XEROX PARC MEMORANDUM

TO: ALTO LAND and in particular, possessors of *1K CONTROL RAMS*

FROM: David Boggs

SUBJECT: Using the Control Ram

DATE: April 17, 1974

This memo is a description of a program called *RAMLOAD*, which runs under the ALTO operating system, and does just what it says, as well as a few other useful things which were easy once it was possible to load the ram. *RAMLOAD* was written to facilitate debugging EARSCODE - Ethernet Alto Research character generator Slot microCODE, which will not all fit in the 512 word debugger ram. With this program, tasks with working microcode can be run in the 1K control ram, freeing the entire 512 words of debugger ram for the task whose microcode is being debugged.

Since there are now two types of ram for the ALTO, some distinction must be made. Hereafer, ROM means some combination of roms on the ALTO control board, and add-on goodies which hang on end of the control board like debuggers with 512 words of ram. RAM means the extra board with 1K of ram which plugs into a slot in the processor.

RAMLOAD gets its parameters from the command line and default values. If you do not specify a parameter, the default is used. In addition there are some global switches which do other useful things as explained below:

GLOBAL SWITCHES (of the form RAMLOAD/switchlist)

/R compare the micro binary file against the contents of the RAM and display differences.
/V compare the micro binary file against the contents of the ROM and display differences.
/C compare the micro binary file against the contents of the constant memory and display differences.

LOCAL SWITCHES (of the form bla/switch)

/F use bla as the name of the micro binary file. Default is "BINFILE."
/M use bla as the name of the instruction memory in the micro binary file. Default is "INSTRUCTION".
/C use bla as the name of the constant memory in the micro binary file. Default is "CONSTANT".
/V bla is an octal number. Use it as the boot locus vector. Bit 15 corresponds to task 0 (emulator). 0 means run task in the RAM. Default is #177777 - keep all tasks in ROM.
/A bla is an octal number, representing the base address of a 5 word area in the RAM which RAMLOAD can use for utility purposes. Default is the top 5 words (#1773). See warnings below about restrictions for specific operations.
/S bla is an octal number interpreted as the beginning address of the emulator main loop (START for microcode hackers). Default is the current START address, #335.

Note that global switches /V and /C do the same things that ;V and ;C do in DEBAL. *RAMLOAD* in effect does a ;L, and also sets the boot locus vector. The /R global switch was added because it was easy and people might want to see if the microcode got smashed after a fiasco.

When *RAMLOAD* is called, it will first display what it thinks it is supposed to do as governed by the switches and defaults, and wait for a confirming carriage return. Once this is received, if any global switches were set, it will do the appropriate test, report the results, and then wait for a confirmation that it should load. If no global switches were set, it will assume you wanted to load, and do it. Loading first sets the boot locus vector, and then loads the RAM so that you can load even the 5 word utility area. When complete it will report the number of instructions loaded, and the highest address ala DEBAL. Next the program will ask if you want to boot (thus moving the tasks specified in the boot locus vector into the newly loaded microcode in the RAM). If you confirm, and if you have an *ETHERNET* board, the machine will do a software initiated boot. If you do not have an *ETHERNET* the boot will be a NOP, and a FINISH is executed. Hitting the boot button after the program is finished will work for those hermits who do not have *ETHERNETS*.

The routine which reads the micro binary file expects the limited subset of block-types that DEBAL puts out. If it encounters an unusual block-type (3, 5, or 6) , it will endeavor to do the right thing, and continue on. When it is finished reading, if any unusual types were encountered, it will list how many of each it read. If the microcode was assembled using DEBAL, this is cause for grave doubts about the correctness of the file, since DEBAL will not currently generate these types.

Where the 5 word utility area is specified can have profound (ie. potentially disasterous) effects on the machine's operation if you are currently running from the RAM. While it is possible to load into the microcode you are currently executing form, this is living very dangerously. However, if you must, observe the following caveats:

* If constant memory is being checked, and you are executing out of the low 256 locations, you are dead.

* the 5 word utility area must be specified in a place you will not be executing from during the *RAMLOAD* program. *RAMLOAD* always saves any word in RAM it modifies for utility purposes, and restores it when it is done, but while in use, it can have an arbitrary value.

A number of things can cause fatal errors during execution. If one happens, an error message is written in the system display area, and the program is aborted.

I will be happy to help anyone else who needs to program the RAM.

FINAL

```
;SYMBOL DEFINITIONS FOR ALTO

$STARTF $L16017,0,0;    NDF1=17
$MOUSE  $L0,14006,100;
$DISP$L0,14007,120;
$MD$L26006,14005,124100;
$DDR$L26010,0,124100;
$XPREG  $L26010,0,124000;
$CSR    $L26011,0,124000;
$TASK$L16002,0,0;
$BLOCK$L16003,0,0;
$MAR$L20001,0,144000;
$LLCY8$L0,22006,200;
$LRSH1$L0,22005,200;
$LLSH1$L0,22004,200;
$BUS=0$L24001,0,0;
$SH<0$L24002,0,0;
$SH=0$L24003,0,0;
$BUS$L24004,0,0;
$ALUCY$L24005,0,0;
$IDISP$L24015,0,0;
$BUSODD $L24010,0,0;
$LMRSH1 $L0,62005,200;  MAGIC RIGHT SHIFT
$LMLSH1 $L0,62004,200;  MAGIC LEFT SHIFT
$EVENFIELD$L24010,0,0;
$SETMODE$L24011,0,0;
$IR$L26014,0,124000;
$ACDEST$L30013,32013,60100;
$DNS$L30012,0,60000;
$ACSOURCE$L0,32016,100;
$L$L40001,36001,144200;
$HALT$L42001,0,0;
$BREAK$L42003,0,0;
$WENB$L42005,0,0;
$READY?$L42006,0,0;
$NOVA$L44002,46003,124100;
$ORT$L0,50002,2;
$ANDT$L0,50003,2;
$XORT$L0,50004,2;
$+1$L0,50005,2;
$-1$L0,50006,2;
$+T$L0,50007,2;
$-T$L0,50010,2;
$-T-1$L0,50011,2;
$+INCT$L0,50012,2;      SYNONYM FOR +T+1
$+T+1$L0,50012,2;
$+SKIP$L0,50013,2;
$T$L52001,54001,124040;
$END$L34000,0,0;
$.T     $L0,50014,2;
$AND NOT T$L0,50015,2;



;DEFINITIONS FOR CONTROL RAM
$SWMODE $L16010,0,0;            NDF1=10  (EMULATOR)
$WRTRAM $L16011,0,0;            NDF1=11
$RDRAM  $L16012,0,0;            NDF1=12

;DISK DEFINITIONS
$KSTAT  $L20012, 14003, 124100 ;        DF1=12 (LHS) BS=3 (RHS)
$RWC    $L24011, 0, 0;                  NDF2=11
$RECNO  $L24012, 0, 0;                  NDF2=12
$INIT   $L24010, 0, 0;                  NDF2=10
$CLRSTAT$L16014, 0, 0;                  NDF1=14
$KCOMM  $L20015, 0, 124000;             DF1=15 (LHS ONLY) REQUIRES BUS DEF
$SWRNRDY$L24014, 0, 0;                  NDF2=14
$KADR   $L20016, 0, 124000;             DF1=16 (LHS ONLY) REQUIRES BUS DEF
$KDATA  $L20017, 14004, 124100;         DF1=17 (LHS)  BS=4 (RHS)
$STROBE $L16011, 0, 0;                  NDF1=11
$NFER   $L24015, 0, 0;                  NDF2=15
$STROBON$L24016, 0, 0;                  NDF2=16
$XFRDAT $L24013, 0, 0;                  NDF2=13
$INCRECNO$L16013, 0, 0;                 NDF1=13
$SINK   $L44000, 0, 124000;             DF3=0  FAKE TO ALLOW BUS SOURCE WITH
```