamping and unclamping the disk were removed many months back when the world of Twiggy and ROMs was changing. So we now have the following commands once again:

DU expr — Disk unclamp. This will issue an unclamp to the drive. Use 1 for the upper drive and 2 for the lower drive.

DC expr — Disk clamp. This will issue a clamp to the drive. Use 1 for the upper drive and 2 for the lower drive.

i) Two commands have been added to allow a memory dump to be retrieved. The new commands are as follows:

RL Retrieve memory dump from lower Twiggy.
RU Retrieve memory dump from upper Twiggy.

Theses commands are intended to be used as follows:

1) Enter Lisabug and insert the disk into either drive. Issue the appropiate command RU or RL.

2) The debugger first attempts to validate the disk. If everything looks ok (at first glance) then the debugger will reload memory and MMU registers from the disk.

3) From here you can look around all you would like.

4) I would not recommend Trace, Go or RM unless you are very clever.

Note: When the MMUs and memory are reloaded from disk the following is left untouched.

1) In domain zero the first 16 MMUs and the last 3 MMUs (ie. 125,126,127) are not loaded.

2) Memory from 0 to $1800 is not loaded from disk.

3) Memory above $D3800 is not touched.

Note: The disk built by the memory dump will look as follows:

```
MEMDUMP:
MMU.DATA              4 22-Feb-83     6   512  Datafile
MEMORY.DATA        1692 22-Feb-83    10   512  Datafile
```

These files can also be looked at with Dumphex if you like. Memory is a straight image for now (ie. no packing yet). The MMUs are dumped domain 0 segment 0 first and domain 3 segment 127 last with 2 words per segment. The first word contains the origin and the second word contains the control and limit registers.

Memory Volume

The memory volume is currently not created at boot time. After booting, a memory volume can be created with the ChangeMem utility. The option still exists to have a memory volume created at boot time. To create a memory volume at boot time patch the desired size in blocks into the longint located at $50 in the file CONFIG.DATA. Memory volumes created at boot time provide a space which is locked down. The memory volume created at boot time is not limited by the 128k restriction which applies to the memory volume created by ChangeMem.

Bugs Fixed

The bug fixes for the Monitor, boot and system files are as follows:

1) The printer (on port &3) is no longer trashed by the changing of screen contrast.

2) During boot the Monitor pulls on keyboard reset so that the drivers get the correct keyboard id.

3) Loader error #5 no longer appears. The loader now relocates the jump table for physically linked programs. This allows the user to move the default stack pointer (within reason) and still run the physically linked program.

4) The DLE expansion problem has been fixed for the printer when attached to the parallel port.

5) The Monitor no longer crashes after the user program encounters fatal errors (ie. range check, ...). Fatal errors also stop exec files now.

6) Boot disks can now be copied with volume to volume transfer (ie. boot blocks remain intact).

7) First character after an exec file stopped was printed twice for some exec files. This has been fixed.

8) Exec files are stopped if EOF is encountered.

Cautions

1) The program SETSP should not be used with this release.

2) Programs which are to be launched in the background must be linked logical.

3) While any other processes are accessing a given volume do not attempt the following:

   a) Krunch the volume
   b) Zero the directory for that volume
   c) Manage that volume with the vMgr
   d) Format it
   e) Unmount the device the volume is on

4) Older versions of the system utilities should not be used with this release.

Notes:

1) When a process terminates it will cause the Monitors prompt line logic to restart if there is no foreground process active. This means that exec files should not be run in both the foreground and the background at the same time. Hopefully this bug will be removed by next release.