This message describes changes between the current Alto-compatible Dolphin microcode release (version 46) and the previous release (version 33). This microcode is sometimes referred to as "AMesa" below. As usual the various files involved in this release are version coordinated--all have version number 46 on the three Ivy directories where I put them. For this reason you should note the version to avoid possible confusion later.
Files related to this release are as follows:

<D0>InitialAlto.Eb!46	install on your disk with Othello
<D0>Alto.Eb!46	for Ethernet boot servers
<D0Source>AMesaSources.Dm!46	sources
<D0Docs>MPCodes.Press!46	interpretation of AMesa maintenance panel codes
<D0Docs>MurrayMPCodes.Press	interpretation of Initial and other maintenance panel codes
<D0Docs>AMesaRelease.Press!46	copy of this message

Version 46 fixes all operational bugs known to me, so report any problems that you have with it. If version 46 doesn't work for some reason, version 28/29 is stored on [Ivy]<D0>NewInitialAltoCSL.Eb!28 and on <D0>LFKB>NewInitialAltoLF.Eb!29 (NOTE: version 33 fixed several problems in version 28/29 but also introduced two bugs, so it seemed on the whole less useful that version 28/29; hence drop back to version 28/29 if 46 doesn't work).

Changes are as follows:

1) Some changes have been made in maintenance panel codes. At startup a microprogram called Initial runs first; it has maintenance panel codes documented in MurrayMPCodes.Press. The MP code 104 is displayed for about 0.4 seconds when Initial is finished and AMesa has commenced; after that MPCodes are interpreted as discussed in [Ivy]<D0Docs>MPCodes.Press!46.

2) The initialization problem which caused the display horizontal control to be screwed up just after you powered on your Dolphin is believed fixed.

3) A problem in version 33 or later Ethernet control microcode which caused occasional MP code 120 to 122 failures is fixed. A problem in version 33 or later disk control microcode is believed fixed.

4) The above microcode release will run on any of the three display/keyboard configurations--you do not need separate microcode versions for LF and CSL keyboards as you did before.

5) Storage is now preserved after a maintenance panel code crash, so it is possible to attach to a D0 with Midas and look at storage after a crash (I doubt that this will be useful except during extraordinary circumstances).

6) Microcode for color display, jasmine scanner, halftoning, and Mesa floating point opcodes is now part of the automatic loadup; those who have been using MicrocodeOverlay.Bcd with version 28 or 33 should delete it. Microcode developers note that the automatic loadup overwrites the area of microstore formerly reserved for the Midas Kernel. If this interferes with someone's debugging, call me.

7) WCBL (76b), ICBL (77b), and Misc opcodes with alpha byte in the range 60b through 77b now trap through SD 151, 152, and 156, respectively; these are for Cedar.

8) Mesa JRAM opcode (367b) now treats the microcode to which it jumps as a subroutine, which can optionally "Return" to continue with the next opcode; formerly it was necessary to exit at P7TailLoc, sometimes inconvenient. Also, you can now use JRAM to start an arbitrary microcode task--simply push a 16d-bit number on the stack in which bits 0..3 are the task and bits 4..15 are the starting microstore address; when the task thus started blocks, the emulator will continue at the next opcode (The LRJ opcode also allows arbitrary tasks to be started this way).

9) Two bugs in RXLPL and WXLPL and one page fault bug in RDBL Mesa opcodes are fixed courtesy of SDD (Frandeen or Johnsson, I think).

10) BitBlt handling of gray mode has been modified so that it is compatible with the Alto. Previous microcode would occasionally fail to align gray patterns in adjoining rectangular areas.

11) Some improvements in the display microcode result in about 1% performance improvement for all emulators and improve the worst case timing enough that display glitching with LF monitors should occur less often.

12) The refresh timer has been slowed from a 64 microsecond period to 256 microseconds; this has been done to reduce timer task overhead from about 4% of all cycles down to 1% of all cycles (i.e., emulators will run about 3% faster with the display off or about 4% faster with full screen display). Although the internal clock will tick more slowly, long-term "drift" with respect to Alto compatible clock update will be lessened, so the clock will run at about the same speed as on the Alto rather than about 0.4% faster. I know of no programs which rely upon the granularity of the 64 microsecond tick used previously, so this change should not affect any software.

Unfortunately, the period over which all bits in storage are refreshed is slowed from 2 msec to 8 msec. 16K ic’s used for storage specify that at the maximum operating temperature (80 degrees C. at the surface of the ic) a period of 2 msec is required for refresh. I have been advised that the refresh period is exponential in absolute temperature such that the required period halves about every 5 degree rise in temperature, so quadrupling the period will lessen maximum operating temperature of a marginal part about 10 degrees C. However, the spec is believed to be so conservative that very few parts require refresh at the maximum rate, and a 3% to 4% speed improvement seems worth the possible replacement of a few storage ic’s.

If anyone runs into problems with this change or has reliable hardware information, please let me know details. Note that it is desirable for storage diagnostics to use an equal or slower refresh period to detect possible problems in this area.

Thanks to Hal Murray for help in checking out this microcode.