Inter-Office Memorandum

To Mesa Users

Date October 19, 1977

From Barbara Koalkin, Jim Sandman

Location Palo Alto

Subject Debugger - Experimental Features

Organization SDD/SD

XEROX

Filed on: [MAXC]<MESA-DOC>XDEBUGFEATURES.BRAVO

There are three experimental features available with the Mesa 3.0 Debugger. Two of the features are hooks to provide extended capabilities, and the third is a more capable Window Manager. One of the debugger commands is used to provide access to FTP to allow users to retrieve files from remote locations from within the debugger. The other new debugger command allows the user to invoke a special debugging package. The Window Manager has been expanded in an attempt to improve the debugger interface. Each feature will be described below in further detail. We encourage feedback on these features and remind you that due to their experimental nature, they are subject to change.

Debugger FTP

The Mesa 3.0 debugger now has the capability to invoke a subset of the standard FTP commands from within the debugger environment without having to exit out to the Alto Exec. ↑F enables the following commands:

Open *host, directory* -- opens a connection to the FTP Server in the specified *host* and (optional) specified *directory.*

Close connection -- closes the currently open FTP connection.

Retrieve *filename* -- will transfer *filename* from the remote host to the local host. The filename must conform to the file naming conventions on the remote host. You may designate multiple files by the use of "*" expansion if the remote server supports them (currently Maxc and IFS do). Note that the byte count is printed out truncated to 16 bits.

Delete *filename* -- will try to delete *filename* from the local host, regardless of whether the file is in use. If it finds it impossible to delete the file due to *some of its own pointers that would be left dangling,* it will not allow you to do so (e.g., you cannot delete XDebug.Image). *Beware of your own references!*

Free pages -- will tell you how many free pages are left on your disk.

List remote file designator *designator* -- lists all files in the remote host corresponding to *designator.* This must conform to the file naming conventions on the remote host, and you may designate multiple files by the use of "*" expansion.

Quit [*confirm*] -- takes you out of FTP mode and returns you to the debugger command processor. The debugger will close your connection when you leave FTP,

if you have forgotten to do so.

In order for you to be able to use the FTP commands you must retrieve the file **<Mesa>Fetch.Bcd** and load it into your debugger. This may be done when you are installing your debugger, in the same way as how you install the Window Manager. Once you are inside the internal debugger, type **New - Fetch**, which will load the FTP package before you install (**control-N**). You may later invoke FTP at any time by typing **control-F** to the external debugger. If you wish to load the FTP package at some later time, simply enter the internal debugger and then load **Fetch**. If you do not have the FTP package loaded, you may still use the **Delete**, **Free pages**, and **Quit** commands. However trying to use any of the other FTP commands will give you the message "-- FTP not installed".

Debugger User Commands

The new **↑UserProc (control-U)** command allows you to load your own debugging package into the debugger and invoke it at any time simply by typing **control-U**. The mechanism for loading is the same as for loading the FTP package. Simply enter the internal debugger either by typing **Mesa XDebug** to the Alto Exec or **control-D** to the external debugger. Then do a **New YourOwnFileName**, followed by an (optional) **STart**. Your program must EXPORT DebugUtilityDefs and provide an implementation for the procedure called **DebugUtilityDefs.CallUserProc.**

Internally, things work as follows: When the **↑UserProc** command is invoked the debugger looks to see if there has been a procedure loaded which has the name **CallUserProc[]**. If so, the debugger will invoke that procedure, which presumably lives in your user module, and may do anything you wish. If it can't find a procedure by that name, you will just get the message, "-- Not loaded".

The debugger is now willing to give you added help in gaining more access to information it already knows about your programs. Taking advantage of the new configuration format for organizing modules, XDebug.Config EXPORTS all of the debugger's interfaces (DebugBreakptDefs, DebugConfigDefs, DebugFtpDefs, DebuggerDefs, DebugInterpretDefs, DebugMiscDefs, and DebugUtilityDefs). A user program can get access to any of the debugger's PUBLIC procedures simply by importing the definitions modules of the procedures that you need. When writing your own user debugging routines, you should look carefully at some of the utility routines that the debugger already provides (eg., **ModuleNameToFrame**, **FrameToModuleName**, **READ**, etc.).

New Window Manager

The new Window Manager has the ability to handle long files (over 64K characters), and optionally uses the keyset as an input device. Several bugs have been fixed and performance has been improved. When the Window Manager is actively working on a command, the cursor is in the shape of hourglass. When it is done with the current task, the cursor returns to the normal shape.

The concept of "the current window" is more visible in this version of the Window Manager. In the released version, simply moving into another window would cause it to become the current window. While this feature made it easy to find windows that were lost, it was annoying to have windows repainted every time you moved out of one. In the current Window Manager, a window is current until the cursor is moved to another window and a mouse button is clicked. As a result, a window is not repainted until it is made current. For example if you load a window which is not current, the new contents will not appear until window is made current.

Selections are made by holding either the Red or Yellow mouse button while not in the scroll bar. Red selects and extends selections of characters. Yellow selects and extends selections of words. Characters typed into a scratch window are automatically selected as they are typed.

Positioning Commands

The positioning commands are all activated by moving the cursor into the scroll bar and clicking some combination of mouse buttons. The commands are:

Scroll Up[Red] -- scroll the line next to the cursor to the top of the window.

Thumb[Yellow] -- scroll to the position in the file corresponding to the relative position of the cursor in the scroll bar.

Scroll Down[Blue] -- scroll the top line of the window down so that it is next to the cursor.

Normalize[RedBlue] -- position the line containing the current selection next to the cursor.

The thermometer in the scroll bar shows the current position of the file in the window.

Menu Commands

The menu will appear when the blue mouse button is pressed and the mouse is not in a scroll bar. A menu command is selected by pointing at it, causing it to turn black, and releasing the mouse button. Except as noted, clicking the red mouse button after moving the mouse appropriately will cause the command to execute. Clicking blue will restore the previous state. The menu commands are as follows:

Create -- move the redbutton cursor to the place for the new window and click red. A scratch window will appear that will accept keyboard input. A maximum of four scratch windows may exist at any one time.

Load -- using the selection of the current window as a filename, load that file into the window selected by moving the redbutton cursor into a window and clicking red. The window containing Debug.Typescript cannot be loaded by the user.

Move -- the upper left-hand corner of the current window will stick to the redbutton cursor as it is moved around. Clicking red will position the window at that spot, while clicking blue will return the window to its former position.

Grow -- the lower right-hand corner of the current window will stick to the redbutton cursor as it is moved around, and the window will turn gray. Moving the cursor will change the size of the window subject to the minimum size restriction. Clicking red will fix the size of the window, while clicking blue will return the window to its former size.

Destroy -- moving the bullseye cursor into a window and clicking red will destroy it, while clicking blue will terminate the command. However, windows belonging to the debugger cannot be destroyed.

Stuff It -- take the selection of the current window and stuff it into the input stream of the window selected by clicking red.

Find -- find the current selection of the current window in the window selected by clicking red. The search begins at the end of the selection of the window being searched. If the search is successful, the text becomes the new selection and is scrolled to the top of the window.

Keys On/Off -- activate/deactivate input from the keyset as described below.

If after seeing the menu you do not wish to execute a menu command, simply move the cursor away from the menu and release the blue mouse button.

Keyset

The significant addition to the Window Manager is the use of the keyset for command and text input. Keyset chords will be described as octal numbers with 1B corresponding to the rightmost key and 20B the leftmost. The keyset forms a chord by ORing all keys depressed and returns the chord when all keys are released. Therefore, caution must be used not to have any books on the keyset when it is active.

Keyset Commands

The keyset commands involve stuffing characters and selections into the default window, which is the Debug.Typescript window when the Window Manager is loaded on top of the debugger. In each case, at the end of the command the default window is made the current window. The keyset commands are as follows:

1B -- Stuff the selection of the current window into the default window.

2B -- Stuff a CR into the default window.

3B -- Stuff the selection of the current window and a CR into the default window.

4B -- Stuff an ESC into the default window.

10B -- Stuff a DEL into the default window.

20B -- Stuff a BS into the default window.

Keyset Input

The keyset can also be used to input text. Holding down RedYellow on the mouse puts the keyset into text input mode. For each chord typed, the corresponding character is put into the input stream of the current window. The characters supported are 'A..'Z (1B..32B), '+ (33B), CtrlA (37B). The last page of this memo contains a label which may be used as a guide to the appropriate codes.

(The Stuff It keyset command is also implemented using the lower left function key on Alto II keyboards only. The current selection is stuffed into the default window.)

Debugger - Experimental Features

A	.	.	.	.	x	J	.	x	.	x	.	S	x	.	.	x	x
B	.	.	.	x	.	K	.	x	.	x	x	T	x	.	x	.	.
C	.	.	.	x	x	L	.	x	x	.	.	U	x	.	x	.	x
D	.	.	x	.	.	M	.	x	x	.	x	V	x	.	x	x	.
E	.	.	x	.	x	N	.	x	x	x	.	W	x	.	x	x	x
F	.	.	x	x	.	O	.	x	x	x	x	X	x	x	.	.	.
G	.	.	x	x	x	P	x	.	.	.	.	Y	x	x	.	.	x
H	.	x	.	.	.	Q	x	.	.	.	x	Z	x	x	.	x	.
I	.	x	.	.	x	R	x	.	.	x	.	+	x	x	.	x	x

BS DEL ESC CR STL