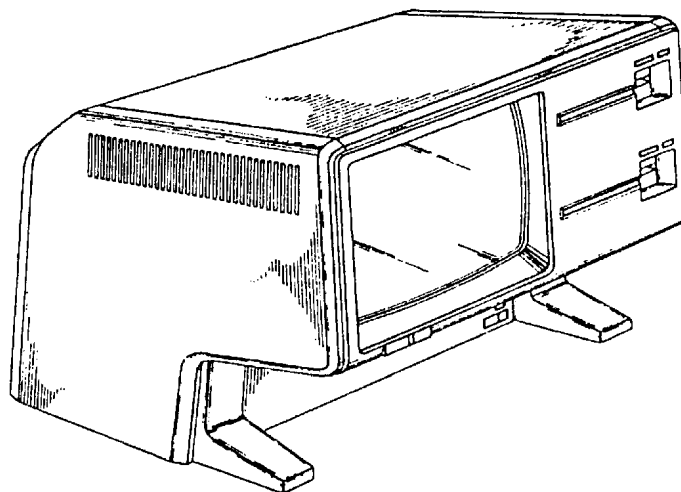


Lisa Boot ROM Listing



This is the listing of the Lisa's Boot ROM program, version 2.48 (a.k.a. version H). This program was written in 68000 assembly language. This listing was produced by the Lisa 68000 TLA Assembler. At the end of this document appears the hex bytes that this listing produces with some commentary.

```

0000|      ;      .NOLIST
0000|      ;
0000|      ;
0000|      ;      LL      II      SSSSSSS      AAA
0000|      ;      LL      II      SS      AA AA
0000|      ;      LL      II      SSSSSSS      AAAAAA
0000|      ;      LL      II      SS      AA      AA
0000|      ;      LLLLLL      II      SSSSSSS      AA      AA
0000|      ;
0000|      ;
0000|      ;      BBBB      OOOO      OOOO      TTTTTTT      RRRRR      OOOO      MM      MM
0000|      ;      BB  BB      OO  OO      OO  OO      TT      RR  RR      OO  OO      MM M M MM
0000|      ;      BBBB      OO  OO      OO  OO      TT      RRRRR      OO  OO      MM  M  MM
0000|      ;      BB  BB      OO  OO      OO  OO      TT      RR  RR      OO  OO      MM      MM
0000|      ;      BBBB      OOOO      OOOO      TT      RR  RR      OOOO      MM      MM
0000|      ;
0000|      ;
0000|      ;      Copyright 1983, 1984 Apple Computer Inc.
0000|      ;      Revision 2H
0000|      ;
0000|      ;
0000|      ;
0000|      ;      Filename:  RMOXX.Y.TEXT, XXX = ROM VERSION # (e.g., 200 for 2.00)
0000|      ;      Y = E (equates)
0000|      ;      = K (kernel tests)
0000|      ;      = S (secondary tests)
0000|      ;      = B (bootstrap code)
0000|      ;      = M (monitor code)
0000|      ;      = G (graphics, icon and message display)
0000|      ;
0000|      ;      Function:  Initializes LISA system for use and performs preliminary
0000|      ;      diagnostic checks.  If all tests pass, the system then
0000|      ;      does a keyboard scan to check for user input.  If any key
0000|      ;      is hit other than caps lock or the mouse button,
0000|      ;      a menu is displayed on the screen showing the available
0000|      ;      boot devices.  If a valid COMMAND key sequence is detected,
0000|      ;      a boot from an alternate device is attempted (see below).
0000|      ;      If no keyboard input is detected, the system first checks
0000|      ;      parameter memory for a valid boot device and, if none, defaults
0000|      ;      to booting from a Profile attached to the builtin parallel port
0000|      ;      for Lisa 1 systems.
0000|      ;
0000|      ;      For Lisa 2 systems, a check is first made to verify a disk
0000|      ;      (internal or external) is connected before defaulting to the
0000|      ;      hard disk boot.  If no disk is detected, the system defaults

```



```

0000|          ;          to booting from the floppy drive.
0000|          ;
0000|          ;
0000|          ;
0000|          ; Inputs:  Checks for keyboard input from the user.  Currently, the following
0000|          ;          key sequences are honored if input after the system "clicks" the
0000|          ;          speaker (CMD refers to the Apple key on the keyboard):
0000|          ;
0000|          ;          CMD/1 - boot from Twiggy drive #1 or integral hard disk
0000|          ;          CMD/2 - boot from Twiggy drive #2 or SONY drive
0000|          ;          CMD/3 - boot from Profile attached to parallel port or integral hard disk
0000|          ;          CMD/4 - boot from I/O slot #1, port 1
0000|          ;          CMD/5 - boot from I/O slot #1, port 2
0000|          ;          CMD/6 - boot from I/O slot #2, port 1
0000|          ;          CMD/7 - boot from I/O slot #2, port 2
0000|          ;          CMD/8 - boot from I/O slot #3, port 1
0000|          ;          CMD/9 - boot from I/O slot #3, port 2
0000|          ;          CMD/ENTER (on key pad) - abort boot, branch to ROM monitor
0000|          ;          CMD/SHIFT/P - abort boot, do power cycling
0000|          ;
0000|          ; OUTPUTS:  Saves various results and contents of system registers in memory
0000|          ;          for examination by system programs or with the ROM monitor.
0000|          ;
0000|          ;          $180-183 : Power-up status (x00000000 = ok)
0000|          ;          $184-185 : Memory sizing error results
0000|          ;          $186-1A5 : Results of memory read/write tests
0000|          ;          $1A6-1A9 : Parity error memory address (if error during mem test)
0000|          ;          $1AA-1AB : Memory error address latch
0000|          ;          $1AC-1AF : D7 save on exception errors
0000|          ;          $1B0-1B1 : Results of MMU tests (context/data bits)
0000|          ;          $1B2      : Keyboard ID (00 = no ID received)
0000|          ;          $1B3      : Boot device ID
0000|          ;          $1B4-1B9 : Boot failure data
0000|          ;          $1BA-1BF : Clock setting (Ey,dd,dh,hm,ms,st)
0000|          ;          $1C0-1DF : Data reg save area (D0 - D7)
0000|          ;          $1E0-1FF : Address reg save area (A0 - A7, A7 = USP)
0000|          ;          $240-260 : System serial #
0000|          ;          $260-267 : Scratch area
0000|          ;          $268-26B : Suspected (logical) memory error address for parity error
0000|          ;          $26C-26F : Save of data written to suspected error address
0000|          ;          $270-273 : Actual (logical) error address found during search
0000|          ;          $274-277 : Save of data read during parity error search
0000|          ;          $278-27B : (Physical) error address read from parity error address latch
0000|          ;          $27C      : Error row for parity chip failure (0 = first row, 7 = last row)
0000|          ;          $27D      : Error column for parity chip failure (9 or 14)
0000|          ;          $27E-280 : Reserved
0000|          ;          $280-293 : Exception data save area

```



```

0000| ; (FC/EXCADR/IR/SR/PC/EXCTYPE/SSP)
0000| ; 44 = NMI or other interrupt
0000| ; 45 = bus error
0000| ; 46 = address error
0000| ; 47 = other exception/interrupt
0000| ; 48 = illegal instruction error
0000| ; 49 = line 1010 or 1111 trap
0000| ; 50 = bus error accessing keyboard VIA
0000| ; 51 = bus error accessing parallel port VIA
0000| ; 57 = bus error accessing disk controller
0000| ; $294-297 : Maximum physical memory address + 1
0000| ; $298-299 : I/O slot 1 card id (0 = no card present)
0000| ; $29A-29B : I/O slot 2 card id
0000| ; $29C-29D : I/O slot 3 card id
0000| ; $29E : Reserved
0000| ; $29F : Reserved
0000| ; $2A0 : Reserved
0000| ; $2A1 : Disk ROM id
0000| ; $2A2-2A3 : Reserved
0000| ; $2A4-2A7 : Minimum physical address
0000| ; $2A8-2AB : Total memory (Max-Min)
0000| ; $2AC : SCC test results
0000| ; $2AD : Slot # of memory board if memory error
0000| ; $2AE : Result of disk controller self-test
0000| ; $2AF : System type (0 = Lisa 1, 1 = Lisa 2, 2 = Lisa 2 with external hard disk,
0000| ; 3 = Lisa 2 with internal hard disk)
0000| ; $2B0-2BF : Keyboard queue (16 bytes)
0000| ; $2C0-480 : ROM scratchpad/stack area
0000| ; $480-800 : Reserved for ROM local variable usage
0000| ;
0000| ; Also saves data in special parameter memory area reserved for boot ROM use if error
0000| ; encountered. Usage is as follows:
0000| ;
0000| ; $FCC161 : Error code
0000| ; $FCC163-165 : Contents of memory error address latch if parity error
0000| ; $FCC167 : Memory board slot # if memory error
0000| ; $FCC169-173 : Last value read from clock
0000| ; $FCC175-17B : Reserved
0000| ; $FCC17D-17F : Checksum
0000| ;
0000| ; Originator: Rich Castro 7/30/81 - Version 0.0 released to manufacturing
0000| ; Modified by: Rich Castro 7/30 - 11/3/81 - Made the following changes:
0000| ; 1) Twiggy bootstrap capability
0000| ; 2) Initial COPS test and keyboard scan
0000| ; 3) Moved parallel card to slot 2
0000| ; 4) Changed ROM interrupt/exception vectors,
0000| ; 5) Created jump table for ROM routines

```

```

0000|      ;
0000|      ;      11/3/81 - Version 0.7 released to the world
0000|      ;
0000|      ;      11/4/81 - 1/15/82 - Made the following changes:
0000|      ;      1) Added support for new memory cards
0000|      ;      2) Added warm-start capability and jump
0000|      ;      table for ROM subroutine usage
0000|      ;      3) Modified MMU reset routine to support
0000|      ;      single step board usage
0000|      ;      4) Added full memory initialization
0000|      ;      5) Added 256K memory parity test
0000|      ;      6) Modified COPS initialization so that
0000|      ;      keyboard commands can be sensed more
0000|      ;      reliably
0000|      ;      7) Added error code display routines and
0000|      ;      display of CPU and IO ROM versions
0000|      ;      8) Added preliminary disk controller test
0000|      ;      9) Updated warm-start check
0000|      ;      10) Modified disk interface test
0000|      ;      11) Changed low memory assignments
0000|      ;      12) Made corrections for no I/O board, disk
0000|      ;      interface error and contrast setting
0000|      ;      13) Modified memory sizing routine to
0000|      ;      catch memory errors
0000|      ;      14) Modify MMU test to avoid context 0
0000|      ;      destruction, add contrast setting for
0000|      ;      new machines, correct disk error and
0000|      ;      CPU ROM messages
0000|      ;      15) Move stack so old memory test still runs
0000|      ;
0000|      ;      1/15/82 - Release version 0.16
0000|      ;      1/18/82 - Fix stack problem and release vrsn 0.17
0000|      ;      1/19/82 - Change stack for call routine and version
0000|      ;      to 0.18
0000|      ;      1/27/82 - Change MMU error routine to do address
0000|      ;      and data line toggling
0000|      ;      1/28/82 - Add video circuitry test
0000|      ;      1/30/82 - Add write wrong parity test
0000|      ;      1/31/82 - Move run time stack to $180
0000|      ;      2/6/82 - Add Profile bootstrap with upgrade for
0000|      ;      OS use (add jump table entries also)
0000|      ;      2/15/82 - Update Twiggy bootstrap and add entry
0000|      ;      for OS use; also add MMU test to
0000|      ;      conditional assembly and add context
0000|      ;      saving to MMUTST2 routine
0000|      ;      2/17/82 - Add correction to memory test for
0000|      ;      reboot problem and leave parity on

```



0000	;	2/24/82	- Add code for clock test and special
0000	;		burn-in cycling
0000	;	2/25/82	- Add code to simulate soft on switch
0000	;		pressed for COPS problem
0000	;	3/1/82	- Removed all changes since ROM 0.18
0000	;		release except for parity enabling,
0000	;		no reset feature, memory sizing change
0000	;		and Profile booting
0000	;	3/1/82	- Restore default stack ptr loc to \$300
0000	;	3/1/82	- Move default stack to \$0400, restore
0000	;		everything except MMU testing
0000	;	3/4/82	- Add MMU initialization and modify
0000	;		Twiggy, Profile boot routines for new
0000	;		load point
0000	;	3/10/82	- Add change for new I/O addresses and
0000	;		fix for Twiggy routine
0000	;	3/10/82	- Change contrast value for new I/O's
0000	;	3/15/82	- Add correction for Profile and COPS
0000	;		routines and display msg when booting
0000	;	3/17/82	- Restore version # at end of file
0000	;	3/18/82	- Release version 0.22
0000	;		
0000	;	4/5/82	- Make initial 2732 version (1.00); add
0000	;		following changes:
0000	;		1) correct MMU error routine bug
0000	;		2) change stack for CALL to \$0400
0000	;		3) add parity disable to WWP routine
0000	;		4) change MMU I/O space code to '9'
0000	;		5) add invalid boot code message
0000	;	4/6/82	- Add speaker click after COPS check
0000	;	4/7/82	- Add jump table entry for speaker
0000	;		routine, 1 second delay before "click"
0000	;		and alpha lock key check
0000	;		
0000	;	4/8/82	- Release version 1.00
0000	;		
0000	;	5/5/82	- Add I/O slot configuration check and
0000	;		I/O slot booting. Also add change to
0000	;		Profile read routine for PCR setting.
0000	;	5/12/82	- Add burnin power-cycling routine as
0000	;		boot option invoked by CMD/P.
0000	;	5/13/82	- Add changes for COPS command timing,
0000	;		Twiggy timeout, Twiggy booting, and
0000	;		add power-cycling routine.
0000	;	5/14/82	- Add fixes for booting via parameter
0000	;		memory and COPS timing experiment.
0000	;	5/17/82	- Add display of loop count and run time,

```

0000|           ;           and alter parameter memory useage for
0000|           ;           power-cycling option.
0000|           ;           5/18/82 - Add display of Twiggy errors, change
0000|           ;           COPS routine for precheck code.
0000|           ;           5/20/82 - Add contrast reset for "warm start",
0000|           ;           add cycle value display, restore COPS
0000|           ;           timeout code.
0000|           ;
0000|           ;           5/21/82 - Release version 1.02
0000|           ;
0000|           ;           5/26/82 - Begin addition of ROM monitor; set
0000|           ;           default to Apple if PM = 00.
0000|           ;           6/1/82 - Make following changes:
0000|           ;           1) Memory sizing retry count to 64
0000|           ;           2) Save results on memory sizing errors
0000|           ;           3) Update NMI routine to check for parity
0000|           ;           errors.
0000|           ;           4) Restore default NMI vector after
0000|           ;           memory test.
0000|           ;           5) Create read clock subroutine and call
0000|           ;           when doing clock display.
0000|           ;           6) Add boot fix to save device id.
0000|           ;           6/1/82 - Change to new sizing algorithm and retry
0000|           ;           count back to 32.
0000|           ;           6/3/82 - Convert to version 1.03
0000|           ;           6/3/82 - Made following changes:
0000|           ;           1) Localize message display to TSTCHK
0000|           ;           2) Do clear screen only in INITVCT and
0000|           ;           in TSTCHK and second monitor level.
0000|           ;           3) Change default video page to last.
0000|           ;           4) Complete first edition of monitor.
0000|           ;           6/7/82 - Modify monitor level2 user interface.
0000|           ;           6/10/82 - Made following changes:
0000|           ;           1) Add boot from Apple as CMD/A.
0000|           ;           2) Clear screen and display only in
0000|           ;           routine TSTCHK.
0000|           ;           3) Add ROM checksum error bit.
0000|           ;           4) Add exception error check to TSTCHK.
0000|           ;           5) Add speaker click just before
0000|           ;           keyboard scan.
0000|           ;           6) Reset to first video page for boot
0000|           ;           from Apple.
0000|           ;           7) Merge in changes from 1.03 file.
0000|           ;           8) Add parity error check to TSTCHK
0000|           ;           9) Change power-cycling so that double
0000|           ;           bus fault used to restart diags
0000|           ;           6/11/82 - Made following changes:

```

```

0000|           ;           1)Increase Twiggy timeout to 2 mins.
0000|           ;           2)Add 5 sec delay in power-cycle mode
0000|           ;           before shutting down.
0000|           ;
0000|           ;           6/14/82 - Release version 1.04
0000|           ;           6/22/82 - Add loop after COPS test if error
0000|           ;           since keyboard not accessible. Also add
0000|           ;           fix for NMI restore after memory test.
0000|           ;           6/30/82 - Made following changes:
0000|           ;           1)Add parameter memory and I/O boot
0000|           ;           checksum routines.
0000|           ;           2)Remove boot id save to parameter
0000|           ;           memory, except for power-cycle.
0000|           ;           3)Change to new boot device id's.
0000|           ;           7/1/82 - 1)Add changes for new Twiggy firmware.
0000|           ;           2)Add fixes for bugs in 1.04:
0000|           ;           a)Add row setting before error display
0000|           ;           to avoid writing over menu line.
0000|           ;           b)Set device codes for Profile and
0000|           ;           I/O slots to allow display if error.
0000|           ;           c)Enable setting of timeout for Twiggy
0000|           ;           reads.
0000|           ;           d)Save error codes for I/O boot in
0000|           ;           memory.
0000|           ;           e)Add option of clearing memory in
0000|           ;           INITMON routine.
0000|           ;           7/7/82 - Made following changes:
0000|           ;           1)Modify checksum routines
0000|           ;           2)Add keyboard/mouse check/reset code
0000|           ;           7/13/82 - Add speed parameter for new Twiggy code
0000|           ;           7/14/82 - Add check for DSKDIAG in disk test,
0000|           ;           change to new Twiggy error codes
0000|           ;           7/15/82 - Made following changes:
0000|           ;           1)Add Profile routine updates.
0000|           ;           2)Restore old boot id codes - new ones
0000|           ;           used only when new Twiggy code
0000|           ;           released.
0000|           ;           3)Upgrade burnin code for new parameter
0000|           ;           memory usage.
0000|           ;           4)Attempt to enable keyboard after MMU
0000|           ;           errors.
0000|           ;           5)Remove I/O boot checksum code until
0000|           ;           conversion to new Twiggy code.
0000|           ;           6)Add video pattern display code..
0000|           ;           7)Remove characters from table and
0000|           ;           make other changes to save bytes.
0000|           ;           8)Upgrade service mode display option

```



```

0000|           ;           to handle count up to $FFFF.
0000|           ;
0000|           ;       7/16/82 - Create version 1.05
0000|           ;       7/19/82 - Add bug fixes for MMU testing, power-
0000|           ;           cycle memory testing, Profile boot
0000|           ;           and service mode display option.
0000|           ;
0000|           ;       7/19/82 - Create version 1.06
0000|           ;       7/20/82 - Add fix for MMU testing to properly
0000|           ;           record context in error
0000|           ;
0000|           ;       7/20/82 - Release version 1.07
0000|           ;       7/21/82 - Make keyboard/mouse reset code changes
0000|           ;           and move check to before first "click"
0000|           ;       7/23/82 - Add extended memory tests
0000|           ;       7/27/82 - Add screen memory test and VIA tests.
0000|           ;           Change default boot for new Twiggy code
0000|           ;           to upper Twiggy. Add conditionals for
0000|           ;           Apple code.
0000|           ;       7/29/82 - Add SCC test, optimize code.
0000|           ;       7/30/82 - Add RAM address uniqueness test.
0000|           ;       8/4/82  - Added the following:
0000|           ;           1)Twiggy mods for interleave
0000|           ;           2)Monitor options CONTINUE and LOOP
0000|           ;           3)Exception routine for line 1111 and
0000|           ;           line 1010 errors.
0000|           ;       8/9/82  - Add Twiggy mod for disk clamp, add mods
0000|           ;           for kernel test failures such as screen
0000|           ;           flash on MMU error.
0000|           ;       8/11/82 - Add memory sizing fix, increase delay
0000|           ;           for COPS and change default boot to
0000|           ;           TWIGGY!!
0000|           ;       8/12/82 - Begin code changes for new user interface
0000|           ;           and add hooks for icon display.
0000|           ;       8/14/82 - Add mods for Twiggy changes to monitor
0000|           ;           DSKDIAG line and add initial timeout.
0000|           ;           Continue user interface changes.
0000|           ;       8/18/82 - Add mouse, cursor code and changes for
0000|           ;       8/23/82 - Add controls for 2716 version of ROM.
0000|           ;           Add changes for Service mode to use
0000|           ;           pull down menu, eliminate keyboard
0000|           ;           queuing while awaiting input.
0000|           ;       8/24/82 - Add dialog box, and window to service
0000|           ;           mode with modified scroll and character
0000|           ;           output routines.
0000|           ;       8/25/82 - Add icons along with routines to display
0000|           ;           during test and for errors.

```

```

0000|      ;      8/27/82 - Add routines for displaying and using
0000|      ;      boot icon menu.
0000|      ;      8/30/82 - Add auto boot from Applenet.
0000|      ;      8/31/82 - Add minor additions to Service mode
0000|      ;      for Set and Loop options.
0000|      ;
0000|      ;      8/31/82 - Create and do internal release of
0000|      ;      2716 (0.24), 2732 (1.15) and 2764 (2.00)
0000|      ;      ROM versions.
0000|      ;
0000|      ;      9/8/82 - Add fixes for I/O slot icon display
0000|      ;      and Profile icon display.
0000|      ;      9/9/82 - Add fix for reboot problem in 2716 ROM.
0000|      ;      Add serial # read routine and test for
0000|      ;      2732 and 2764 ROM versions. Expand
0000|      ;      stack for serial read routine.
0000|      ;      9/10/82 - Add fix for device code display for ROM
0000|      ;      versions 0.24 and 1.15.
0000|      ;
0000|      ;      9/10/82 - Create and do internal release of new
0000|      ;      ROM versions 0.25, 1.16 and 2.01.
0000|      ;
0000|      ;      9/13/82 - Add fixes for memory sizing and I/O
0000|      ;      slot booting.
0000|      ;
0000|      ;      9/14/82 - Create and release ROM versions 0.26,
0000|      ;      1.17 and 2.02.
0000|      ;      9/22/82 - Add fixes and code for:
0000|      ;      1)Default video latch setting
0000|      ;      2)Mask for I/O and exception errors
0000|      ;      3)Message display on external calls
0000|      ;      to ROM monitor
0000|      ;      4)Contrast setting before screen test
0000|      ;      5)Disable of NMI key on power-up
0000|      ;      6)Boot failure after first load
0000|      ;      7>Error tones for failures
0000|      ;      8)Loop mode setting of NMI key
0000|      ;      9/23/82 - Add
0000|      ;      1)Power cycling
0000|      ;      2)Full service mode menu
0000|      ;      3)Loop mode test choice display
0000|      ;      9/24/82 - Add dump memory option to service mode
0000|      ;      9/25/82 - Modify display memory option to allow
0000|      ;      count and address data on same line
0000|      ;      9/29/82 - Add jump table entry for READMMU
0000|      ;      9/30/82 - Add:
0000|      ;      1)"No reset" feature

```



```

0000|           ;           2)Verify Disk option for service mode
0000|           ;           3)Optimize cursor routines and
0000|           ;           remove unused CursorShield routine.
0000|           ;           4)Invert rectangles when selected from
0000|           ;           keyboard.
0000|           ;           5)Display boot menu only if down keycode
0000|           ;           detected.
0000|           ;           10/5/82 - Add:
0000|           ;           1)Memory error decoding to board level
0000|           ;           2)New size and position for alert box
0000|           ;           3)New test icon display
0000|           ;           4)Diskette # for Twiggy errors
0000|           ;           10/6/82 - Add:
0000|           ;           1)Continue keyboard scan after COPS
0000|           ;           errors
0000|           ;           2)Set extended memory test bit for
0000|           ;           loop on memory test option
0000|           ;           3)Display I/O slot card # on errors
0000|           ;           4)Change boot menu to "pull-down" format
0000|           ;           5)Change to new icons
0000|           ;           10/7/82 - Add:
0000|           ;           1)SCC test
0000|           ;           2>Error if no serial # (allow continue)
0000|           ;           3)Two passes of memory tests for extended
0000|           ;           mode, one for regular mode
0000|           ;           10/9/82 - Create version 2.03
0000|           ;           10/10/82 - Add bug fixes and I/O slot ROM check in
0000|           ;           config scan.
0000|           ;           10/12/82 - Create and release version 2.04.
0000|           ;           10/13/82 - Make following changes:
0000|           ;           1)Add keyboard reset code
0000|           ;           2)Remove SCC test
0000|           ;           3)Add bug fixes for making alert box
0000|           ;           and displaying bad keyboard
0000|           ;           10/14/82 - Add display of check marks for test icons
0000|           ;           10/18/82 - Add fixes for Monitor entry, Profile boot,
0000|           ;           looping on diag tests
0000|           ;           10/20/82 - Add message translations
0000|           ;           10/21/82 - 1)Adjust alert box and button dimensions
0000|           ;           2)Add boot from all ports on I/O slots
0000|           ;           3)Add fix for CMD key detection in monitor
0000|           ;           4)Change powercycle window to alert box
0000|           ;           5)Extend verify timeout to 4 minutes
0000|           ;           10/22/82 - 1)Add keyboard reset on external entry to ROM
0000|           ;           monitor

```

```

0000|      ;      2)Make Dump Memory routine conditional on
0000|      ;      final LISA ROM
0000|      ;      10/25/82 - 1)Change wait for disk error to branch to
0000|      ;      monitor - CONTINUE option then continues
0000|      ;      with the same boot device
0000|      ;      2)Change RETRY phrase to RESTART
0000|      ;      10/27/82 - Made following changes:
0000|      ;      1)RESET instruction on startup
0000|      ;      2)Jump table entries for access to memory
0000|      ;      test and display decimal routines
0000|      ;      3)Optimize warm start reset check and
0000|      ;      MMU error loop routines
0000|      ;      4)Change default video page to $2F for
0000|      ;      no memory found.
0000|      ;      5)Rewrite screen memory test. Change main
0000|      ;      memory test to go from low memory
0000|      ;      to base of screen memory.
0000|      ;      6)Move inverse video check to after screen
0000|      ;      test, doing rewrite only of screen page.
0000|      ;      7)Add new boot failure code, with hooks to
0000|      ;      catch booting errors after ROM has
0000|      ;      released control to boot loader for Twiggy
0000|      ;      and Profile booting.
0000|      ;      10/29/82 - Add display for uncompressed slot card icons.
0000|      ;      Modify TONE routine to init PCR reg.
0000|      ;      11/1/82 - Change external entry to monitor interface
0000|      ;      so that error code displayed on same line as
0000|      ;      message if no icon displayed
0000|      ;      11/3/82 - Made following changes:
0000|      ;      1)Move creation of test icon display till
0000|      ;      after keyboard reset so translation can
0000|      ;      be done if necessary
0000|      ;      2)Do cursor, mouse init only once so
0000|      ;      cursor posn not reset
0000|      ;      3)Optimize mouse, cursor routines
0000|      ;      4)Correct COPSCMD routine
0000|      ;      5)Upgrade check for Profile routine and
0000|      ;      optimize Profile read code
0000|      ;      11/8/82 - Conditionally add check for keyboard connected
0000|      ;      routine.
0000|      ;      11/9/82 - Create version 2.07
0000|      ;      11/11/82 - Modify ROM checksum algorithm
0000|      ;      11/12/82 - Add diskette eject on power-off
0000|      ;      11/13/82 - 1)Remove Dump Memory/Verify Disk from Service
0000|      ;      mode menu
0000|      ;      2)Add speaker beep and specific read/write
0000|      ;      loop for memory sizing and lo mem errors

```



```

0000|      ;      11/15/82 - 1)Add keyboard/mouse disconnect check
0000|      ;      2)Remove memory "clear" from sizing test - now
0000|      ;      done after memory testing
0000|      ;      11/16/82 - 1)Change power-cycle invoking to CMD/SHIFT/P
0000|      ;      key sequence.
0000|      ;      2)Change customer monitor mode invoking to
0000|      ;      CMD/ENTER (on key pad) key sequence.
0000|      ;      3)Add wait for profile loop in boot menu
0000|      ;      display routine
0000|      ;      4)Add timeout to general wait for clock
0000|      ;      input routine
0000|      ;      5)Increase delay for poweroff wait loop
0000|      ;      6)Optimize character display routine
0000|      ;      11/18/82 - 1)Add save of error code to special parameter
0000|      ;      memory area for use during burnin.
0000|      ;      2)Add context check for MMU testing
0000|      ;      3)Create version 2.08 for internal release
0000|      ;      11/19/82 - 1)Change initial position of cursor to center
0000|      ;      of screen.
0000|      ;
0000|      ;      11/19/82 - Release versions 2.08 (internal) and
0000|      ;      2.09 (for manufacturing)
0000|      ;
0000|      ;      12/15/82 - Add:
0000|      ;      1)Setting of VIA PCR reg for later use
0000|      ;      2)Reset of keyboard before boot
0000|      ;      3)Fix for slot 3 card check for boot menu
0000|      ;      12/16/82 - Add:
0000|      ;      1)Move Profile cmd buffer to location $304
0000|      ;      2)Change default boot device to Profile
0000|      ;      3)Remove support for third boot port on
0000|      ;      each slot
0000|      ;      4)Expand id range for test card search
0000|      ;      5)Don't display Restart button after boot
0000|      ;      error
0000|      ;      6)New icons
0000|      ;      12/18/82 - Fix memory test bug
0000|      ;
0000|      ;      1/3/83 - Fix bug in reporting parity circuitry
0000|      ;      failure. Change version to 2.10.
0000|      ;      1/7/83 - Make following changes:
0000|      ;      1)Change keyboard sequences for I/O slot
0000|      ;      booting
0000|      ;      2)Extend timeout for initial Profile check
0000|      ;      1/11/83 - Change SCC test to use max baud rate for
0000|      ;      loopback test
0000|      ;      1/12/83 - Add running of expansion card status routines

```

```

0000|           ;           when configuration check is done
0000|           ;           1/18/83 - Add fixes for:
0000|           ;           1)Continuing after memory error
0000|           ;           2)Checking for no reset function
0000|           ;           4)Read of I/O slot ROM for icon data -le 2 meg
0000|           ;           ensure odd address for icon count
0000|           ;           5)Default boot setting when loop on memory
0000|           ;           test selected
0000|           ;           1/21/83 - Add save of disk controller self-test status
0000|           ;
0000|           ;           1/28/83 - Create and release ROM version 2.11
0000|           ;
0000|           ;           3/15/83 - Extend Profile timeout for case where drive
0000|           ;           may be parking head. (bug RM016)
0000|           ;           4/20/83 - Add fixes for:
0000|           ;           1)Memory sizing (bug RM015).
0000|           ;           2)Garbage sent out serial port (RM014).
0000|           ;           3)Removed 6504 (bug RM013).
0000|           ;           4)Never ready Profile (bug RM011).
0000|           ;           Also do some code optimization in icon
0000|           ;           routines to make room for fixes. (RM000)
0000|           ;           4/22/83 - Do code optimization for setting bus error
0000|           ;           vector (labeled as RM000).
0000|           ;           Add changes for following requests:
0000|           ;           1)Display ROM id's on bootup (CHG001)
0000|           ;           2)Loop on address 1Meg-2 if sizing error (CHG002)
0000|           ;           3)Turn off contrast before doing poweroff (CHG003)
0000|           ;           4)Change copyright notice. (CHG005)
0000|           ;           Also modify alert msg display routine (CHG005).
0000|           ;           4/26/83 - Add loop on CPU diags if no memory or I/O
0000|           ;           board installed. Also toggle LED. (CHG004)
0000|           ;           4/27/83 - Do only basic memory test on warm-start. (CHG006)
0000|           ;           Add fix for NMI bug (RM010).
0000|           ;           5/9/83 - Made following changes:
0000|           ;           1)Change ROM id display to rev # (D) (CHG001)
0000|           ;           2)Change ROM test failure to loop at fixed address
0000|           ;           $00FE00C8 (end of jump table) (CHG007)
0000|           ;           3)Make correction for screen not cleared when
0000|           ;           continuing from I/O slot error to boot menu.
0000|           ;           (CHG008)
0000|           ;           5/10/83 - Add change to enable display of uncompressed icons
0000|           ;           upon external entry to ROM Monitor (CHG008).
0000|           ;
0000|           ;           5/12/83 - Create and release rev D of boot ROM.
0000|           ;
0000|           ;           8/8/83 - Add changes for Pepsi system: (CHG009)
0000|           ;           1) New icons.

```

```

0000|      ;      2) Display of icons with id #'s.
0000|      ;      8/9/83 - Add save of disk ROM id in low memory. (CHG010)
0000|      ;      Add fixes for:
0000|      ;      1) SCC init for Applebus. (CHG011)
0000|      ;      2) Test card boot search. (CHG012)
0000|      ;      8/10/83 - Delete inverse video check. (CHG013)
0000|      ;      Add fix to beep routine. (CHG014)
0000|      ;      8/16/83 - Delete memory address and ping pong routines,
0000|      ;      add routines to decode parity error to
0000|      ;      chip. (CHG015)
0000|      ;      9/1/83 - Add retry for hard disk booting. (CHG016)
0000|      ;      Add jump table entry for write to
0000|      ;      parameter memory routine. (CHG017)
0000|      ;      9/2/83 - Add new font, modify display routines. (CHG018)
0000|      ;      Add wait for hard disk ready when
0000|      ;      power-cycling. (CHG019)
0000|      ;      9/6/83 - Add setting of video latch whenever boot
0000|      ;      error causes jump to ROM low memory default
0000|      ;      vectors. (CHG020)
0000|      ;      Add fix for memory test/initialization
0000|      ;      bug. (CHG021)
0000|      ;      9/7/83 - Add read of disk controller ROM self-test
0000|      ;      results. (CHG022)
0000|      ;      Add skip of disk eject on power-off if any
0000|      ;      disk controller errors occurred. (CHG023)
0000|      ;
0000|      ;      9/8/83 - Release for testing (rev 3B) with Pepsi systems.
0000|      ;
0000|      ;      10/10/83 - 1)Make Pepsi icon changes. (CHG024)
0000|      ;      2)Add fix for proper setting of carry bit
0000|      ;      on floppy or hard disk boots. (CHG025)
0000|      ;      3)Add fix for video reset on boot from not ready
0000|      ;      Profile. (CHG026)
0000|      ;      10/12/83 - Add change to reset SCC for Applebus before
0000|      ;      doing memory test. (CHG027)
0000|      ;      10/20/83 - Add fix for service mode bus error problem. (CHG028)
0000|      ;
0000|      ;      10/20/83 - Release as rev E for Lisa and Pepsi systems.
0000|      ;
0000|      ;      12/15/83 - 1)Add new code to determine system type. (CHG029)
0000|      ;      2)Change default boot device for Lisa 2
0000|      ;      system if no hard disk connected. (CHG030)
0000|      ;      3)Extend timeout for hard disk ready. (CHG031)
0000|      ;      4)Add bug fix for wrong icon display on Lisa 2.
0000|      ;      (CHG032)
0000|      ;      5)Add bug fix for menu display when mouse or

```



```

0000|                                     ;          keyboard not connected. (CHG033)
0000|                                     ;          6) Remove save of error code in parameter memory.
0000|                                     ;          (CHG034)
0000|                                     ;          12/16/83 - Release as rev 'X' for testing
0000|                                     ;
0000|                                     ;          12/21/83 - Release as official rev 'F' for all systems
0000|                                     ;
0000|                                     ;          1/25/84 - 1) Add code to properly initialize Profile-reset
0000|                                     ;                  and parity-reset lines for Profile booting      (CHG036)
0000|                                     ;          2/7/84 - 1) Extend hard disk default read timeout to 16
0000|                                     ;                  seconds for Widget systems. (CHG037)
0000|                                     ;                  2) Add delay after hard disk reset for Widget
0000|                                     ;                  systems. (CHG038)
0000|                                     ;          2/8/84 - Release as rev G for testing
0000|                                     ;
0000|                                     ;          2/24/84 - Release as official rev H
0000|                                     ;
0000|-----
0000|                                     . PAGE
0000|-----
0000|                                     ; Macro definitions
0000|-----
0000|
0000|                                     .MACRO BSR6
0000|                                     LEA    @1,A6
0000|                                     BRA     %1
0000| @1
0000|                                     .ENDM
0000|
0000|                                     .MACRO BSRS6
0000|                                     LEA     @1,A6
0000|                                     BRA.S   %1
0000| @1
0000|                                     .ENDM
0000|
0000|                                     .MACRO RTS6
0000|                                     JMP      (A6)
0000|                                     .ENDM
0000|
0000|                                     .MACRO BSR4
0000|                                     LEA     @1,A4
0000|                                     BRA     %1
0000| @1
0000|                                     .ENDM
0000|
0000|                                     .MACRO BSRS4
0000|                                     LEA     @1,A4

```



```

0000|          BRA.S    %1
0000|          @1
0000|          .ENDM
0000|
0000|          .MACRO    RTS4
0000|          JMP      (A4)
0000|          .ENDM
0000|
0000|          .MACRO    BSR2
0000|          LEA      @1,A2
0000|          BRA      %1
0000|          @1
0000|          .ENDM
0000|
0000|          .MACRO    BSRS2
0000|          LEA      @1,A2
0000|          BRA.S    %1
0000|          @1
0000|          .ENDM
0000|
0000|          .MACRO    RTS2
0000|          JMP      (A2)
0000|          .ENDM
0000|
0000|          .MACRO    DISABLE
0000|          MOVE     SR,-(SP)
0000|          ORI      #$0700,SR
0000|          .ENDM
0000|
0000|          .MACRO    ENABLE
0000|          MOVE     (SP)+,SR
0000|          .ENDM
0000|
0000|          .PAGE
0000|          ;-----
0000|          ;          Conditionals for assembly
0000|          ;-----
0000|
0000| 0000 0001          DIAGS          .EQU    1          ;controls assembly of selected diags
0000| 0000 0001          NEWLISA        .EQU    1          ;controls extra code for new LISA's
0000| 0000 0001          BURNIN         .EQU    1          ;controls code for burnin cycling
0000| 0000 0001          NORESET        .EQU    1          ;controls code for reset feature
0000| 0000 0000          EXTERNAL       .EQU    0          ;controls listing of externally
0000|                                     ; callable routines only (w/ EQU's)
0000| 0000 0001          ROM16K         .EQU    1          ;controls code to be added when 16K
0000|                                     ; ROM's available
0000| 0000 0001          NEWTWIG        .EQU    1          ;controls code for new Twiggy firmware

```



```

0000|                                     ; interface
0000| 0000 0000      FINLISA      .EQU    0      ;controls code for final LISA's
0000| 0000 0001      FINKBD      .EQU    1      ;controls check for final keyboard
0000| 0000 0000      AAPL        .EQU    0      ;controls Apple monitor code
0000| 0000 0001      USERINT     .EQU    1      ;controls code for new user interface
0000| 0000 0000      DEBUG       .EQU    0      ;controls global equate allocation
0000| 0000 0000      ROM4K       .EQU    0      ;controls code for 2716 version
0000| 0000 0000      ROM8K       .EQU    0      ;controls code for 2732 version
0000| 0000 0001      BMENU       .EQU    1      ;controls format of boot menu
0000|                                     ; 1 = pull down menu
0000| 0000 0001      FULLSCC      .EQU    1      ;controls code for SCC tests
0000| 0000 0000      INVERTCK     .EQU    0      ;controls code for inverse video check      CHG013
0000|
0000|          .IF      EXTERNAL = 1
0000|          .ENDC
0000|          .PAGE
0000|
0000|          ;-----
0000|          ;      GENERAL EQUATES
0000|          ;-----
0000| 00FE 0000      ROMBASE      .EQU    $00FE0000      ;BASE ADDRESS FOR ROM
0000| 0000 00FE      ROMSLCT     .EQU    $00FE          ;MSB'S OF ROM ADDRESS
0000| 00FC 0000      IOSPACE     .EQU    $00FC0000      ;START OF IO SPACE
0000| 00FC E800      VIDLTCH     .EQU    $00FCE800      ;VIDEO ADDRESS LATCH
0000| 0000 002F      DEFVID      .EQU    $2F            ;default setting for video latch
0000|                                     ; (end of 512K board in slot 1)
0000| 0000 00AF      DEFVID2     .EQU    $AF            ;default video latch setting and LED on
0000|
0000|          .IF      DEBUG = 0
0000| 0000 0110      SCRNBASE     .EQU    $110           ;ptr to base address for video page
0000|          .ELSE
0000|          .ENDC
0000|
0000|          .IF      USERINT = 0
0000|          .ELSE
0000| 0000 005A      RBYTES      .EQU    90             ;BYTES FOR EACH DISPLAY ROW
0000|          .ENDC
0000| 0000 010E      TOPOFFSET    .EQU    270           ;offset for first row from top of screen
0000| 0000 00E1      RLONGS      .EQU    225           ;longs for each row
0000| 0000 0000      R0          .EQU    0             ;ROW 0 OFFSET
0000| 0000 005A      R1          .EQU    R0+90         ;ROW 1 OFFSET, ETC.
0000| 0000 00B4      R2          .EQU    R1+90
0000| 0000 010E      R3          .EQU    R2+90
0000| 0000 0168      R4          .EQU    R3+90
0000| 0000 01C2      R5          .EQU    R4+90
0000| 0000 021C      R6          .EQU    R5+90
0000| 0000 0276      R7          .EQU    R6+90
0000| 0000 0008      BUSVCTR     .EQU    $0008         ;BUS EXCEPTION VECTOR

```

```

0000| 0000 000C      ADRVCTR      .EQU    $000C      ;ADDRESS EXCEPTION VECTOR
0000| 0000 0010      ILLVCTR      .EQU    $0010      ;ILLEGAL INSTRUCTION VECTOR
0000| 0000 0028      L10VCTR      .EQU    $0028      ;line 1010 trap
0000| 0000 002C      L11VCTR      .EQU    $002C      ;line 1111 trap
0000| 0000 007C      NMIVCT      .EQU    $007C      ;NMI VECTOR LOCATION
0000| 0000 0080      TRPVCT0      .EQU    $0080      ; TRAP 0 VECTOR LOCATION
0000| 0020 0000      MAXADR      .EQU    $00200000    ; MAX RAM ADDRESS + 1 (2 meg)
0000| 0010 0000      ONEMEG      .EQU    $00100000    ; 1 meg in hex
0000| 0008 0000      HALFMEG      .EQU    $00080000    ; 1/2 meg
0000| 0004 0000      QTRMEG      .EQU    $00040000    ; 256K
0000| 0002 0000      ROW2ADR      .EQU    $00020000    ; 128K - START OF 2ND MEMORY ROW
0000| 0000 0480      STKBASE      .EQU    $0480      ; DEFAULT BASE FOR STACK
0000| 0000 0480      CALLBASE      .EQU    $0480      ; STACK BASE FOR USE BY CALL ROUTINE
0000| 00FC E012      SETUP      .EQU    $00FCE012    ; ADDRESS TO TURN SETUP BIT OFF
0000| 00FC E010      SETUPON      .EQU    $00FCE010    ; ADDRESS TO TURN SETUP ON
0000| AA55 A55A      PATRN      .EQU    $AA55A55A    ; PATTERN FOR MEMORY TESTING
0000| 0000 A55A      PATRN2      .EQU    $A55A      ; PATTERN FOR MMU TEST
0000| 00FC E01E      PARON      .EQU    $00FCE01E    ;PARITY ENABLE
0000| 00FC E01C      PAROFF      .EQU    $00FCE01C    ;PARITY DISABLE
0000| 00FC F000      MEALTCH      .EQU    $00FCF000    ;MEMORY ERROR ADDRESS LATCH
0000| 00FC F801      STATREG      .EQU    $00FCF801    ;ERROR STATUS REGISTER
0000| 0000 0000      SFER      .EQU    0      ; SOFT ERROR BIT
0000| 0000 0001      PBIT      .EQU    1      ; HARD ERROR (PARITY) BIT
0000| 0000 0002      VRBIT      .EQU    2      ; VR BIT LOCATION
0000| 0000 0004      VIDBIT      .EQU    4      ; VID BIT
0000| 0000 0005      CSBIT      .EQU    5      ; CSYNC BIT
0000| 0000 0006      INVIDBIT      .EQU    6      ; INVERSE VIDEO BIT
0000| 0000 0020      RETRYCNT      .EQU    32      ;RETRY COUNT FOR MEMORY SIZING
0000| 00FC E018      VTIRDIS      .EQU    $00FCE018    ;VERTICAL RETRACE DISABLE
0000| 00FC E01A      VTIRENB      .EQU    $00FCE01A    ;VERTICAL RETRACE ENABLE
0000| 0008 0000      HEX512K      .EQU    $80000      ;512K in hex
0000| 0002 0000      HEX128K      .EQU    $20000      ;128K in hex
0000| 0001 8000      HEX96K      .EQU    $18000      ;96K in hex
0000| 0000 8000      HEX32K      .EQU    $8000      ;32K in hex
0000| 0000 2000      HEX8K      .EQU    $2000      ;8K in hex
0000| 0000 0800      HEX2K      .EQU    $0800      ;2K in hex
0000| 0000 0800      LOMEM      .EQU    HEX2K      ;amount of memory initially tested
0000| 00FC E006      DG2ON      .EQU    $00FCE006    ;WRITE WRONG PARITY ENABLE
0000| 00FC E004      DG2OFF      .EQU    $00FCE004    ;WRITE WRONG PARITY DISABLE
0000| 0003 D090      ONESEC      .EQU    $3D090      ;1 second delay constant
0000| 0007 A120      TWOSEC      .EQU    ONESEC*2      ;2 second delay
0000| 0013 12D0      FIVESEC      .EQU    ONESEC*5      ;5 second delay
0000| 0000 F424      QTRSEC      .EQU    ONESEC/4      ;0.25 second delay
0000| 0000 61A8      TNTHSEC      .EQU    ONESEC/10      ;0.1 second delay
0000| 0006 7C28      KBDDLY      .EQU    <ONESEC*17>/10    ;1.7 second delay
0000| 0001 E848      HALFSEC      .EQU    ONESEC/2      ;0.5 second delay
0000|

```



```

0000|                ; Equates for memory parity error routine
0000|
0000| 0000 0040      MSRCHSZ      .EQU    64                ;main memory error range          CHG015
0000| 0000 8000      VSRCHSZ      .EQU    32768           ;video memory error range         CHG015
0000| 0000 FFFF 8000      VMSK      .EQU    $FFFF8000      ;mask for video errors           CHG015
0000| 0000 0000 0003      ADRMSK     .EQU    $03            ;mask for error byte address     CHG015
0000| 0008 0000      PHYTOLOG     .EQU    $80000          ;physical to logical address offset CHG015
0000|
0000|                .IF      EXTERNAL = 1
0000|                .ENDC
0000|
0000|                ; Equates for VIA registers (offsets from $XXD181 or $XXD101)
0000|
0000| 00FC DD81      VIA1BASE     .EQU    $00FCDD81          ;BASE ADDRESS FOR COPS 6522
0000| 0000 0000      ORB1        .EQU    $0                ;PORT B OUTPUT REG
0000| 0000 0002      ORA1        .EQU    $2                ;PORT A OUTPUT REG
0000| 0000 0004      DDRB1       .EQU    $4                ;PORT B DATA DIRECTION REG
0000| 0000 0006      DDRA1       .EQU    $6                ;PORT A DATA DIRECTION REG
0000| 0000 000C      T1LL1       .EQU    $C                ;LOW ORDER T1 LATCH
0000| 0000 000E      T1LH1       .EQU    $E                ;HIGH ORDER T1 LATCH
0000| 0000 0010      T2CL1       .EQU    $10              ;LOW ORDER T2 COUNTER
0000| 0000 0012      T2CH1       .EQU    $12              ;HIGH ORDER T2 COUNTER
0000| 0000 0014      SHR1        .EQU    $14              ;SHIFT REG
0000| 0000 0016      ACR1        .EQU    $16              ;AUXILIARY CONTROL REG
0000| 0000 0018      PCR1        .EQU    $18              ;PERIPHERAL CONTROL REG
0000| 0000 001A      IFR1        .EQU    $1A              ;INTERRUPT FLAG REG
0000| 0000 001C      IER1        .EQU    $1C              ;INTRPT ENABLE REG
0000| 0000 001E      PORTA1      .EQU    $1E              ;PORT A WITH NO HANDSHAKE
0000|
0000| 0000 0004      FDIR         .EQU    4                ;PORT B, BIT 4 HAS FDIR STATE
0000| 00FC D901      VIA2BASE     .EQU    $00FCD901          ;BASE ADDRESS FOR OTHER 6522
0000| 0000 0000      ORB2        .EQU    $0                ;PORT B OUTPUT REG
0000| 0000 0000      IRB2        .EQU    $0                ;PORT B INPUT REG
0000| 0000 0008      ORA2        .EQU    $8                ;PORT A OUTPUT REG
0000| 0000 0008      IRA2        .EQU    $8                ;PORT A INPUT REG
0000| 0000 0010      DDRB2       .EQU    $10              ;PORT B DATA DIRECTION REG
0000| 0000 0018      DDRA2       .EQU    $18              ;PORT A DATA DIRECTION REG
0000| 0000 0030      T1LL2       .EQU    $30              ;LOW ORDER T1 LATCH
0000| 0000 0038      T1LH2       .EQU    $38              ;HIGH ORDER T1 LATCH
0000| 0000 0040      T2CL2       .EQU    $40              ;LOW ORDER T2 COUNTER
0000| 0000 0048      T2CH2       .EQU    $48              ;HIGH ORDER T2 COUNTER
0000| 0000 0060      PCR2        .EQU    $60              ;PERIPHERAL CONTROL REG
0000| 0000 0078      PORTA2      .EQU    $78              ;PORT A WITH NO HANDSHAKE
0000|
0000| 0000 0006      DSKDIAG      .EQU    6                ;port B, bit 6 is disk alive indicator
0000|
0000| 00FC D01C      CSTRB        .EQU    $00FCD01C          ;STROBE FOR CONTRAST LATCH

```

```

0000|          ; Equates for PIA registers (offsets from $XXA001) (SLOT 2)
0000| 00FC A001    PIABASE      .EQU    $00FCA001      ;BASE ADDRESS FOR PIA CARD IN SLOT 2
0000| 0000 0000    INDATA      .EQU    $0
0000| 0000 0002    OUTDATA     .EQU    $2
0000| 0000 0004    INCSR      .EQU    $4
0000| 0000 0006    OUTCSR     .EQU    $6
0000|
0000|          ; Equates for SCC
0000|
0000| 00FC D241    SCCBCTL      .EQU    $FCD241          ;SCC channel B control
0000| 0000 0002    ACTL        .EQU    2              ;offset to SCC channel A control
0000| 0000 0004    SCCDATA     .EQU    4              ;offset to SCC data regs
0000| 0000 0000    RXBF        .EQU    0              ;receive buffer full bit
0000| 0000 0002    TXBE        .EQU    2              ;transmit buffer empty bit
0000|
0000|          .PAGE
0000|          .IF USERINT = 0
0000|          .ELSE
0000| 0000 0000 0000    MMU        .EQU    0              ;MMU ERROR
0000| 0000 0001 0001    CPUSEL    .EQU    1              ;CPU selection logic error
0000| 0000 0002 0002    VID        .EQU    2              ;CPU VIDEO LOGIC ERROR
0000| 0000 0003 0003    PAR        .EQU    3              ;CPU PARITY LOGIC ERROR
0000| 0000 0004 0004    CPUINTR    .EQU    4              ;UNEXPECTED INTERRUPT OCCURRED
0000| 0000 0005 0005    BUSEXCP    .EQU    5              ;BUS ERROR
0000| 0000 0006 0006    ADREXCP    .EQU    6              ;ADDRESS ERROR
0000| 0000 0007 0007    MISEXCP    .EQU    7              ;MISC EXCEPTION
0000| 0000 0008 0008    ILLEXCP    .EQU    8              ;ILLEGAL INSTRUCTION ERROR
0000| 0000 0009 0009    TRPEXCP    .EQU    9              ;line 1111 or 1010 trap
0000|
0000| 0000 000A 000A    VIA1        .EQU    10             ;COPS VIA ERROR
0000| 0000 000B 000B    VIA2        .EQU    11             ;PARALLEL PORT VIA ERROR
0000| 0000 000C 000C    IOCOPS     .EQU    12             ;IO BOARD COPS ERROR
0000| 0000 000D 000D    KBDCOPS    .EQU    13             ;KEYBOARD COPS ERROR
0000| 0000 000E 000E    CLK        .EQU    14             ;CLOCK ERROR
0000| 0000 000F 000F    RS232A     .EQU    15             ;RS232 PORT A ERROR
0000| 0000 0010 0010    RS232B     .EQU    16             ;RS232 PORT B ERROR
0000| 0000 0011 0011    DISK        .EQU    17             ;DISK ERROR
0000| 0000 0012 0012    IOEXCP     .EQU    18             ;UNEXPECTED IO EXCEPTION OCCURRED
0000| 0000 0013 0013    IOCOPS2    .EQU    19             ;COPS reset code error
0000| 0000 0014 0014    IOKBD      .EQU    20             ;I/O or keyboard failure
0000|
0000| 0000 0015 0015    MEM        .EQU    21             ;MEMORY ERROR
0000| 0000 0016 0016    MPAR        .EQU    22             ;memory parity error
0000|
0000| 0000 0017 0017    KBDOUT      .EQU    23             ;KEYBOARD DISCONNECTED
0000| 0000 0018 0018    MOUSOUT    .EQU    24             ;MOUSE DISCONNECTED
0000| 0000 0019 0019    IO1ERR     .EQU    25             ;I/O slot 1 failure

```

```

0000| 0000 001A      IO2ERR      .EQU    26      ;I/O slot 2 failure
0000| 0000 001B      IO3ERR      .EQU    27      ;I/O slot 3 failure
0000| 0000 001C      ALTBOOT      .EQU    28      ;alternate boot key request
0000| 0000 001D      BTMENU       .EQU    29      ;boot menu request
0000| 0000 001E      WRMSTRT      .EQU    30      ;warm-start indicator
0000|
0000| 0000 001F      LOOP          .EQU    31      ;loop on test
0000| 0E7F FFFF      ERRMSK       .EQU    $0E7FFFFF ;MASK FOR ERROR CHECKING
0000| 0000 000F      CPUMSK       .EQU    $0000000F ;MASK FOR CPU ERROR CHECKING
0000| 0000 03F0      EXMSK        .EQU    $000003F0 ;mask for exception error checking
0000| 001F DC00      IOMSK        .EQU    $001FDC00 ;MASK FOR I/O ERROR CHECKING
0000| 0060 0000      MEMMSK       .EQU    $00600000 ;mask for memory error checking
0000| 0180 0000      OTHRMSK      .EQU    $01800000 ;mask for keyboard/mouse check
0000| 0E00 0000      IOSMSK       .EQU    $0E000000 ;mask for I/O slot error checking
0000| 001E 3FFA      CONTMSK      .EQU    $001E3FFA ;mask for CONTINUE option - allow continue
0000|                                     ; for MMU,VIDEO,CLK,RS232,MEM,MPAR,KBDOUT,
0000|                                     ; MOUSOUT,and IO slot errors
0000| 0018 3000      SCANMSK      .EQU    $00183000 ;mask for results of initial keyboard scan
0000| 7000 0000      ALTBMSK      .EQU    $70000000 ;mask for D7 when CONTINUE option invoked
0000| 008F FFFF      BOOTMSK      .EQU    $008FFFFF ;mask for errors that continue to boot attempt
0000| 001F FFFF      CPIOMSK      .EQU    $001FFFFF ;mask for checking for CPU and IO errors
0000|                                     .ENDC
0000|
0000|                                     ; Equates for error codes displayed to user
0000|
0000|                                     .IF NEWTWIG = 0
0000|                                     .ELSE
0000| 0000 0028      EMMU          .EQU    40      ;MMU ERROR
0000| 0000 0029      ECPUSEL      .EQU    41      ;CPU selection logic error
0000| 0000 002A      EVID         .EQU    42      ;CPU VIDEO LOGIC ERROR
0000| 0000 002B      ECPAR        .EQU    43      ;CPU PARITY LOGIC ERROR
0000| 0000 002C      ECPUINTR     .EQU    44      ;UNEXPECTED INTERRUPT OCCURRED
0000| 0000 002D      EBUSEXCP     .EQU    45      ;BUS ERROR
0000| 0000 002E      EADREXCP     .EQU    46      ;ADDRESS ERROR
0000| 0000 002F      EMISEXCP     .EQU    47      ;MISC EXCEPTION
0000| 0000 0030      EILLEXCP     .EQU    48      ;ILLEGAL INSTRUCTION ERROR
0000| 0000 0031      ETRPEXCP     .EQU    49      ;line 1111 or 1010 trap
0000|
0000| 0000 0032      EVIA1        .EQU    50      ;COPS VIA ERROR
0000| 0000 0033      EVIA2        .EQU    51      ;PARALLEL PORT VIA ERROR
0000| 0000 0034      EIOCOP       .EQU    52      ;IO BOARD COPS ERROR
0000| 0000 0035      EKBCOP       .EQU    53      ;KEYBOARD COPS ERROR
0000| 0000 0036      ECLK         .EQU    54      ;CLOCK ERROR
0000| 0000 0037      ERS232A      .EQU    55      ;RS232 PORT A ERROR
0000| 0000 0038      ERS232B      .EQU    56      ;RS232 PORT B ERROR
0000| 0000 0039      EDISK        .EQU    57      ;DISK ERROR
0000| 0000 003A      EIOEXCP      .EQU    58      ;UNEXPECTED IO EXCEPTION OCCURRED

```

```

0000| 0000 003B      EIOCOP2      .EQU    59          ;IO board COPS code error
0000| 0000 003C      EIOKBD       .EQU    60          ;I/O or keyboard error
0000|
0000| 0000 0046      EMEM          .EQU    70          ;R/W MEMORY ERROR
0000| 0000 0047      EPAR          .EQU    71          ;PARITY ERROR
0000| 0000 004B      EBOOT         .EQU    75          ;general boot failure error code
0000|                ; Special COPS error codes for burnin cycling
0000|
0000| 0000 003D      SERR1          .EQU    61          ;error setting initial time
0000| 0000 003E      SERR2          .EQU    62          ;error setting alarm
0000|                .ENDC
0000|                ; Secondary status flag (STATFLGS) equates
0000| 0000 0000      NORSTRT       .EQU    0          ;governs display of restart button
0000| 0000 0001      NOCONT        .EQU    1          ;error disallows Monitor CONTINUE option
0000| 0000 0002      MSBTN         .EQU    2          ;mouse button detected
0000| 0000 0003      CMDFLG        .EQU    3          ;cmd button up/down
0000| 0000 0004      MOUSE         .EQU    4          ;mouse button up/down
0000| 0000 0005      CHKCMD        .EQU    5          ;if =1 user input from keyboard must
0000|                ; be prefaced by CMD key
0000| 0000 0006      BTN           .EQU    6          ;flag for button use
0000| 0000 0007      MENU          .EQU    7          ;flag for menu use
0000|
0000|                ; MMU equates
0000|
0000| 0000 8000      MMUSADRL       .EQU    $00008000    ;STARTING MMU LIMIT ADDRESS
0000| 0000 8008      MMUSADRB       .EQU    $00008008    ;STARTING MMU BASE ADDRESS
0000| 00FE 8000      MMUEADRL       .EQU    $00FE8000    ;ENDING MMU LIMIT ADDRESS
0000| 00FE 8008      MMUEADRB       .EQU    $00FE8008    ;ENDING MMU BASE ADDRESS
0000| 0002 0000      ADRI28K        .EQU    $00020000    ;128K IN HEX - INCR FOR MMU REGS PTRS
0000| 0000 0100      PAG128K       .EQU    $00000100    ;128K PAGE INCREMENT FOR ORG REGS
0000| 0000 0700      MEMLMT        .EQU    $0700        ;LIMIT VALUE FOR MEMORY SEGMENTS
0000| 0000 08FF      NMEMLMT       .EQU    $08FF        ;INVERSE OF VALUE (HIGH NIBBLE IGNORED)
0000|
0000|                .IF ROM4K = 0
0000| 0000 0900      IOLMT          .EQU    $0900        ;LIMIT VALUE FOR I/O SEGMENT
0000| 0000 06FF      NIOLMT        .EQU    $06FF        ;INVERSE
0000| 0000 0901      IOLMT2        .EQU    $0901        ;limit value for no reset feature
0000| 0000 0FFE      RSTLMT        .EQU    $0FFE        ;inverse mask for no reset feature
0000|                .ELSE
0000|                .ENDC
0000| 0000 0F00      SPLMT          .EQU    $0F00        ;LIMIT VALUE FOR SPECIAL I/O SPACE
0000| 0000 00FF      NSPLMT        .EQU    $00FF        ;INVERSE
0000| 0000 0C00      INVPAG        .EQU    $0C00        ;INVALID PAGE LIMIT
0000| 0000 8008      MMUOB         .EQU    $00008008    ;ADDRESS OF ORG REG 0 (FOR LOW MEMORY)
0000| 0000 8000      MMUOL         .EQU    $00008000    ;ADDRESS OF LIMIT REG 0
0000| 00FC 8008      MMU126B       .EQU    $00FC8008    ;ADDRESS OF ORG REG 126 (FOR I/O SPACE)
0000| 00FC 8000      MMU126L       .EQU    $00FC8000    ;ADDRESS OF LIMIT REG 126

```



```

0000| 00FE 8008      MMU127B      .EQU    $00FE8008      ;ADDRESS OF BASE REG 127 (FOR ROM SPACE)
0000| 00FE 8000      MMU127L      .EQU    $00FE8000      ;ADDRESS OF LIMIT REG 127
0000| 00FC E00A      SEG1ON       .EQU    $00FCE00A      ;CONTEXT SELECTION BIT 1 ENABLE
0000| 00FC E008      SEG1OFF      .EQU    $00FCE008      ;CONTEXT SELECTION BIT 1 DISABLE
0000| 00FC E00E      SEG2ON       .EQU    $00FCE00E      ;CONTEXT SELECTION BIT 2 ENABLE
0000| 00FC E00C      SEG2OFF      .EQU    $00FCE00C      ;CONTEXT SELECTION BIT 2 DISABLE
0000|
0000|                ; Equates for serial number read routine
0000|
0000| 0000 0009      Dlycnst       .equ     9              ;constant for delay loop
0000| 0000 00AC      TKiller       .equ     172            ;time killer constant
0000| 0000 0007      BytesPerRead .equ     7              ;bytes per read
0000| 0000 000E      WordsPerRead .equ     BytesPerRead*2    ;during reading one byte fits into one word
0000| 0000 0070      HalfSize      .equ     WordsPerRead*8    ;half the size of ScrachSize
0000| 0000 00E0      ScrachSize    .equ     HalfSize*2        ;size of the scrach array
0000|
0000|                ;I/O segment 126
0000| 00FE 8000      Snum          .equ     $0fe8000        ;location of SN1 & SN2
0000|
0000|                ;special I/O segment 127
0000| FFFF FFFC      dLcnt         .equ     -4              ;displacement for local variable LOOP COUNTER
0000| FFFF FFF8      dSavArray     .equ     dLcnt-4          ;disp. for Save Array pointer
0000| FFFF FF18      dScrach       .equ     dSavArray-ScrachSize
0000|
0000|                ;disp. for pointer to local array SCRACH
0000| FFFF FF18      dStack        .equ     dScrach          ;disp. for the Link
0000|
0000|                ; Equates for COPS and keyboard scan
0000|
0000| 0000 0086      MOUSDWN       .EQU     $86              ;MOUSE BUTTON PRESSED
0000| 0000 00FF      CMDKEY       .EQU     $FF              ;LEFT COMMAND KEY
0000| 0000 00FD      ALPHKEY      .EQU     $FD              ;ALPHA LOCK KEY "DOWN"
0000|
0000|                .IF NEWTWIG = 0
0000|                .ELSE
0000| 0000 00F4      KEY1          .EQU     $F4              ;'1' key - for Twiggy #1 boot
0000| 0000 00F1      KEY2          .EQU     $F1              ;'2' key - for Twiggy #2 boot
0000| 0000 00F2      KEY3          .EQU     $F2              ;'3' key - for Profile boot
0000| 0000 00F0      AKEY          .EQU     $F0              ;'A' key - for I/O slot #3, port 1
0000| 0000 00EE      BKEY          .EQU     $EE              ;'B' key - for I/O slot #3, port 2
0000| 0000 00ED      CKEY          .EQU     $ED              ;'C' key - for I/O slot #3, port 3
0000|                ;DKEY          .EQU     $FB              ;'D' key - for I/O slot #1, port 4
0000|                ;EKEY          .EQU     $E0              ;'E' key - for I/O slot #2, port 4
0000|                ;FKEY          .EQU     $E9              ;'F' key - for I/O slot #3, port 4
0000| 0000 00AF      ENTRKEY      .EQU     $AF              ;Right ENTER key - for Monitor invoking
0000| 0000 00FE      SHFTKEY      .EQU     $FE              ;Shift key - used for power-cycling
0000| 0000 00C4      PKEY          .EQU     $C4              ;'P' key - for Power-cycling
0000|                .ENDC
0000|
0000| 0000 0080      RSTCODE       .EQU     $80              ;reset code

```



```

0000| 0000 00FD      KUNPLG      .EQU    $FD          ;keyboard unplugged
0000| 0000 00FE      ICERR       .EQU    $FE          ;I/O board COPS RAM error
0000| 0000 00FF      KCERR       .EQU    $FF          ;keyboard COPS RAM error
0000| 0000 0007      MSUNPLG     .EQU    $07          ;mouse unplugged
0000| 0000 0087      MSPLG       .EQU    $87          ;mouse plugged in
0000|
0000|                .IF      EXTERNAL = 1
0000|                .ENDC
0000|
0000|                ; Equates for Boot device id's
0000|                .IF      NEWTWIG = 0
0000|                .ELSE
0000| 0000 0000      TWIG1        .EQU    $0            ;TWIGGY DRIVE #1
0000| 0000 0001      TWIG2        .EQU    $1            ;TWIGGY DRIVE #2
0000| 0000 0002      PROFILE     .EQU    $2            ;PROFILE HARD DISK
0000| 0000 0003      IO1PORT1    .EQU    $3            ;I/O SLOT 1, port 1
0000| 0000 0004      IO1PORT2    .EQU    $4            ;I/O SLOT 1, port 2
0000| 0000 0006      IO2PORT1    .EQU    $6            ;I/O SLOT 2, port 1
0000| 0000 0007      IO2PORT2    .EQU    $7            ;I/O SLOT 2, port 2
0000| 0000 0009      IO3PORT1    .EQU    $9            ;I/O SLOT 3, port 1
0000| 0000 000A      IO3PORT2    .EQU    $A            ;I/O SLOT 3, port 2
0000| 0000 000F      PC          .EQU    $F            ;power cycle mode
0000| 0000 0010      MON         .EQU    $10           ;abort boot, go to monitor id
0000|
0000|                .ENDC
0000|                .IF USERINT = 0
0000|                .ENDC
0000|
0000|                ; Equates for device code display (ASCII codes)
0000|
0000|                .IF NEWTWIG = 0
0000|                .ELSE
0000| 0000 0031      TWG1          .EQU    $31           ;Twiggy drive #1
0000| 0000 0032      TWG2          .EQU    $32           ;Twiggy drive #2
0000| 0000 0033      PRO          .EQU    $33           ;Profile
0000| 0000 0034      IOS1          .EQU    $34           ;I/O slot 1
0000| 0000 0037      IOS2          .EQU    $37           ;I/O slot 2
0000| 0000 0041      IOS3          .EQU    $41           ;I/O slot 3
0000|                .ENDC
0000|
0000|                ;-----
0000|                ; Equates for Disk controller shared memory/Twiggy boot
0000|                ;-----
0000|
0000| 0000 0001      TWIGGY        .EQU    1              ;controls Twiggy code assembly (1 = YES)
0000| 00FC C001      DISKMEM      .EQU    $00FCC001      ;base address of shared memory
0000| 0000 0002      CMD          .EQU    2              ;offset for command byte

```



```

0000| 0000 0004      DRV      .EQU    CMD+2      ;offset for drive #
0000| 0000 0006      SIDE      .EQU    DRV+2      ;side #
0000| 0000 0008      SCTR      .EQU    SIDE+2     ;sector #
0000| 0000 000A      TRAK      .EQU    SCTR+2     ;track #
0000|
0000|                .IF NEWTWIG = 0
0000|                .ENDC
0000|                .IF NEWTWIG = 1
0000| 0000 000C      SPEED      .EQU    TRAK+2     ;motor speed control
0000| 0000 000E      CNFRM      .EQU    SPEED+2    ;confirm for format cmd
0000| 0000 0010      STAT      .EQU    CNFRM+2    ;error status
0000| 0000 0012      INTLV      .EQU    STAT+2     ;interleave factor          CHG022
0000| 0000 0014      TYPE      .EQU    INTLV+2    ;drive type id              CHG009
0000| 0000 0016      STST      .EQU    TYPE+2     ;self-test result           CHG022
0000| 0000 0030      ROMV      .EQU    $30        ;ROM version #
0000| 0000 0058      RTRYCNT    .EQU    $58        ;retry count
0000| 0000 005E      INTSTAT    .EQU    $5E        ;interrupt status
0000| 0000 00BA      CHKCNT     .EQU    $BA        ;data checksum error count
0000| 0000 00C4      CHKCNT2    .EQU    $C4        ;address checksum error count
0000| 0000 03E8      DSKBUFF     .EQU    $3E8      ;start of disk buffer
0000| 0000 0400      DSKDATA     .EQU    DSKBUFF+24 ;first 12 bytes are header
0000| 00FC C031      DISKROM     .EQU    $FCC031    ;absoulte address for disk ROM id
0000| 0000 0005      SLOTMR      .EQU    5         ;id bit for slow timers      CHG029
0000| 0000 0006      FASTMR      .EQU    6         ;id bit for fast timers      CHG029
0000|
0000| 0000 0000      READS      .EQU    0         ;read sector w/checksum
0000| 0000 0001      WRT        .EQU    1         ;
0000| 0000 0002      UNCLAMP     .EQU    2         ;unclamp diskette
0000| 0000 0003      FMT        .EQU    3         ;
0000| 0000 0004      VFY        .EQU    4         ;verify disk
0000| 0000 0009      CLAMP      .EQU    9         ;clamp disk
0000| 0000 00FF      OK         .EQU    $FF        ;confirmation for format
0000|
0000| 0000 0083      SEEK        .EQU    $83        ;seek cmd
0000|                .ENDC
0000|
0000| 0000 0081      EXRW        .EQU    $81        ;execute cmd
0000| 0000 0085      CLRSTAT     .EQU    $85        ;clear status cmd
0000| 0000 0086      ENBLINT     .EQU    $86        ;enable intrpt
0000| 0000 0087      DSABLINT    .EQU    $87        ;disable intrpt
0000| 0000 0088      SLEEP       .EQU    $88        ;loop in RAM cmd
0000| 0000 0089      DIE        .EQU    $89        ;loop in ROM cmd
0000|
0000| 0000 0000      DRV1        .EQU    0         ;drive #1 ID
0000| 0000 0080      DRV2        .EQU    $80        ;drive #2 ID
0000| 0000 0001      TRK1        .EQU    1         ;track 1
0000| 0000 0000      TOPSIDE     .EQU    0         ;top side of disk

```

```

0000| 0000 0001      BOTSIDE      .EQU    1              ;bottom side of disk
0000|
0000|              .IF NEWTWIG = 0
0000|              .ENDC
0000|              .IF NEWTWIG = 1
0000| 0000 000C      HDRLEN      .EQU    12              ;length of Twiggy header
0000| 0000 0200      SECLEN      .EQU    512             ;length of one sector
0000| 0001 FFF4      TWGHDR      .EQU    $1FFF4          ;address to load boot header
0000| 0002 0000      TWGDATA      .EQU    $20000          ;address to load boot data
0000| 0000 06A6      LASTBLK     .EQU    1702            ;last block #
0000| 0000 06A6      DSKSIZE      .EQU    1702            ;total amount of blocks
0000|              .ENDC
0000|
0000|              ; Equates for parameter memory used by boot ROM
0000|
0000| 00FC C161      STATSTRT     .EQU    $FCC161          ;start of special parameter memory area for boot ROM
0000| 00FC C161      STATSAB      .EQU    STATSTRT        ;save of error code
0000| 00FC C17D      STATSUM      .EQU    $FCC17D          ;checksum word for special area
0000| 0000 0008      STATWRDS     .EQU    8              ;length in words (16 bytes)
0000|
0000| 00FC C181      PMSTRT      .EQU    $FCC181          ;start of system paramter memory
0000| 00FC C189      DVCCODE      .EQU    $FCC189          ;boot device code
0000| 00FC C18D      MEMCODE      .EQU    $FCC18D          ;mouse/memory test indicator byte
0000| 0000 0007      MOUSEON      .EQU    7              ;bit for mouse attached (1=yes)
0000| 0000 0006      EXMEM        .EQU    6              ;bit for extended memory test (1=yes)
0000| 00FC C1FD      PMCHKSM      .EQU    $FCC1FD          ;checksum word
0000| 0000 0020      PMWRDS      .EQU    32              ;length in words (64 bytes)
0000|
0000|              ; Equates for disk errors
0000|
0000|              .IF NEWTWIG = 0
0000|              .ENDC
0000|              .IF NEWTWIG = 1              ;new firmware, new error codes
0000| 0000 0007      DRVERR      .EQU    07              ;no disk in drive
0000| 0000 0007      NODISK      .EQU    DRVERR          ;another name for it
0000| 0000 0014      WRPERR      .EQU    20              ;write protect error
0000| 0000 0016      CLMPERR     .EQU    22              ;clamp error
0000| 0000 0017      RDWRERR     .EQU    23              ;read error
0000| 0000 0019      UCLMPERR    .EQU    25              ;unclamp error
0000| 0000 0026      BADTHDR     .EQU    38              ;bad header (not a boot file id)
0000| 0000 0027      TIMEOUT     .EQU    39              ;timeout error
0000|
0000| 0012 0000      CMDTIME      .EQU    $120000          ;timeout for taking command (15 secs)
0000| 00C0 0000      FDIRTIME     .EQU    $C00000          ;timeout for setting FDIR (2 mins)
0000| 0180 0000      VFYTIME      .EQU    <FDIRTIME*2>    ;timeout for verify disk operation (4 mins)
0000| 0018 0000      EJCTTIME     .EQU    $180000          ;timeout for ejecting disk (15 secs)
0000| 001C 8000      DSKTMOUT     .EQU    $1C8000          ;timeout for initial speed check (15 secs)

```



```

0000| 00C0 0000      INSRTTIM      .EQU    FDIRTIME      ;timeout for disk to be inserted (2 mins)
0000| 0180 0000      FMTTIME      .EQU    VFYTIME      ;timeout for format operation
0000|
0000|      .ENDC
0000|
0000|      ; Equates for disk interrupt status
0000|
0000| 0000 0000      DSK1IN      .EQU    0      ;drive #1 disk in place
0000| 0000 0001      BUTN1      .EQU    1      ;drive #1 button pushed
0000| 0000 0002      RWF1      .EQU    2      ;read/write/format done on drive #1
0000| 0000 0004      DSK2IN      .EQU    4      ;drive #2 disk in place
0000| 0000 0005      BUTN2      .EQU    5      ;drive #2 button pushed
0000| 0000 0006      RWF2      .EQU    6      ;read/write/format done on drive #2
0000|
0000|      ; Equates for disk status command response
0000|
0000| 0000 0002      DSKIN      .EQU    2      ;disk inserted
0000| 0000 0003      BUTN      .EQU    3      ;button pressed
0000|
0000| 00FC C015      DRVTYPE      .EQU    $FCC015      ;drive type id (0 = Twiggy,          CHG009
0000|                                          ; 1 = single SONY, 2 = double SONY)      CHG009
0000|
0000|      ;-----
0000|      ; Equates for use with Profile boot
0000|      ;-----
0000|
0000| 0000 0001      PROFILE      .EQU    1      ;controls assembly of Profile code
0000| 0000 0000      OCD      .EQU    0      ;OPEN CABLE DETECT INPUT
0000| 0000 0001      BSY      .EQU    1      ;BUSY LINE INPUT
0000| 0000 0304      CMDBUFR      .EQU    $304      ;BUFFER FOR COMMAND BYTES
0000| 0000 01B4      STATBFR      .EQU    $1B4      ;STATUS BYTE BUFFER (uses BOOTDATA area)
0000| 0000 01B4      STAT1      .EQU    $1B4      ;STATUS BYTE 1
0000| 0000 01B5      STAT2      .EQU    $1B5      ;STATUS BYTE 2
0000| 0000 01B6      STAT3      .EQU    $1B6      ;STATUS BYTE 3
0000| 0000 01B6      STAT4      .EQU    $1B6      ;STATUS BYTE 4
0000| C140 C000      STATMSK      .EQU    $C140C000      ;MASK FOR DON'T CARE STATUS BITS      CHG016
0000| 0000 0005      PCMSZ      .EQU    5      ;BYTES FOR READ CMD - 1
0000| 0000 0000      PCMD      .EQU    0      ; COMMAND CODE
0000| 0000 0001      BLKH      .EQU    1      ; HIGH BLOCK ADDRESS
0000| 0000 0002      BLKM      .EQU    2      ; MID BLOCK ADDRESS
0000| 0000 0003      BLKL      .EQU    3      ; LOW BLOCK ADDRESS
0000| 0000 0004      RETRY      .EQU    4      ; RETRY COUNT
0000| 0000 0005      THRESH      .EQU    5      ; THRESHOLD COUNT
0000| 0001 FFEC      HDRBUFR      .EQU    $1FFEC      ;BUFFER FOR HEADER
0000| 0000 0004      FILEID      .EQU    4      ; OFFSET TO FILEID
0000| 0000 AAAA      BOOTPAT      .EQU    $AAAA      ;FILEID FOR BOOT PATTERN
0000| 0002 0000      DATABFR      .EQU    $20000      ;BUFFER FOR DATA
0000| 0000 0014      HDRSIZE      .EQU    20      ;HEADER LENGTH

```

```

0000| 0000 0200          BLKSIZE      .EQU    512              ;BLOCK SIZE
0000| 0120 0000          STRTIME      .EQU   $1200000         ;STARTUP TIMEOUT after power-up = about 3 minutes
0000| 0090 0000          RSTRTIME     .EQU   $900000          ;STARTUP TIMEOUT after reset = ABOUT 100 SECS
0000| 0018 0000          RDTIME       .EQU   $180000          ;READ TIMEOUT = ABOUT 16 SECS
0000| 0000 0500          BSYTIME      .EQU   $0500            ;Wait for busy high = about 10 ms
0000| 0000 FFFF          RSPTIME     .EQU   $FFFF            ;RESPONSE TIMEOUT = ABOUT 500 ms
0000| 0000 000A          RCNT         .EQU    10              ;BOOT RETRY COUNT
0000| 0000 0003          TCNT         .EQU     3              ;THRESHOLD COUNT FOR 30% SPARING
0000|
0000|          ; Equates for Profile boot error conditions
0000|          .IF NEWTWIG = 0
0000|          .ELSE
0000| 0000 0050          NODSK          .EQU    80              ;DISK NOT ATTACHED
0000| 0000 0051          DSKBSY        .EQU    81              ;DISK NOT READY
0000| 0000 0052          BADRSP        .EQU    82              ;UNEXPECTED RESPONSE
0000| 0000 0053          STATNZ        .EQU    83              ;NONZERO STATUS BYTE
0000| 0000 0054          BADHDR        .EQU    84              ;INCORRECT HEADER
0000| 0000 0055          TMOUT         .EQU    85              ;TIMEOUT ERROR
0000|          .ENDC
0000|
0000|          ;-----
0000|          ; Equates for I/O slot booting
0000|          ;-----
0000|
0000| 00FC 0001          SLOT1L         .EQU   $FC0001          ;I/O slot 1 SL address
0000| 00FC 4001          SLOT2L         .EQU   $FC4001          ;I/O slot 2 SL address
0000| 00FC 8001          SLOT3L         .EQU   $FC8001          ;I/O slot 3 SL address
0000| 0000 000E          STBIT          .EQU    14              ;status bit in id
0000| 0000 000D          ICBIT          .EQU    13              ;icon bit in id
0000| 0000 000C          TSTBIT         .EQU    12              ;test card bit in id
0000| 0002 0000          STENTRY        .EQU   $20000          ;entry point for status routine
0000| 0002 0002          BENTRY        .EQU   $20002          ;boot routine entry point
0000| 0002 0004          ICONPTR        .EQU   $20004          ;pointer to icons, if any
0000| 0000 8001          APPLNET        .EQU   $8001           ;id for Applenet card
0000| 0000 9FFF          APPLQUAL      .EQU   $9FFF           ;qualifier for Applenet search
0000| 0000 1000          TSTCRD        .EQU   $1000           ;id for test card
0000| 0000 1800          TSTQUAL        .EQU   $1800           ;qualifier for test card search
0000|
0000|          ; Error codes for I/O slot booting
0000|
0000|          .IF NEWTWIG = 0
0000|          .ELSE
0000| 0000 005A          NOC              .EQU    90              ;no card installed
0000| 0000 005B          INV              .EQU    91              ;not bootable card
0000| 0000 005C          BADSM          .EQU    92              ;invalid checksum
0000| 0000 005D          BADST          .EQU    93              ;bad status returned
0000|          .ENDC

```

```

0000|
0000|
0000|          .PAGE
0000|          .IF      BURNIN = 1
0000|
0000|          ;-----
0000|          ;   Special equates for burnin cylcing code
0000|          ;-----
0000|
0000| 00FC C191          INITFLG          .EQU      $FCC191          ;first pass flag (01 = no)
0000| 00FC C193          HOURSAB          .EQU      $FCC193          ;save of last hour value from clock
0000| 00FC C195          LCNTHI          .EQU      $FCC195          ;loop count
0000| 00FC C197          LCNTLO          .EQU      $FCC197
0000| 00FC C199          TIMEFLG          .EQU      $FCC199          ;flag to indicate hour save needed
0000| 00FC C19B          MINSAB          .EQU      $FCC19B          ;save of minute value for Twiggy test
0000| 00FC C19D          DSKCNTH          .EQU      $FCC19D          ;disk read error count - high byte
0000| 00FC C19F          DSKCNTL          .EQU      $FCC19F          ;disk read error count - low byte
0000| 00FC C1A1          CLKSAVE          .EQU      $FCC1A1          ;saved clock value
0000| 00FC C1B1          ALRMSAB          .EQU      $FCC1B1          ;saved alarm value last set
0000| 00FC C1C1          CYCLCNT          .EQU      $FCC1C1          ;count of minutes for power cycling
0000| 00FC C1C3          CYCLVAL          .EQU      $FCC1C3          ;# of mins between power cycles
0000| 00FC C1C5          MINCNT          .EQU      $FCC1C5          ;count of minutes for debug mode
0000| 00FC C1FF          ENDFM          .EQU      $FCC1FF          ;end of parameter memory
0000| 0000 0000          SET1          .EQU      $0          ;initial alarm/year/dd setting
0000| 1000 0000          SET2          .EQU      $10000000          ;d/hh/mm/ss/t setting
0000| 0000 01BC          HOUR          .EQU      $1BC          ;location of latest hour value read
0000| 0000 01BD          MINUTE          .EQU      $1BD          ;location of latest minute value read
0000| 00E0 F000          ONEHOUR          .EQU      $00E0F000          ;one hour setting for alarm
0000| 0003 C000          ONEMIN          .EQU      $0003C000          ;one minute setting for alarm
0000| 0000 9000          TENSECS          .EQU      $00009000          ;ten seconds
0000| 0010 0000          DLYTIME          .EQU      $100000          ;delay for screen display
0000|
0000|          .ENDC
0000|
0000|          .PAGE
0000|
0000|          ;-----
0000|          ;   Equates for Monitor code and screen handling
0000|          ;-----
0000|
0000|          ;   Ascii code equates
0000|
0000| 0000 0000 003F          QUESTN          .EQU      $3F          ; ?
0000| 0000 0000 000D          RET          .EQU      $0D          ; CR
0000| 0000 0000 0008          BS          .EQU      $08          ; backspace
0000|
0000|          ;   Keyboard code equates
0000|
0000| 0000 0000 00F3          KEY4          .EQU      $F3          ; '4'

```



```

0000| 0000 00E4      KEY5          .EQU    $E4          ; '5'
0000| 0000 00E1      KEY6          .EQU    $E1          ; '6'
0000| 0000 00E2      KEY7          .EQU    $E2          ; '7'
0000| 0000 00E3      KEY8          .EQU    $E3          ; '8'
0000| 0000 00D0      KEY9          .EQU    $D0          ; '9'
0000| 0000 00F6      SKEY          .EQU    $F6          ; 'S'
0000| 0000 00FF      CmdDwn        .EQU    CMDKEY       ; Command key down
0000| 0000 007F      CmdUp         .EQU    $7F         ; Command key up
0000| 0000 0006      MousUp        .EQU    $06         ; Mouse button up
0000|
0000|                ; Low memory usage
0000|
0000| 0000 02C0      KBDBFR         .EQU    $2C0         ; keyboard buffer start
0000| 0000 0300      KBDEND        .EQU    $300         ; and end (64 chars max)
0000|
0000|                .IF USERINT = 0
0000|                .ELSE
0000| 0000 0000 0300      CRTROW       .EQU    $300         ; display row ptr
0000| 0000 0000 0302      CRTCOL     .EQU    $302         ; display col ptr
0000|                .ENDC
0000| 0000 0000 000C      MAXTEST     .EQU    12           ; max test # for LOOP option
0000|                .IF USERINT = 0
0000|                .ELSE
0000|                ; Equates for new user interface code
0000|
0000| 0000 0000 005A      ROWBYTES    .EQU    90           ; width of screen in bytes
0000| 0000 0000 02D0      MaxX       .EQU    720         ; width in pixels
0000| 0000 0000 016C      MaxY       .EQU    364         ; length in pixels
0000| 0000 0000 05A0      MENULINE   .EQU    1440        ; bottom line loc for menu
0000| 0000 0000 05FA      DESKLINE   .EQU    1530        ; top line loc for desktop
0000| 0000 0000 7FF8      DESKMT     .EQU    32760       ; bottom line loc for desktop
0000| AAAA 5555      DESKPATRN      .EQU    $AAAA5555    ; pattern for "grey" desktop
0000|
0000| 0000 0000 0014      WROW        .EQU    20           ; window row
0000| 0000 0000 0002      WCOL       .EQU    2           ; starting window col
0000| 0000 0000 0056      WINDWIDTH  .EQU    86          ; width of window in bytes
0000| 0000 0000 0140      WINDHIGH   .EQU    320         ; height of window in pixel lines
0000| 0000 0000 00B4      WMIDROW    .EQU    <WINDHIGH/2>+WROW ; middle row in window
0000| 0000 0000 002D      WMIDCOL    .EQU    <WINDWIDTH/2>+WCOL ; middle col in window
0000| 0000 0000 0017      W14COL     .EQU    <WINDWIDTH/4>+WCOL ; col 1/4 across window
0000| 0000 0000 0041      W34COL     .EQU    <WINDWIDTH/4>*3+WCOL ; col 3/4 across window
0000| 0000 0000 070A      WINDSTRT   .EQU    <WROW*ROWBYTES>+WCOL ; start of window
0000|
0000| 0000 0000 0031      ALBOXROW    .EQU    49           ; starting row for alert box
0000| 0000 0000 0006      ALBOXCOL   .EQU    6           ; starting col for alert box
0000| 0000 0000 004E      ALRTWIDTH  .EQU    78          ; width of alert box
0000| 0000 0000 00A4      ALRTHIGH   .EQU    164         ; height of alert box

```



```

0000| 0000 1140      ALRTSTRT      .EQU    <ALBOXROW*ROWBYTES>+ALBOXCOL ;upper left corner of alert box
0000| 0000 0083      MIDALROW      .EQU    ALBOXROW+<ALRTHIGH/2>          ;middle row of alert box
0000| 0000 002D      MIDALCOL      .EQU    ALBOXCOL+<ALRTWIDTH/2>        ;middle col of alert box
0000|
0000| 0000 000A      BTNWIDTH      .EQU    10                          ;width of button
0000| 0000 001C      BTNHIGH      .EQU    28                          ;height of button
0000| 0000 10E0      BTNSPC       .EQU    48*ROWBYTES                  ;space between upper left corner of buttons
0000| 0000 0392      BTNMSPC      .EQU    <10*ROWBYTES>+BTNWIDTH+4      ;position of button label relative to button
0000| 0000 0045      BTNRROW      .EQU    ALBOXROW+20                  ;starting display row for buttons
0000| 0000 0034      BTNCOL       .EQU    52                          ;starting display col for buttons
0000| 0000 1876      BTN1STRT     .EQU    <BTNRROW*ROWBYTES>+BTNCOL      ;location of first button
0000| 0000 2956      BTN2STRT     .EQU    BTN1STRT+BTNSPC              ;location of second button
0000| 0000 3A36      BTN3STRT     .EQU    BTN2STRT+BTNSPC              ;location of third button
0000| 0000 1C08      BTN1MSG      .EQU    BTN1STRT+BTNMSPC              ;location of button descriptions
0000| 0000 2CE8      BTN2MSG      .EQU    BTN2STRT+BTNMSPC
0000| 0000 3DC8      BTN3MSG      .EQU    BTN3STRT+BTNMSPC
0000|
0000| 0000 05A2      MENUSTRT     .EQU    MENULINE+2                    ;start of pull down menu
0000| 0000 000B      MENULEN      .EQU    11                          ;length of menu per entry
0000| 0000 03DE      MENUSPC      .EQU    990                          ;vertical space between menu entries
0000| 0000 0012      MENUWIDTH    .EQU    18                          ;width of pull down menu
0000| 0000 0111      MENULOC      .EQU    273                          ;start pt for menu heading
0000| 0000 0658      MENU1MSG     .EQU    MENUSTRT+182                  ;location of first menu entry
0000| 0000 0010      MBARLEN      .EQU    16                          ;height of menu bar
0000|
0000| 0000 0007      MITEMS       .EQU    7                            ;number of menu items
0000| 0000 20B4      MENUEND      .EQU    MENUSTRT+<MITEMS*MENUSPC>      ;bottom of menu
0000|
0000|                .IF  BMENU = 1
0000| 0000 0012      BMENUWIDTH   .EQU    MENUWIDTH                    ;width of pull down menu
0000| 0000 0022      BMENULEN     .EQU    34                          ;length of each boot menu entry
0000| 0000 0BF4      BMENUSPC     .EQU    <BMENULEN*90>                  ;vertical space between boot menu entries
0000|                .ENDC
0000|
0000| 0000 0042      DBOXWIDTH    .EQU    84-MENUWIDTH                  ;width of dialog box
0000| 0000 0014      DBOXHIGH     .EQU    20                          ;height of dialog box
0000| 0000 0168      DBOXTOP      .EQU    4*ROWBYTES                  ;dialog box spacing down from menu line
0000| 0000 0014      DBOXLEFT     .EQU    MENUWIDTH+2                  ;dialog box spacing left from menu
0000| 0000 071E      DBOXSTRT     .EQU    MENUSTRT+DBOXLEFT+DBOXTOP      ;start of dialog box
0000| 0000 0018      DBOXROW      .EQU    <DBOXSTRT/90>+4              ;pixel row for dialog msg
0000| 0000 0018      DBOXCOL      .EQU    MENUWIDTH+6                  ;byte col for dialog msg
0000|
0000| 0000 07BC      SVCTOP       .EQU    <DBOXHIGH+2>*ROWBYTES          ;service window spacing down
0000|                                ; from top of dialog box
0000| 0000 0014      SVCLEFT      .EQU    MENUWIDTH+2                  ;service window spacing left from menu
0000| 0000 0EDA      SVCSTRT      .EQU    DBOXSTRT+SVCTOP              ;left corner for service window
0000| 0000 0042      SVCWIDTH     .EQU    84-MENUWIDTH                  ;width of service window

```



```

0000| 0000 0140          SVCHIGH          .EQU    320                ;length of service window
0000|
0000| 0000 003E          FIRSTROW          .EQU    <SVCSTRT/90>+20        ;first row for display of msgs
0000| 0000 0018          FIRSTCOL          .EQU    MENUWIDTH+6          ;first column
0000| 0000 012C          ROWSLEFT          .EQU    SVCHIGH-20          ;pixel rows to bottom of service window
0000| 0000 001B          CHARROWS          .EQU    <ROWSLEFT/10>-3      ;rows used for character display
0000| 0000 014C          LASTROW           .EQU    <CHARROWS*10>+FIRSTROW ;last pixel row for display
0000| 0000 0058          LASTCOL           .EQU    FIRSTCOL+SVCWIDTH-2    ;last column
0000| 0000 000A          ROWLINES          .EQU    10                  ;pixel row lines per character
0000| 0000 0042          ROWLEN            .EQU    <LASTCOL-FIRSTCOL>|1+1 ;bytes per pixel row (must be even!)
0000| 0000 001B          NROWS             .EQU    <LASTROW-FIRSTROW>/ROWLINES ;number of character rows
0000| 0000 0008          CHRHIGH           .EQU    8                  ;character height in pixel lines
0000| 0000 0001          CHRWIDTH          .EQU    1                  ;width of char in bytes
0000| 0000 000A          CHRSPC            .EQU    CHRHIGH+2          ;vert pixel lines between chars
0000|
0000| 0000 0006          ICONWIDTH          .EQU    6                  ;width in bytes of icons
0000| 0000 0020          ICONHIGH           .EQU    32                  ;height of icons in pixel rows
0000|
0000| 0000 0031          TSTROW             .EQU    ALBOXROW            ;starting row for test alert box
0000| 0000 000A          TSTCOL             .EQU    10                  ;starting col for test alert box
0000| 0000 1144          TSTWSTRT           .EQU    <TSTROW*ROWBYTES>+TSTCOL ;test alert box start
0000| 0000 0046          TSTWWIDTH          .EQU    70                  ;width for test alert box
0000| 0000 0054          TSTWHIGH           .EQU    84                  ;height for test alert box
0000| 0000 0040          TSTMROW            .EQU    TSTROW+15          ;row for test message display
0000| 0000 000E          TSTMCOL            .EQU    TSTCOL+4          ;col for test message display
0000| 0000 005B          MIDTSTROW           .EQU    TSTROW+<TSTWHIGH/2>    ;middle row of test box
0000| 0000 004B          CHKROW             .EQU    MIDTSTROW-<ICONHIGH/2> ;row for check mark display
0000| 0000 0055          TSTIROW            .EQU    CHKROW+10          ;row for test icon display
0000| 0000 0014          TSTICOL            .EQU    TSTCOL+10          ;col for test icon display
0000| 0000 000E          TSTISPC            .EQU    ICONWIDTH+8        ;space between test icons
0000|
0000| 0000 1DF6          CPUSTRT            .EQU    <TSTIROW*ROWBYTES>+TSTICOL ;upper left corner for CPU icon
0000| 0000 1E04          MEMSTRT            .EQU    CPUSTRT+TSTISPC      ;upper left corner for MEM icon
0000| 0000 1E12          IOSTRT             .EQU    MEMSTRT+TSTISPC      ;upper left corner for I/O icon
0000| 0000 1E20          XCRDSTRT           .EQU    IOSTRT+TSTISPC      ;upper left corner for slot icon
0000|
0000| 0000 0073          ERRROW             .EQU    MIDALROW-<ICONHIGH/2> ;row for error icon display
0000| 0000 0010          ERRCOL             .EQU    ALBOXCOL+10        ;col for error icon display
0000| 0000 287E          ERRSTRT            .EQU    <ERRROW*ROWBYTES>+ERRCOL ;start address for error icon display
0000| 0000 0073          ALRTROW            .EQU    ERRROW            ;row for alert icon display
0000| 0000 0010          ALRTCOL            .EQU    ERRCOL            ;col for alert icon display
0000| 0000 0097          CODEROW            .EQU    ERRROW+36          ;row for error code display
0000| 0000 0012          CODECOL            .EQU    ERRCOL+2          ;col for error code display
0000| 0000 007E          MSGROW             .EQU    ALRTROW+11        ;row for alert/error message display
0000| 0000 0018          MSGCOL             .EQU    ALRTCOL+8          ;col for alert/error message display
0000| 0000 0010          MEMROW             .EQU    16                  ;offset row for memory board id # display
0000| 0000 0004          MEMCOL             .EQU    4                  ;offset col for memory board id # display

```

```

0000| 0000 0012          DISKROW          .EQU    18          ;offset row for diskette id # display
0000| 0000 0004          DISKCOL          .EQU     4          ;offset col for diskette id # display      CHG024
0000| 0000 0016          SLOTRW          .EQU    22          ;offset row for slot card id # display
0000| 0000 0003          SLOTCOL          .EQU     3          ;offset col for slot card id # display
0000| 0000 0006          DRVROW          .EQU     6          ;offset row for drive id # display      CHG009
0000| 0000 0003          DRVCOL          .EQU     3          ;offset col for drive id # display      CHG009
0000| 0000 0005          INSRTRW          .EQU     5          ;offset row for insert rqst id # display CHG009/CHG024
0000| 0000 0004          INSRCOL          .EQU     4          ;offset col for insert rqst id # display CHG009
0000|
0000|
0000| 0000 00A4          DEFROW          .EQU    WMIDROW-<ICONHIGH/2> ;row for default boot icon display
0000| 0000 003E          DEFCOL          .EQU    W34COL-<ICONWIDTH/2> ;col for default boot icon display
0000| 0000 39E6          DEFSTRT          .EQU    <DEFROW*ROWBYTES>+DEFCOL ;start address for default boot icon
0000| 0000 0014          ALTCOL          .EQU    W14COL-<ICONWIDTH/2> ;col for alternate boot icon display
0000| 0000 1B72          COL1STRT          .EQU    <78*ROWBYTES>+6      ;start for first column of boot icons
0000| 0000 1B7E          COL2STRT          .EQU    COL1STRT+12        ;start for second col of boot icons
0000| 0000 39BC          COL2MID          .EQU    <DEFROW*ROWBYTES>+ALTCOL ;middle of second col
0000| 0000 1B8A          COL3STRT          .EQU    COL2STRT+12        ;start for third col of boot icons
0000| 0000 1680          ICONCSPC          .EQU    <64*ROWBYTES>        ;space between left corner of icons in col
0000| 0000 0440          ICONMSPC          .EQU    <12*ROWBYTES>+6+2    ;start addr offset for boot icon alternate keycode
0000| 0000 000C          ICONRSPC          .EQU    12          ;space in between cols in same row
0000|
0000| 0000 0445          ALTKYADDR          .EQU    ICONMSPC+5          ;address offset for display of alternate keycode in menu bar
0000|
0000| 0000 0056          PCWIDTH          .EQU    86          ;width of window for power cycling msgs
0000| 0000 00C0          PCHIGH          .EQU    192          ;height of window
0000| 0000 070A          PCSTRT          .EQU    WINDSTRT          ;upper left corner of window
0000| 0000 0059          PCROW          .EQU    ALBOXROW+40         ;first row for power cycle msgs
0000| 0000 000C          PCCOL          .EQU    ALBOXCOL+6          ;first col for power cycle msgs
0000|
0000| 0000 0003          ROMIDROW          .EQU     3          ;cursor row ptr for ROM id display      CHG001
0000| 0000 0050          ROMIDCOL          .EQU    80          ;cursor col ptr for ROM id display      CHG001
0000|
0000|          .IF  DEBUG = 0
0000| 0000 0480          GLOBALS          .EQU    STKBASE          ;start of global area for mouse/cursor
0000|          .ELSE
0000|          .ENDC
0000|
0000| 0000 0480          ClockBytes          .EQU    Globals          ;clock data save area
0000| 0000 0486          MousX          .EQU    ClockBytes+6        ;mouse X-coordinate (word)
0000| 0000 0488          MousY          .EQU    MousX+2          ;mouse Y-coordinate (word)
0000| 0000 048A          MousDx          .EQU    MousY+2          ;mouse delta-x (byte)
0000| 0000 048B          MousDy          .EQU    MousDx+1         ;mouse delta-y (byte)
0000| 0000 048C          MousScaling          .EQU    MousDy+1        ;0=disabled, else=enabled (byte)
0000| 0000 048E          MousThresh          .EQU    MousScale|1+1    ;mouse movement threshold (word)
0000|
0000| 0000 0490          CrsrHotx          .EQU    MousThresh+2      ;hotspot X-coordinate (word)

```

```

0000| 0000 0492      CrsrHoty      .EQU    CsrHotX+2      ;hotspot Y-coordinate (word)
0000| 0000 0494      CrsrHeight     .EQU    CsrHotY+2      ;cursor height, 0-32 (word)
0000| 0000 0496      CrsrX          .EQU    CsrHeight+2     ;cursor X-coordinate (word)
0000| 0000 0498      CrsrY          .EQU    CsrX+2         ;cursor Y-coordinate (word)
0000| 0000 049A      CrsrTracking   .EQU    CsrY+2         ;0=disabled, else=enabled (byte)
0000| 0000 049B      CrsrBusy       .EQU    CsrTracking+1   ;0=not busy, else=busy (byte)
0000| 0000 049C      CrsrVisible    .EQU    CsrBusy+1      ;0=not visible, else=visible (byte)
0000| 0000 049E      CrsrHidden     .EQU    CsrVisible|1+1  ;<=0 implies hidden (word)
0000| 0000 04A0      CrsrObscured   .EQU    CsrHidden+2   ;0=not obscured, else=obscured (byte)
0000|
0000| 0000 04A2      SavedData       .EQU    CsrObscured+2   ;data from under cursor (128 bytes)
0000| 0000 0522      SavedX         .EQU    SavedData+128   ;saved data X-coordinate (word)
0000| 0000 0524      SavedY         .EQU    SavedX+2        ;saved data Y-coordinate (word)
0000| 0000 0526      SavedRows      .EQU    SavedY+2        ;rows of saved data (word)
0000| 0000 0528      SavedAddr      .EQU    SavedRows+2     ;saved data screen address (long)
0000|
0000| 0000 052C      LwrRight        .EQU    SavedAddr+4     ;saved lower right corner address (word)
0000| 0000 052E      MsgLen         .EQU    LwrRight+2     ;length of dialog box msg (word)
0000|
0000|                      .IF  BMENU = 0
0000|                      .ELSE
0000| 0000 0530      MenuBase        .EQU    Msglen+2        ;address of last boot menu "box" (word)
0000| 0000 0532      IconAddr       .EQU    MenuBase+2      ;address for last boot icon displayed (word)
0000|                      .ENDC
0000|
0000| 0000 0534      IconCnt         .EQU    IconAddr+2      ;count of boot icons displayed (byte)
0000|
0000| 0000 0535      DRIVE          .EQU    IconCnt+1        ;drive id for dump/verify options (byte)
0000| 0000 0536      BLKNUM         .EQU    DRIVE+1         ;block # for dump option (word)
0000| 0000 0538      CONTXT         .EQU    BLKNUM+2        ;context for dump of MMU contents (byte)
0000|
0000| 0000 053A      RectCnt         .EQU    CONTXT+2        ;count for active rect table (word)
0000| 0000 053A      RectTable      .EQU    RectCnt         ;active rectangle table (same start)
0000|
0000|                      .ENDC
0000|                      .PAGE
0000|
0000| ;-----
0000| ; The following memory locations are reserved for ROM use to save test data
0000| ;-----
0000|
0000| 0000 0180      STATUS          .EQU    $0180          ;POWER-UP STATUS (0=OK)
0000| 0000 0184      SIZRSLT        .EQU    STATUS+4        ;memory sizing test results
0000| 0000 0186      MEMRSLT        .EQU    SIZRSLT+2       ;MEMORY TEST RESULTS
0000| 0000 0188      BOOTMEM        .EQU    MEMRSLT+2       ;result for boot area of memory (128K)
0000| 0000 01A6      PEADDR         .EQU    MEMRSLT+32      ;PARITY ERROR ADDRESS
0000| 0000 01AA      ADRLTCH        .EQU    PEADDR+4        ;CONTENTS OF MEMORY ADDRESS LATCH
0000| 0000 01AC      D7SAV         .EQU    ADRLTCH+2       ;save for D7 when exception occurs

```

```

0000| 0000 01B0          MMURSLT      .EQU    $01B0          ;MMU TEST RESULTS
0000| 0000 01B2          KEYID        .EQU    $01B2          ;Keyboard ID
0000| 0000 01B3          BOOTDVCE     .EQU    $01B3          ;BOOT DEVICE CODE
0000| 0000 01B4          BOOTDATA     .EQU    $01B4          ;BOOT FAILURE DATA
0000| 0000 01BA          CLKDATA      .EQU    $01BA          ;CLOCK SETTING READ
0000| 0000 01C0          DATARGS      .EQU    $01C0          ;DATA REG SAVE AREA
0000| 0000 01E0          ADDRREGS     .EQU    $01E0          ;ADDRESS REG SAVE AREA
0000| 0000 01F8          A6SAV        .EQU    $01F8          ;SAVE AREA FOR REG A6
0000| 0000 01FC          USPSAV       .EQU    $01FC          ;SAVE AREA FOR USER STACK PTR
0000|
0000| 0000 0240          SERNUM        .EQU    $0240          ;saved serial number (28 bytes)
0000| 0000 0260          KBDQPTR      .EQU    $0260          ;ptr for keyboard queue
0000|
0000| 0000 0268          XPCTADDR     .EQU    $0268          ;memory test address for parity error      CHG015
0000| 0000 026C          XPCTDATA     .EQU    $026C          ;memory test expected data                CHG015
0000| 0000 0270          ACTADDR      .EQU    $0270          ;parity error address, phase 2            CHG015
0000| 0000 0274          ACTDATA      .EQU    $0274          ;actual data read on parity error, phase 2 CHG015
0000| 0000 0278          PEADR2       .EQU    $0278          ;address read from error latch            CHG015
0000| 0000 027C          PCHPROW      .EQU    $027C          ;parity chip row                          CHG015
0000| 0000 027D          PCHIP        .EQU    $027D          ;parity chip id                           CHG015
0000|
0000| 0000 0280          EXCFC         .EQU    $0280          ; bus function code
0000| 0000 0282          EXCADR       .EQU    $0282          ; address of error
0000| 0000 0286          EXCIR        .EQU    $0286          ; instruction reg
0000| 0000 0288          EXCSR        .EQU    $0288          ; status reg
0000| 0000 028A          EXCPC        .EQU    $028A          ; PC at time of exception
0000| 0000 028E          EXCTYPE      .EQU    $028E          ; exception type
0000| 0000 0290          SUPSTK       .EQU    $0290          ;SUPERVISOR STACK PTR
0000| 0000 0294          MAXMEM       .EQU    $0294          ;MAX MEMORY ADDRESS + 1
0000| 0000 0298          IO1ID        .EQU    $0298          ;I/O SLOT 1 ID
0000| 0000 029A          IO2ID        .EQU    $029A          ;I/O SLOT 2 ID
0000| 0000 029C          IO3ID        .EQU    $029C          ;I/O SLOT 3 ID
0000| 0000 029E          IO1STAT      .EQU    $029E          ;I/O SLOT 1 STATUS
0000| 0000 029F          IO2STAT      .EQU    $029F          ;I/O SLOT 2 STATUS
0000| 0000 02A0          IO3STAT      .EQU    $02A0          ;I/O SLOT 3 STATUS
0000| 0000 02A1          IOROM        .EQU    $02A1          ;I/O ROM VERSION #
0000| 0000 02A2          STATFLGS     .EQU    $02A2          ;additional status indicators
0000| 0000 02A4          MINMEM       .EQU    $02A4          ;MINIMUM PHYSICAL ADDRESS
0000| 0000 02A8          TOTLMEM     .EQU    $02A8          ;total amount of memory
0000| 0000 02AC          SCCRSLT      .EQU    $02AC          ;SCC test results
0000| 0000 02AD          MEMSLOT      .EQU    $02AD          ;Slot # for memory board if memory error
0000| 0000 02AE          DSKRSLT     .EQU    $02AE          ;Disk controller self-test status byte (0=no error)
0000| 0000 02AF          SYSTYPE      .EQU    $02AF          ;System type (0 = Lisa 1; 1, 2, 3 = Lisa 2)  CHG029
0000| 0000 02B0          KBDQ         .EQU    $02B0          ;KEYBOARD QUEUE
0000| 0000 02C0          QEND         .EQU    $02C0          ;END OF Q
0000|
0000|
0000|          .INCLUDE RM248.K.TEXT

```



```

0000|
0000|          .IF      EXTERNAL = 1
0000|          .ENDC
0000|
0000|          .PAGE
0000|          ;          .ABSOLUTE          ;makes listing look nicer
0000|          .PROC    LISAROM,0
0000|
0000|          .ORG      0          ; ORG'ED AT 0 BUT RUNS AT $00FE0000
0000|
0000|          ; Reset vectors here to pick up SP and PC values
0000|
0000|          BASE
0000| 0000          .WORD    $0000          ;initial SP
0002| 0480          .WORD    STKBASE
0004|
0004| 00FE          .WORD    ROMSLCT          ;initial PC (assumes use of MMU reg 127)
0006| 00F6          .WORD    BEGIN
0008|
0008|          ; Set up next locations for exception vectors
0008|
0008| 00FE          BUSVCT .WORD    ROMSLCT          ; BUS ERROR VECTOR
000A| 0030          .WORD    EXCPERR
000C| 00FE          ADRVCT .WORD    ROMSLCT          ; ADDRESS ERROR
000E| 0030          .WORD    EXCPERR
0010| 00FE          ILLVCT .WORD    ROMSLCT          ; ILLEGAL INSTRUCTION
0012| 0030          .WORD    EXCPERR
0014| 00FE          DIVOVCT .WORD    ROMSLCT          ; DIVIDE BY ZERO ERROR
0016| 0030          .WORD    EXCPERR
0018| 00FE          CHKVCT .WORD    ROMSLCT          ; CHK INSTRUCTION
001A| 0030          .WORD    EXCPERR
001C| 00FE          TRAPVCT .WORD    ROMSLCT          ; TRAPV INSTRUCTION
001E| 0030          .WORD    EXCPERR
0020| 00FE          PRIVCT .WORD    ROMSLCT          ; PRIVILEGE VIOLATION
0022| 0030          .WORD    EXCPERR
0024| 00FE          TRCVCT .WORD    ROMSLCT          ; TRACE OPERATION
0026| 0030          .WORD    EXCPERR
0028| 00FE          L10VCT .WORD    ROMSLCT          ; OPCODE 1010 DETECTED
002A| 0030          .WORD    EXCPERR
002C| 00FE          L11VCT .WORD    ROMSLCT          ; OPCODE 1111 DETECTED
002E| 0030          .WORD    EXCPERR
0030|
0030|          ;-----
0030|          ; Exception and interrupt vector handler for ROM - resets SP and
0030|          ; tries a restart
0030|          ;-----
0030|

```

```

0030| 3E7C 0480      EXCPERR MOVEA  #STKBASE,SP      ;reset stack ptr
0034| 4287           CLR.L   D7                    ;clear error indicator      CHG004
0036| 6000 015C      BRA     ROMTST                ;and restart diags      CHG004
003A|
003A|               .PAGE
003A|
003A|               ;-----
003A|               ; Subroutine for saving registers and stack pointers
003A|               ;-----
003A|       SAVEREGS
003A| 21CF 0290      MOVE.L  SP,SUPSTK                ;save sup stack ptr
003E|       SAVEREG2
003E| 21CE 01F8      MOVE.L  A6,A6SAV                ;save other regs (that aren't reset)
0042| 4E6E           MOVE.L  USP,A6
0044| 21CE 01FC      MOVE.L  A6,USPSAV
0048| 3C7C 01F8      MOVE    #A6SAV,A6              ;set ptr for saving regs
004C| 48E6 FFFC      MOVEM.L D0-D7/A0-A5,-(A6)
0050| 4E75           RTS
0052|
0052|               ; use spare bytes for message
0052|
0052| 53 45 52 56 49 43 45  SVCMSG .ASCII 'SERVICE MODE' ;      RM000
0059| 20 4D 4F 44 45
005E| 00             .BYTE   0                      ;      RM000
005F|
005F| 00             .ORG    $60
0060|
0060|               ; The next set of vectors cover spurious and autovector interrupts
0060|
0060| 00FE           SPURVCT .WORD  ROMSLCT            ; SPURIOUS INTERRUPT
0062| 0030           .WORD  EXCPERR
0064| 00FE           LVL1VCT .WORD  ROMSLCT            ; INTERNAL I/O INTERRUPTS (DISK,VERT TRACE,ETC.)
0066| 0030           .WORD  EXCPERR
0068| 00FE           LVL2VCT .WORD  ROMSLCT            ; KEYBOARD INTERRUPT
006A| 0030           .WORD  EXCPERR
006C| 00FE           LVL3VCT .WORD  ROMSLCT            ; I/O SLOT 2 INTERRUPT
006E| 0030           .WORD  EXCPERR
0070| 00FE           LVL4VCT .WORD  ROMSLCT            ; I/O SLOT 1
0072| 0030           .WORD  EXCPERR
0074| 00FE           LVL5VCT .WORD  ROMSLCT            ; I/O SLOT 0
0076| 0030           .WORD  EXCPERR
0078| 00FE           LVL6VCT .WORD  ROMSLCT            ; RS-232
007A| 0030           .WORD  EXCPERR
007C| 00FE           LVL7VCT .WORD  ROMSLCT            ; NMI
007E| 00CA           .WORD  NMIEXP                    ;      RM010
0080|
0080|               .IF     EXTERNAL = 1

```

```

0080|          .ENDC
0080|
0080|          .PAGE
0080|          .ORG    $80
0080|          ;-----
0080|          ;  Jump Table for calling by external routines
0080|          ;-----
0080|
0080|          JMPTBL
0080| 4EFA 25D0      JMP      DORESET      ;go to restart point          RM000
0084| 4EFA 24AE      JMP      INITMON     ;jump to ROM Monitor
0088|
0088|          .IF  USERINT = 0
0088|          .ELSE
0088| 4EFA 3664      JMP      CONVRTD5     ;convert row ptr and display message
008C|          .ENDC
008C|
008C| 4EFA 052C      JMP      WRITMMU      ;write to set of MMU registers
0090| 4EFA 1EDE      JMP      PROREAD     ;Profile read a block routine
0094| 4EFA 1CE0      JMP      TWGREAD     ;Twiggy read a sector routine
0098|
0098|          .IF  DIAGS = 1
0098| 4EFA 0E16      JMP      RAMTEST     ;basic memory test
009C| 4E71          NOP                  ;
009E| 4E75          RTS                  ;          CHG015
00A0| 4E71          NOP                  ;          CHG015
00A2| 4E75          RTS                  ;          CHG015
00A4|          .ENDC
00A4|
00A4| 4EFA 052A      JMP      READMMU      ;read MMU registers
00A8| 4EFA 08AC      JMP      COPSCMD     ;Send COPS command
00AC|
00AC|          .IF  DIAGS = 1
00AC| 4EFA 11F2      JMP      READCLK     ;Read clock setting
00B0|          .ELSE
00B0|          .ENDC
00B0|
00B0| 4EFA 157E      JMP      DSPDEC      ;display hex error code in decimal
00B4| 4EFA 074C      JMP      CONSET2     ;for setting contrast
00B8| 4EFA 0A3C      JMP      TONE        ;to beep speaker
00BC| 4EFA 17CE      JMP      VFYCHKSM   ;verify checksum
00C0|
00C0|          .IF  ROM4K = 0
00C0| 4EFA 17BC      JMP      WRTSUM      ;rewrite parameter memory          CHG017
00C4| 4EFA 0B30      JMP      RDSERN     ;go read system serial #
00C8|          .ENDC
00C8|

```

```

00C8| ;***** Loop point for ROM test failure *****
00C8| 60FE SPIN BRA.S SPIN ;hang system CHG007
00CA|
00CA| ;*****
00CA|
00CA| .IF EXTERNAL = 1
00CA| .ENDC
00CA|
00CA| ;-----
00CA| ; NMI Exception Handler RM010
00CA| ;-----
00CA|
00CA| 4239 00FC E012 NMIEXP CLR.B SETUP ;enable memory access RM010
00D0| 0839 0001 00FC F801 BTST #1,STATREG ;parity error? RM010
00D8| 6614 BNE.S @1 ;skip if not to ignore RM010
00DA| 31F9 00FC F000 01AA MOVE MEALTC,ADRLTC ;save address if yes RM010
00E2| 4A39 00FC E01C TST.B PAROFF ;and toggle to clear error bit RM010
00E8| 4A39 00FC E01E TST.B PARON ; RM010
00EE| 4A39 00FC E010 @1 TST.B SETUPON ;return to SETUP state RM010
00F4| 4E73 RTE ; RM010
00F6|
00F6| .PAGE
00F6| ;-----
00F6| ; First do "warm-start" no reset check - scan I/O MMU regs to see if set up
00F6| ;-----
00F6|
00F6| BEGIN
00F6| .IF NORESET = 1
00F6| 3039 00FC 8000 MOVE MMU126L,D0 ;check reg 126 for special I/O space
00FC| 0240 0FFF ANDI #0FFF,D0 ;ignore don't care bits
0100| 0C40 0901 CMPI #IOLMT2,D0 ;for no reset, 126L = $x901 (x = random value)
0104| 664C BNE.S BEGIN2 ;skip if not set up
0106| 0279 0FFF 00FC 8008 ANDI #0FFF,MMU126B ;else also check 126B = $x000
010E| 6642 BNE.S BEGIN2
0110|
0110| ; Check OK - set MMU for ROM access and change SETUP before vectoring
0110|
0110| 33FC 0700 0000 8000 MOVE #MEMLMT,MMU0L ;set low memory for r/w (to save regs,etc.)
0118| 33FC 0901 00FC 8000 MOVE #IOLMT2,MMU126L ;set for I/O space access (reset value)
0120| 33FC 0F00 00FE 8000 MOVE #SPLMT,MMU127L ;set access for ROM space
0128| 4279 00FE 8008 MOVE #0,MMU127B
012E| 4239 00FC E012 CLR.B SETUP ;enable memory access
0134|
0134| 21CF 0290 MOVE.L SP,SUPSTK ;save supervisor stack ptr
0138| 3E7C 0480 MOVEA #STKBASE,SP ;move stack pointer for ROM use
013C| 6100 FF00 BSR.S SAVEREG2 ;save other registers
0140|

```



```

0140|                ; Restore ROM Monitor environment
0140|
0140|                BSR4   CONSET           ;go set default contrast
0140| 49FA 0006      #       LEA       @1,A4
0144| 6000 06B4      #       BRA       CONSET
0148|              #@1
0148| 95CA          SUBA.L   A2,A2           ;set for no icons
014A| 4280          CLR.L    D0             ; error codes
014C| 97CB          SUBA.L   A3,A3           ; or messages
014E| 6000 23F4      BRA     INIT1         ;exit directly to monitor (avoid resaving regs)
0152|
0152|                .ENDC
0152|
0152|                ;-----
0152|                ; Do second warm-start check to see if contrast should be reset
0152|                ;-----
0152|
0152| 4287          BEGIN2 CLR.L    D7           ;clear for error use
0154| 3039 00FE 8000  MOVE      MMU127L,D0      ;check reg 127 for ROM space
015A| 0240 0FFF      ANDI      #$0FFF,D0       ;ignore don't care bits
015E| 0C40 0F00      CMPI      #SPLMT,D0       ;expect 127L = $xF00 (x = random value)
0162| 6630          BNE.S     ROMTST           ;skip if not
0164| 0279 0FFF 00FE 8008 ANDI      #$0FFF,MMU127B ;else check if 127B = $x000
016C| 6626          BNE.S     ROMTST
016E|
016E|                ; Check OK - set MMU for I/O and ROM access and go set contrast
016E|
016E| 08C7 001E      BSET      #WRMSTRT,D7      ;set warm start indicator
0172| 7000          MOVEQ     #0,D0            ;clear for use
0174| 33FC 0900 00FC 8000 MOVE      #IOLMT,MMU126L ;set access for I/O space
017C| 33C0 00FC 8008 MOVE      D0,MMU126B
0182| 33FC 0F00 00FE 8000 MOVE      #SPLMT,MMU127L ;set access for ROM space
018A|
018A| 4E70          BEGIN3 RESET              ;ensure clean I/O state for "warm-start"
018C|
018C|                .IF NEWLISA = 1
018C|                BSR4     CONOFF          ;and go disable contrast
018C| 49FA 0006      #       LEA       @1,A4
0190| 6000 066E      #       BRA       CONOFF
0194|              #@1
0194|                .ELSE
0194|                .ENDC
0194|
0194|                .PAGE
0194|
0194|                ;-----
0194|                ; Start diagnostics - do ROM checksum test first; expected result = 0

```



```

0194| ;-----
0194|
0194| ROMTST
0194|     .IF  DIAGS = 1
0194|
0194|         CLR.L  D0                ;clear for checksum use
0196|         LEA    BASE,A0          ;init ROM address ptrs
019A|         LEA    LAST,A1
019E| DOSUM  ADD     (A0)+,D0          ;read location and add to sum
01A0|         ROL    #1,D0            ;rotate to catch multiple bit errors
01A2|         CMPA.L A0,A1            ;loop until done
01A4|         BNE.S  DOSUM
01A6|         ADD     (A0)+,D0        ;add checksum word
01A8|         BNE     SPIN            ;loop if error
01AC|         TST.L  D7              ;in loop mode?
01AE|         BMT.S  ROMTST          ;restart test if yes
01B0|
01B0|     .ENDC
01B0|     .PAGE
01B0| ;-----
01B0| ; Next do read/write and address test of MMU supervisor regs
01B0| ; Register Usage (by this routine and/or its subroutines):
01B0| ;     A0 = MMU reg pointer      D0 = test pattern
01B0| ;     A1 = last MMU limit reg addr  D1 = contents read from MMU reg
01B0| ;     A2 = MMU address increment  D2 = OR mask of results
01B0| ;     A3 = last MMU base reg addr  D3 = pattern expected at last error
01B0| ;     A4 = used for return address  D4 = final value for MMU reg
01B0| ;     A5 = MMU address of last error  D5 = unused
01B0| ;     A6 = used for return address  D6 = unused
01B0| ;     A7 = stack pointer          D7 = error indicator (0 = R/W error)
01B0| ;-----
01B0|
01B0| MMUTST
01B0|     .IF  DIAGS = 1
01B0|         BSR4    MMUINIT          ;initialize test variables
01B0|         #       LEA     @1,A4
01B4|         #       BRA     MMUINIT
01B8|         #@1
01B8|         BSR6    MMURW            ;and go do read/write test
01B8|         #       LEA     @1,A6
01BC|         #       BRA     MMURW
01C0|         #@1
01C0|         BNE.S   MMUERR          ;abort if error
01C2|
01C2|         BSRS4   MMUINIT          ;reinitialize
01C2|         #       LEA     @1,A4
01C6|         #       BRA.S  MMUINIT

```

```

01C8|                                     #@1
01C8|                                     BSR6   MMUACHK       ;and do address test
01C8| 4DFA 0006                         #       LEA       @1,A6
01CC| 6000 00A2                         #       BRA       MMUACHK
01D0|                                     #@1
01D0| 6604                             BNE.S   @2           ;skip if error
01D2| 6000 00F2                         BRA     SETMMU       ;else go do initial MMU setup
01D6| 4647                             @2     NOT     D7       ;set address error indicator
01D8|
01D8|                                     .PAGE
01D8|                                     ;-----
01D8|                                     ; The following code is used to toggle every address and data line
01D8|                                     ; going to the MMU if an error in the MMU context 0 tests is found.
01D8|                                     ; Reset signals indicate read/write or addressing error.
01D8|                                     ;-----
01D8|
01D8| MMUERR TST     D7                   ;check error type
01DA| 6702                         BEQ.S   @2
01DC| 4E70                         RESET                                ;two reset signals for address error
01DE| 4E70                         @2     RESET                   ;only one for R/W error
01E0|
01E0|                                     ; Toggle every data and address bit
01E0|
01E0| MMULP MOVE.L   #$00028000,A0       ;set MMU limit reg start address
01E6| 7201                         MOVEQ   #1,D1           ;and starting data pattern
01E8| 7407                         MOVEQ   #7,D2           ;and loop count
01EA|                             BSRS4   TSTLOOP       ;go toggle for limit regs
01EA| 49FA 0004                         #       LEA       @1,A4
01EE| 6010                         #       BRA.S   TSTLOOP
01F0|                                     #@1
01F0| 207C 0002 8008                 MOVE.L   #$00028008,A0       ;set MMU base reg start address
01F6| 7405                         MOVEQ   #5,D2           ;and loop count
01F8|                             BSRS4   TSTLOOP       ;go test base regs
01F8| 49FA 0004                         #       LEA       @1,A4
01FC| 6002                         #       BRA.S   TSTLOOP
01FE|                                     #@1
01FE|
01FE| BRA.S   MMUERR                   ;and loop indefinitely
0200|                                     ; Subroutine to do reg testing
0200|
0200| 2008                         TSTLOOP MOVE.L   A0,D0       ;save starting address
0202| 3081                         REGTST  MOVE   D1,(A0)       ;do write
0204| 3610                         MOVE   (A0),D3       ;then read
0206| E349                         LSL     #1,D1       ;update pattern
0208| 4840                         SWAP    D0           ;get address
020A| E348                         LSL     #1,D0       ;update and restore
020C| 4840                         SWAP    D0

```

```

020E| 2040          MOVE.L  D0,A0
0210| 5342          SUBQ    #1,D2          ;loop until done
0212| 66EE          BNE.S   REGTST
0214|              RTS4          ;exit
0214| 4ED4          #        JMP      (A4)
0216|
0216|              .PAGE
0216|              ;-----
0216|              ; Subroutine to do initial setup for MMU testing
0216|              ;-----
0216|
0216| 303C A55A      MMUINIT MOVE    #PATRN2,D0      ;set test pattern
021A| 7200          MOVEQ    #0,D1          ;clear for result/error use
021C| 7400          MOVEQ    #0,D2          ; use MOVEQ for speed
021E| 247C 0002 0000 MOVE.L  #ADR128K,A2      ;set up increment value
0224| 007C 0710      ORI     #$0710,SR      ;set extend bit and disable interrupts
0228|              RTS4
0228| 4ED4          #        JMP      (A4)
022A|
022A|              .PAGE
022A|              ;-----
022A|              ; Subroutine to do MMU Read/Write Test for all registers in one context.
022A|              ; Zero bit set in CCR if no errors.
022A|              ;-----
022A|
022A| 207C 0000 8000 MMURW  MOVE.L  #MMUSADRL,A0      ;SET MMU LIMIT START ADDR
0230| 227C 00FE 8000      MOVE.L  #MMUEADRL,A1      ;SET MMU LIMIT END ADDR
0236| 267C 00FE 8008      MOVE.L  #MMUEADRB,A3      ;SET MMU BASE END ADDR
023C|
023C|              RWCHK1 BSR4     CHKRW          ;GO DO READ/WRITE CHECK
023C| 49FA 0006          #        LEA     @1,A4
0240| 6000 0072          #        BRA     CHKRW
0244|              #@1
0244| 4640              NOT      D0          ;INVERT FOR NEXT PASS
0246|              BSRS4    CHKRW          ;GO DO AGAIN
0246| 49FA 0004          #        LEA     @1,A4
024A| 6068              #        BRA.S   CHKRW
024C| 4640              RWCHK2 NOT      D0          ;INVERT BACK TO ORIGINAL PATTERN
024E|              BSRS4    CHKRW          ;ONE MORE TIME
024E| 49FA 0004          #        LEA     @1,A4
0252| 6060              #        BRA.S   CHKRW
0254|              #@1
0254| E350              RWCHK3 ROXL     #1,D0      ;SET UP NEW PATTERN
0256| B3C8              CMPA.L  A0,A1          ;CHECK IF DONE
0258| 6704              BEQ.S   CHKBASE      ;IF YES GO CHECK FOR BASE REG TESTING
025A| D1CA              ADDA.L  A2,A0          ;ELSE BUMP MMU ADDR
025C| 60DE              BRA.S   RWCHK1

```

```

025E|
025E| B7C8          CHKBASE CMPA.L  A0,A3          ;DONE WITH BASE?
0260| 670A          BEQ.S   @2              ;EXIT IF YES
0262| 207C 0000 8008      MOVE.L  #MMUSADRB,A0      ;ELSE SET STARTING BASE ADDRESS
0268| 224B          MOVEA.L  A3,A1              ; AND ENDING ADDRESS
026A| 60D0          BRA.S   RWCHK1            ;GO CHECK BASE REGS
026C|
026C| 4A42          @2      TST     D2              ;check for errors
026E|              RTS6              ;and exit test
026E| 4ED6          #      JMP     (A6)
0270|
0270|              .PAGE
0270|              ;-----
0270|              ; Subroutine to do MMU address check
0270|              ; Leaves limit registers with invalid page value, base regs with 0
0270|              ;-----
0270|
0270| 207C 0000 8000      MMUACHK MOVE.L  #MMUSADRL,A0      ;SET MMU LIMIT START ADDR
0276| 227C 00FE 8000      MOVE.L  #MMUEADRL,A1      ;SET MMU LIMIT END ADDR
027C| 267C 00FE 8008      MOVE.L  #MMUEADRB,A3      ;SET MMU BASE END ADDR
0282| 383C 0C00          MOVE     #INVPAG,D4      ;SET FINAL VALUE FOR LIMIT REGS
0286|
0286| 3210          ACHK1  MOVE     (A0),D1          ;READ REG
0288| B141          EOR     D0,D1              ;CHECK IF EXPECTED
028A| 0241 0FFF          ANDI     #$0FFF,D1          ;MASK DON'T CARES
028E| 6620          BNE.S   MADRERR            ;EXIT IF ERROR
0290|
0290| 3084          MMUSET  MOVE     D4,(A0)          ;SET FINAL REG VALUE
0292| E350          ROXL    #1,D0              ;SET UP NEW PATTERN
0294| B3C8          CMPA.L  A0,A1              ;CHECK IF DONE
0296| 6704          BEQ.S   ACHK2            ;IF YES GO CHECK FOR BASE REG TESTING
0298| D1CA          ADDA.L  A2,A0              ;ELSE BUMP MMU ADDR
029A| 60EA          BRA.S   ACHK1
029C|
029C| B7C8          ACHK2  CMPA.L  A0,A3          ;DONE WITH BASE?
029E| 670C          BEQ.S   @2              ;EXIT IF YES
02A0| 207C 0000 8008      MOVE.L  #MMUSADRB,A0      ;ELSE SET STARTING BASE ADDRESS
02A6| 224B          MOVEA.L  A3,A1              ; AND ENDING ADDRESS
02A8| 7800          MOVEQ   #0,D4              ;SET FINAL VALUE FOR BASE REGS
02AA| 60DA          BRA.S   ACHK1            ;GO CHECK BASE REGS
02AC|
02AC| 4A42          @2      TST     D2              ;check for errors
02AE|              RTS6              ;and exit test
02AE| 4ED6          #      JMP     (A6)
02B0|              ; Handle MMU address error
02B0|
02B0| 8441          MADRERR OR     D1,D2          ;save error bits

```

```

02B2| 60DC          BRA.S   MMUSET          ; and continue test
02B4|
02B4|          .ELSE
02B4|          .ENDC
02B4|
02B4|          ;-----
02B4|          ; Subroutine to do MMU actual read/write
02B4|          ;-----
02B4|
02B4| 3080          CHKRW   MOVE    D0,(A0)          ;do write
02B6| 3210          MOVE    (A0),D1          ;read back
02B8| B141          EOR     D0,D1          ;compare
02BA| 0241 0FFF      ANDI    #$0FFF,D1          ;mask don't cares
02BE| 6602          BNE.S   RWERR          ;skip if error
02C0|          RTS4          ;else return
02C0| 4ED4          #        JMP      (A4)
02C2|
02C2|          ; Error collection
02C2|
02C2| 8441          RWERR   OR      D1,D2          ;save error bits
02C4|          RTS4          ;and return
02C4| 4ED4          #        JMP      (A4)
02C6|
02C6|          .PAGE
02C6|
02C6|          ;-----
02C6|          ; Now do setup of MMU supervisor registers for RAM and I/O space access.
02C6|          ; Also do read check after write and abort if error.
02C6|          ;-----
02C6|
02C6|          ; Do origin registers first
02C6|
02C6| 207C 0000 8008      SETMMU  MOVE.L  #MMUSADRB,A0      ;GET MMU PTR
02CC| 7000          MOVEQ   #0,D0          ;clear for use
02CE| 7200          MOVEQ   #0,D1
02D0| 3802          MOVE    D2,D4          ;SAVE PREVIOUS RESULTS IF ANY
02D2| 7400          MOVEQ   #0,D2
02D4| 7C00          MOVEQ   #0,D6
02D6| 247C 0002 0000      MOVE.L  #ADR128K,A2          ;ADDRESS INCREMENT
02DC| 267C 0000 0100      MOVE.L  #PAG128K,A3          ;SET UP BASE ADDRESS INCREMENT
02E2| 7C10          MOVEQ   #16,D6          ;LOOP COUNT
02E4|          LOADORG  BSR4     CHKRW          ;DO WRITE/READ CHECK
02E4| 49FA 0004          #        LEA      @1,A4
02E8| 60CA          #        BRA.S   CHKRW
02EA|          #@1
02EA| D08B          ADD.L    A3,D0          ;COMPUTE NEXT MEMORY BASE ADDRESS
02EC| D1CA          ADDA.L  A2,A0          ;BUMP MMU ORG PTR
02EE| 5346          SUBQ    #1,D6

```

```

02F0| 66F2                BNE.S   LOADORG       ;LOOP UNTIL DONE
02F2|
02F2|                ; Set base for I/O and special I/O space
02F2|
02F2| 207C 00FC 8008        MOVEA.L #MMU126B,A0    ;PT TO ORG REG 126
02F8| 7000                MOVEQ   #0,D0          ;set data value
02FA|                BSR4      CHKRW
02FA| 49FA 0004          #        LEA      @1,A4
02FE| 60B4                #        BRA.S   CHKRW
0300|                #@1
0300| D1CA                ADDA.L   A2,A0          ;BUMP PTR TO REG 127
0302|                BSR4      CHKRW
0302| 49FA 0004          #        LEA      @1,A4
0306| 60AC                #        BRA.S   CHKRW
0308|                #@1
0308|
0308|                ; Now do limit registers
0308|
0308| 207C 0000 8000        MOVEA.L #MMUSADRL,A0    ;GET MMU LIMIT REG PTR
030E| 303C 0700          MOVE      #MEMLMT,D0      ;LIMIT FOR 128K MEMORY SEGMENTS
0312| 7200                MOVEQ   #0,D1          ;use as working reg
0314| 7C10                MOVEQ   #16,D6         ;LOOP COUNT
0316|                LOADLMT BSR4      CHKRW
0316| 49FA 0004          #        LEA      @1,A4
031A| 6098                #        BRA.S   CHKRW
031C|                #@1
031C| D1CA                ADDA.L   A2,A0          ;BUMP MMU PTR
031E| 5346                SUBQ     #1,D6
0320| 66F4                BNE.S   LOADLMT       ;LOOP UNTIL DONE
0322|
0322|                ; Now do MMU limit reg setup for I/O and Special I/O access
0322|
0322| 207C 00FC 8000        MOVEA.L #MMU126L,A0    ;PT TO LMT REG 126
0328| 303C 0900          MOVE      #IOLMT,D0      ;SET FOR I/O SPACE, FULL ACCESS
032C|                BSR4      CHKRW
032C| 49FA 0004          #        LEA      @1,A4
0330| 6082                #        BRA.S   CHKRW
0332|                #@1
0332| D1CA                ADDA.L   A2,A0          ;BUMP PTR TO REG 127
0334| 303C 0F00          MOVE      #SPLMT,D0      ;SET FOR SPECIAL I/O, FULL ACCESS
0338|                BSR4      CHKRW
0338| 49FA 0006          #        LEA      @1,A4
033C| 6000 FF76          #        BRA.S   CHKRW
0340|                #@1
0340|
0340|                .IF   DIAGS = 1
0340|                ; Check if errors detected

```

```

0340| 4A42          TST      D2          ;CHECK ERROR MASK
0342| 6600 FE94     BNE      MMUERR      ;ABORT IF ERROR
0346| 3404          MOVE     D4,D2        ;ELSE RESTORE PREVIOUS RESULTS
0348|              .ENDC
0348|
0348|              .PAGE
0348|
0348|      ;-----
0348|      ; Complete testing of MMU by checking other context regs.
0348|      ; Uses reg D6 for context indicator.
0348|      ;-----
0348|              .IF      DIAGS = 1
0348|
0348|      MMUTST2 MOVEQ    #0,D6          ;FOR CONTEXT INDICATOR
034A|          BSR4      MMUINIT        ;REINITIALIZE FOR TESTING
034A| 49FA 0006     #        LEA        @1,A4
034E| 6000 FEC6     #        BRA        MMUINIT
0352|              #@1
0352|
0352| 4A39 00FC E00A     TST.B    SEG1ON      ;SET FOR CONTEXT 1
0358| 7C01          MOVEQ    #1,D6          ;SET CONTEXT INDICATOR
035A|
035A|          BSR4      CONCHK          ;CHECK IF MMU CONTEXT CHANGED
035A| 49FA 0006     #        LEA        @1,A4
035E| 6000 00B0     #        BRA        CONCHK
0362| 6700 0090     BEQ       MMUERR2        ;EXIT IF NO - SEG BIT ERROR
0366|
0366|          BSR6      MMURW          ;ELSE GO DO R/W TEST
0366| 4DFA 0006     #        LEA        @1,A6
036A| 6000 FEBE     #        BRA        MMURW
036E| 6600 0084     BNE      MMUERR2        ;exit if error
0372|
0372| 4A39 00FC E00E     TST.B    SEG2ON      ;SET FOR CONTEXT 3
0378| 7C03          MOVEQ    #3,D6
037A|
037A|          BSR4      CONCHK          ;CHECK IF MMU CONTEXT CHANGED
037A| 49FA 0006     #        LEA        @1,A4
037E| 6000 0090     #        BRA        CONCHK
0382|              #@1
0382| 6770          BEQ.S    MMUERR2        ;EXIT IF NO - SEG BIT ERROR
0384|
0384|          BSR6      MMURW          ;ELSE GO TEST
0384| 4DFA 0006     #        LEA        @1,A6
0388| 6000 FEA0     #        BRA        MMURW
038C|              #@1
038C| 6666          BNE.S    MMUERR2        ;exit if error
038E|
038E| 4A39 00FC E008     TST.B    SEG1OFF      ;SET FOR CONTEXT 2

```



```

0394| 7C02                MOVEQ    #2,D6
0396|
0396|                BSRS4    CONCHK    ;CHECK IF MMU CONTEXT CHANGED
0396| 49FA 0004            #        LEA      @1,A4
039A| 6074                #        BRA.S   CONCHK
039C|                #@1
039C| 675C                BEQ.S    MMUERR3    ;EXIT IF NO - SEG BIT ERROR
039E|
039E|                BSRS6    MMURW     ;ELSE GO TEST
039E| 4DFA 0006            #        LEA      @1,A6
03A2| 6000 FE86            #        BRA.S   MMURW
03A6|                #@1
03A6| 6652                BNE.S    MMUERR3    ;exit if error
03A8|
03A8| 4A39 00FC E00C        TST.B    SEG2OFF    ;RESET FOR CONTEXT 0 REGS
03AE|
03AE|                ; Now do MMU addressing check of remaining context regs
03AE|
03AE|                BSR4     MMUINIT    ;REINITIALIZE
03AE| 49FA 0006            #        LEA      @1,A4
03B2| 6000 FE62            #        BRA     MMUINIT
03B6|                #@1
03B6|
03B6| 4A39 00FC E00A        TST.B    SEG1ON     ;SET FOR CONTEXT 1
03BC| 7C01                MOVEQ    #1,D6
03BE|                BSR6     MMUACHK    ;TEST CONTEXT 1
03BE| 4DFA 0006            #        LEA      @1,A6
03C2| 6000 FEAC            #        BRA     MMUACHK
03C6|                #@1
03C6| 662C                BNE.S    MMUERR2    ;exit if error
03C8| 4A39 00FC E00E        TST.B    SEG2ON     ;TEST CONTEXT 3
03CE| 7C03                MOVEQ    #3,D6
03D0|                BSR6     MMUACHK
03D0| 4DFA 0006            #        LEA      @1,A6
03D4| 6000 FE9A            #        BRA     MMUACHK
03D8| 661A                BNE.S    MMUERR2    ;exit if error
03DA|
03DA| 4A39 00FC E008        TST.B    SEG1OFF    ;TEST CONTEXT 2
03E0| 7C02                MOVEQ    #2,D6
03E2|                BSR6     MMUACHK
03E2| 4DFA 0006            #        LEA      @1,A6
03E6| 6000 FE88            #        BRA     MMUACHK
03EA|                #@1
03EA| 660E                BNE.S    MMUERR3    ;exit if error
03EC|
03EC| 4A39 00FC E00C        TST.B    SEG2OFF    ;RESET TO CONTEXT 0
03F2| 6014                BRA.S    MMULPCHK    ;go check for loop mode

```

```

03F4|
03F4| 4A39 00FC E008      MMUERR2 TST.B   SEG1OFF      ;ENSURE RESET FOR CONTEXT 0
03FA| 4A39 00FC E00C      MMUERR3 TST.B   SEG2OFF
0400|
0400| E85E                ROR      #4,D6      ;get context indicator
0402| 8446                OR       D6,D2      ;save with error bits (if any)
0404| 08C7 0000            BSET     #MMU,D7    ;set error indicator
0408|      MMULPCHK
0408| 4A87                TST.L   D7          ;in loop mode?
040A| 6B00 FDA4            BMI.S   MMUTST     ;restart full MMU test if yes
040E| 6030                BRA.S   START     ;else continue to next test
0410|
0410|      .PAGE
0410|
0410|      ;-----
0410|      ; Subroutine to verify context change made - does comparison to ensure
0410|      ; destruction of context 0 mapping avoided. Zero bit set if error.
0410|      ;-----
0410|
0410| 3839 00FC 8000        CONCHK  MOVE    MMU126L,D4    ;check limit reg for I/O space
0416| 0244 0FFF            ANDI     #$0FFF,D4    ;mask don't care
041A| 0C44 0900            CMPI     #IOLMT,D4    ;still in same context?
041E| 661E                BNE.S   CONOK        ;exit if not
0420|
0420| 3839 00FE 8000        MOVE     MMU127L,D4    ;else also check reg for ROM space
0426| 0244 0FFF            ANDI     #$0FFF,D4    ;mask don't care
042A| 0C44 0F00            CMPI     #SPLMT,D4    ;also set up?
042E| 660E                BNE.S   CONOK        ;exit if not
0430|
0430| 3839 0000 8000        MOVE     MMU0L,D4     ;else do final check on reg for memory access
0436| 0244 0FFF            ANDI     #$0FFF,D4
043A| 0C44 0700            CMPI     #MEMLMT,D4    ;return with match results to caller
043E|
043E|      CONOK  RTS4
043E| 4ED4      #          JMP      (A4)
0440|
0440|      .ENDC
0440|      .IF ROM4K = 0
0440|      .IF DIAGS = 0
0440|      .ENDC      ;{DIAGS}
0440|      .ENDC      ;{ROM4K}
0440|      .PAGE
0440|
0440|      ;-----
0440|      ; Reset SETUP bit to enable system access and continue with testing
0440|      ;-----
0440|
0440| 4239 00FC E012        START   CLR.B   SETUP      ;TURN OFF SETUP TO ENTER MAP LAND ...
0446|
0446|      ;-----

```

```

0446|                ; Now do memory sizing - assumes 128K minimum memory increment
0446|                ; Register usage:
0446|                ;     A0 = minimum physical address          D0 = scratch use
0446|                ;     A1 = maximum physical address/scratch D1 = incr for search address
0446|                ;     A2 = unused                            D2 = unused
0446|                ;     A3 = next base memory addr to test     D3 = inverted sizing test pattern
0446|                ;     A4 = return address                     D4 = sizing test pattern
0446|                ;     A5 = unused                            D5 = retry count
0446|                ;     A6 = saved error mask                   D6 = error mask
0446|                ;-----
0446|
0446| 4280          MEMSIZ CLR.L   D0                ;setup regs for loop
0448| 2040          MOVE.L  D0,A0
044A| 2240          MOVE.L  D0,A1
044C| 2640          MOVE.L  D0,A3
044E| 2C40          MOVE.L  D0,A6
0450| 7202          MOVEQ   #2,D1                ;size at 128K boundaries      RM000
0452| 4841          SWAP    D1                    ;                                RM000
0454| 283C AA55 A55A MOVE.L  #PATRN,D4            ;set test patterns for sizing    CHG002
045A| 3604          MOVE     D4,D3                ;use only lower word
045C| 4643          NOT      D3                    ;and its inverse
045E|
045E|          CHKLO   BSR4    CHKMEM            ;first search for low memory address
045E| 49FA 0006      #        LEA     @1,A4
0462| 6000 00A4      #        BRA     CHKMEM
0466|          #@1
0466| 4A46          TST      D6                    ;memory found?
0468| 6752          BEQ.S    SAVELO                ;yes - go save address
046A| 4646          NOT      D6                    ;else invert to check if all bits in error
046C| 4A46          TST      D6                    ;if not, assume memory error
046E| 6648          BNE.S    @3                    ; and go save address
0470| D7C1          @2      ADDA.L  D1,A3            ;else bump search address
0472| 224B          MOVEA.L  A3,A1                ;set as next working address
0474| B3FC 0020 0000 CMPA.L  #MAXADR,A1            ;at max address?
047A| 66E2          BNE.S    CHKLO                ;continue search if not
047C|
047C|                ; No memory found - toggle LED and check for I/O board; if no I/O (bus error)    CHG004
047C|                ; diagnostics are restarted          CHG004
047C|
047C| 13FC 00AF 00FC E800 MOVE.B  #DEFVID2,VIDLTCH ;set LED on and default video latch setting (PAGE 2F) CHG004
0484| 303C 61A8      MOVE     #TINTHSEC,D0        ;delay for .1 sec          CHG004
0488| 5340          @9      SUBQ    #1,D0          ;                                CHG004
048A| 66FC          BNE.S    @9                    ;                                CHG004
048C| 13FC 002F 00FC E800 MOVE.B  #DEFVID,VIDLTCH ;reset LED and leave video latch setting    CHG004
0494| 207C 00FC DD81 MOVE.L   #VIA1BASE,A0        ;check for I/O board          CHG004
049A| 4A10          TST.B    (A0)                  ;bus error will occur if not installed    CHG004
049C|

```

```

049C|                ; Go into read/write loop if no memory found but I/O installed
049C|
049C|                BSR2    LOTONE        ;beep speaker for error
049C| 45FA 0006        #        LEA        @1,A2
04A0| 6000 00B2        #        BRA        LOTONE
04A4|                #@1
04A4|                BSR4    CONSET        ;set contrast
04A4| 49FA 0006        #        LEA        @1,A4
04A8| 6000 0350        #        BRA        CONSET
04AC|                #@1
04AC| 207C 000F FFFE        MOVE.L    #ONEMEG-2,A0        ;set default memory address (to span both boards)    CHG002
04B2| 2084                @4        MOVE.L    D4,(A0)        ;go into read/write loop    CHG002
04B4| 2610                MOVE.L    (A0),D3        ;
04B6| 60FA                BRA.S    @4
04B8|
04B8|                ; Low memory address found - save and continue
04B8| 4646                @3        NOT        D6        ;reinvert and
04BA| 3C46                MOVE        D6,A6        ; save results
04BC|
04BC| 204B                SAVELO    MOVEA.L    A3,A0        ;save low address
04BE| B1FC 0010 0000        CMPA.L    #ONEMEG,A0        ;check for min low address
04C4| 6F06                BLE.S    @1        ;skip if OK
04C6| 207C 0010 0000        MOVEA.L    #ONEMEG,A0        ;else set at min value (one 512K board in slot 1)
04CC|
04CC|                .PAGE
04CC|                ;-----
04CC|                ; Now check for high memory address; search to max address of 2 meg
04CC|                ;-----
04CC|
04CC| 244B                @1        MOVEA.L    A3,A2        ;save low address as first high address
04CE|
04CE|                TSTHI
04CE| D7C1                ADDA.L    D1,A3        ;compute next 128K increment
04D0| 224B                MOVEA.L    A3,A1        ;use as new search value
04D2| B3FC 0020 0000        CMPA.L    #MAXADR,A1        ;done?
04D8| 6728                BEQ.S    SIXXIT
04DA|
04DA|                ; Following patch added to detect for wraparound problem of old memory boards
04DA| 2008                MOVE.L    A0,D0        ;check low address    RM015
04DC| 6604                BNE.S    CHKHI        ;old memory boards start at address 0    RM015
04DE| B651                CMP        (A1),D3        ;are we wrapped back to already tested location?
04E0| 671E                BEQ.S    WRAPXIT        ;skip if yes (i.e., old memory board)    RM015
04E2|
04E2|                ; Else continue with check for high address
04E2|
04E2|                CHKHI    BSRS4    CHKMEM        ;go do memory search
04E2| 49FA 0004        #        LEA        @1,A4

```

```

04E6| 6020      #      BRA.S   CHKMEM
04E8|          #@1
04E8| 4A46      TST      D6          ;any errors?
04EA| 670E      BEQ.S    SAVEHI     ;skip if not to save address
04EC| 4646      NOT      D6          ;else invert to see if all bits in error
04EE| 4A46      TST      D6
04F0| 67DC      BEQ.S    TSTHI     ;skip if yes to ignore address
04F2| 300E      MOVE     A6,D0      ;else get previous results
04F4| 4646      NOT      D6          ;reinvert and
04F6| 8046      OR       D6,D0      ; add new results
04F8| 3C40      MOVE     D0,A6      ;save for later
04FA|
04FA| 244B      SAVEHI   MOVEA.L A3,A2 ;save as new potential high address
04FC| 4253      CLR      (A3)        ;clear test pattern
04FE| 60CE      BRA.S    TSTHI     ; and continue loop
0500|
0500| 4251      WRAPXIT  CLR      (A1) ;clear test pattern
0502|
0502| D5C1      SIZXIT   ADDA.L D1,A2  ;high address = last valid addr + 128K
0504| 224A      MOVEA.L  A2,A1      ;save for later use
0506| 605E      BRA.S    RSTMMU     ;continue on
0508|
0508|          .PAGE
0508|
0508|          ;-----
0508|          ; Subroutine to do memory check for sizing.  If error, tries successive
0508|          ; memory locations up to retry count (D5).  Returns with error mask in D6.
0508|          ;-----
0508|
0508| 7A20      CHKMEM   MOVEQ   #RETRYCNT,D5 ; set retry count in case of errors
050A| 4246      CLR      D6          ; clear for error mask
050C| 3284      @1      MOVE     D4,(A1) ; check if true data stores
050E| 3343 0002    MOVE     D3,2(A1) ; and try complement to next location
0512| B851      CMP      (A1),D4
0514| 6706      BEQ.S    @2          ; continue if yes
0516| 3011      MOVE     (A1),D0      ; else get error bits
0518| B940      EOR      D4,D0
051A| 8C40      OR       D0,D6      ; and save them
051C|
051C| B669 0002    @2      CMP      2(A1),D3 ; check second location
0520| 6708      BEQ.S    @3          ; exit if data correct
0522| 3029 0002    MOVE     2(A1),D0 ; else read again
0526| B740      EOR      D3,D0      ; get error bits
0528| 8C40      OR       D0,D6      ; save in error mask
052A|
052A| 3344 0002    @3      MOVE     D4,2(A1) ; now try in reverse order
052E| 3283      MOVE     D3,(A1)
0530| B869 0002    CMP      2(A1),D4 ; check second location

```

```

0534| 6708                BEQ.S   @4           ; skip if OK
0536| 3029 0002          MOVE    2(A1),D0       ; else save error bits
053A| B940                EOR     D4,D0
053C| 8C40                OR      D0,D6
053E|
053E| B651                @4      CMP     (A1),D3   ; and check first
0540| 6706                BEQ.S   @5           ; continue if yes
0542| 3011                MOVE    (A1),D0       ; else get error bits
0544| B740                EOR     D3,D0
0546| 8C40                OR      D0,D6           ; and save them
0548|
0548| 4A46                @5      TST     D6         ; any errors?
054A| 6706                BEQ.S   @6           ; skip if no
054C| 4A99                TST.L   (A1)+        ; else bump search address to next pair
054E| 5345                SUBQ    #1,D5         ; decr retry count
0550| 66BA                BNE.S   @1           ; continue until count exhausted
0552|
0552|                    @6      RTS4           ; return with results in D6
0552| 4ED4                #      JMP     (A4)
0554|
0554|                    .PAGE
0554|                    ;-----
0554|                    ; Subroutine to set parms for lo tone from speaker
0554|                    ;-----
0554|
0554| 7060                LOTONE  MOVEQ   #$60,D0     ;set frequency
0556| 323C 00FA            MOVE    #250,D1         ;and duration
055A| 7404                MOVEQ   #4,D2           ;and volume (medium)
055C|                    BSR4    TONE2         ;go do tone
055C| 49FA 0006            #      LEA     @1,A4
0560| 6000 05A4            #      BRA     TONE2
0564|                    #@1
0564|                    RTS2
0564| 4ED2                #      JMP     (A2)
0566|
0566|
0566|                    .PAGE
0566|                    ;-----
0566|                    ; Now reset MMU regs according to low and high physical address
0566|                    ; Remainder of MMU regs in context 0 are set to invalid page
0566|                    ;
0566|                    ; Register Usage:
0566|                    ;   A0 - low physical address      D0 - scratch/value stored in base reg
0566|                    ;   A1 - high physical address     D1 - value stored in limit reg
0566|                    ;   A2 - MMU base reg ptr         D2 - unused
0566|                    ;   A3 - MMU limit reg ptr        D3 - holds base reg page incr value
0566|                    ;   A4 - used for return address   D4 - used for count of regs to set

```

```

0566|          ;      A5 - MMU address increment      D5 - low physical page
0566|          ;      A6 - not used                    D6 - high physical page
0566|          ;-----
0566|
0566|          ; First translate memory addresses to 512 byte page values for MMU use
0566|
0566|      RSTMMU
0566| 2A08          MOVE.L  A0,D5          ;GET MEMORY ADDRESS VALUES
0568| 2C09          MOVE.L  A1,D6
056A| 7009          MOVEQ   #9,D0          ;SET SHIFT COUNT
056C| E0AD          LSR.L   D0,D5          ;TRANSLATE TO PAGE VALUES
056E| E0AE          LSR.L   D0,D6
0570|
0570|          ; Now initialize for MMU write operations
0570|
0570| 247C 0000 8008      MOVEA.L #MMUSADRB,A2      ;SET MMU BASE REG PTR
0576| 267C 0000 8000      MOVEA.L #MMUSADRL,A3      ;SET LIMIT REG PTR
057C| 2A7C 0002 0000      MOVE.L  #ADR128K,A5      ;SET INCREMENT VALUE FOR MMU ADDRESSES
0582| 263C 0000 0100      MOVE.L  #PAG128K,D3      ;SET INCR VALUE FOR BASE REG CONTENTS
0588| 787E          MOVEQ   #126,D4      ;SET REG COUNT - NO RESETTNG OF REGS 126,127
058A| 3005          MOVE    D5,D0          ;SET BASE VALUE
058C| 323C 0700      MOVE    #MEMLMT,D1      ;SET LIMIT VALUE
0590|
0590|          ; Remap MMU regs for existing memory
0590|
0590|      REMAP  BSRS4  WRTMMU          ;REWRITE SET OF MMU REGS
0590| 49FA 0004      #      LEA      @1,A4
0594| 6024          #      BRA.S    WRTMMU
0596|          #@1
0596| 5344          SUBQ    #1,D4          ;DECR REG COUNT
0598| D083          ADD.L   D3,D0          ;BUMP PAGE ADDRESS
059A| BC80          CMP.L   D0,D6          ;CHECK IF AT UPPER LIMIT
059C| 66F2          BNE.S   REMAP          ;LOOP IF NO
059E|
059E|          ; Now map remainder of regs for invalid access
059E|
059E| 4240          CLR     D0          ;SET NEW BASE REG VALUE
05A0| 323C 0C00      MOVE    #INVPAG,D1      ; AND LIMIT REG VALUE
05A4|          MAPINV BSRS4  WRTMMU          ;GO DO REWRITE
05A4| 49FA 0004      #      LEA      @1,A4
05A8| 6010          #      BRA.S    WRTMMU
05AA|          #@1
05AA| 5384          SUBQ.L  #1,D4          ;DECR COUNT
05AC| 66F6          BNE.S   MAPINV          ;LOOP UNTIL DONE
05AE|
05AE|          ; Finally reset video page to last page of memory
05AE|

```

```

05AE| EC8E                LSR.L   #6,D6           ;compute address for video page
05B0| 5346                SUBQ    #1,D6           ;decr to last page
05B2| 13C6 00FC E800      MOVE.B  D6,VIDLTCH       ;and set video latch
05B8|
05B8| 6066                BRA.S   MEMTST1         ;SKIP TO NEXT TEST
05BA|                    .IF     EXTERNAL = 1
05BA|                    .ENDC
05BA|
05BA|
05BA|                    ;-----
05BA|                    ; Subroutine to set MMU regs (context 0) - can also be called by external
05BA|                    ; routines that provide the following register inputs:
05BA|                    ;
05BA|                    ;     A2 = base reg address           D0 = value for base reg
05BA|                    ;     A3 = limit reg address          D1 = value for limit reg
05BA|                    ;     A5 = reg address increment
05BA|                    ;-----
05BA|
05BA| 4A39 00FC E010      WRTMMU  TST.B   SETUPON        ;TURN SETUP ON TO ENABLE MMU ACCESS
05C0| 3480                MOVE    D0,(A2)           ;SET BASE REG
05C2| 3681                MOVE    D1,(A3)           ;SET LIMIT REG
05C4| D5CD                ADDA.L  A5,A2             ;BUMP MMU PTRS
05C6| D7CD                ADDA.L  A5,A3
05C8| 4239 00FC E012      CLR.B   SETUP            ;BACK TO MAP LAND
05CE|                    RTS4                      ;AND BACK TO CALLER
05CE| 4ED4                #      JMP      (A4)
05D0|
05D0|                    .IF  ROM16K = 1
05D0|                    ;-----
05D0|                    ; Subroutine to read MMU regs - for call by external routines.
05D0|                    ; Inputs:
05D0|                    ;     D2 = context to read (0-3)
05D0|                    ;     A2 = base reg address
05D0|                    ;     A3 = limit reg address
05D0|                    ;     A4 = return address
05D0|                    ;     A5 = reg address increment
05D0|                    ; Outputs:
05D0|                    ;     D0 = value of base reg
05D0|                    ;     D1 = value of limit reg
05D0|                    ;     A2,A3 incremented by value in A5
05D0|                    ; Side Effects:
05D0|                    ;     D3 trashed
05D0|                    ;-----
05D0|
05D0| 363C 0FFF      READMMU  MOVE    #$0FFF,D3         ;set mask for result
05D4| 4A39 00FC E010      TST.B   SETUPON        ;turn setup on to enable MMU access
05DA|
05DA| 4A42                TST      D2                ;check context

```



```

05DC| 6722                BEQ.S    @9                ;skip if context 0
05DE| 0C02 0001          CMP.B    #1,D2            ;context 1?
05E2| 6716                BEQ.S    @2                ;
05E4| 0C02 0002          CMP.B    #2,D2            ;context 2?
05E8| 6708                BEQ.S    @1                ;
05EA| 4A39 00FC E00E      TST.B    SEG2ON           ;must be context 3
05F0| 6008                BRA.S    @2                ;set both seg bits
05F2| 4A39 00FC E00E      @1    TST.B    SEG2ON           ;set for context 2
05F8| 6006                BRA.S    @9                ;
05FA| 4A39 00FC E00A      @2    TST.B    SEG1ON           ;set for context 1 and 3
0600|
0600|                ; read the regs
0600|
0600| 3012                @9    MOVE     (A2),D0         ;read base reg
0602| C043                AND      D3,D0             ;clear junk
0604| 3213                MOVE     (A3),D1             ;read limit reg
0606| C243                AND      D3,D1             ;clear junk
0608| D5CD                ADDA.L   A5,A2             ;incr ptrs
060A| D7CD                ADDA.L   A5,A3
060C| 4A39 00FC E008      TST.B    SEG1OFF           ;restore to context 0
0612| 4A39 00FC E00C      TST.B    SEG2OFF           ;
0618| 4239 00FC E012      CLR.B    SETUP            ;back to map land
061E|                    RTS4                      ;and back to caller
061E| 4ED4                #      JMP      (A4)
0620|
0620|                .ENDC
0620|                .IF      EXTERNAL = 1
0620|                .ENDC
0620|
0620|                .PAGE
0620|                ;-----
0620|                ; Begin memory testing by checking first 800 hex locations (2K).
0620|                ; If error here, abort other testing and go into loop since can't relay
0620|                ; meaningful results.
0620|                ;-----
0620|
0620| 2448                MEMTST1 MOVEA.L A0,A2          ;save memory lo, hi addresses
0622| 2649                MOVEA.L A1,A3
0624|
0624|                .IF      DIAGS = 1
0624|
0624| 91C8                SUBA.L   A0,A0                ;set test addresses
0626| 327C 0800          MOVEA   #LOMEM,A1            ;upper address                RM000
062A|                    BSR4     RAMTEST              ;go do memory test
062A| 49FA 0006          #      LEA      @1,A4
062E| 6000 0880          #      BRA      RAMTEST
0632|                    #@1

```

```

0632| 6738                BEQ.S    INITMEM        ;skip if OK
0634|
0634|                ; Error in low memory - reset video latch, beep speaker and go into R/W loop
0634|
0634| 200A                MOVE.L    A2,D0            ;get low physical address
0636| E088                LSR.L     #8,D0            ;convert to page value
0638| EE88                LSR.L     #7,D0
063A| 13C0 00FC E800      MOVE.B    D0,VIDLTCH      ;set video latch to bad area
0640|                BSR4      CONSET      ;ensure contrast on
0640| 49FA 0006            #          LEA      @1,A4
0644| 6000 01B4            #          BRA      CONSET
0648|                #@1
0648|
0648|                BSR2      LOTONE        ;beep speaker twice for low memory error
0648| 45FA 0006            #          LEA      @1,A2
064C| 6000 FF06            #          BRA      LOTONE
0650|                #@1
0650| 303C 61A8            MOVE      #TNTHSEC,D0      ;delay for about 1/10 sec                RM000
0654| 5340                TONEDLY SUBQ      #1,D0
0656| 66FC                BNE.S     TONEDLY
0658|                BSR2      LOTONE        ;do second beep
0658| 45FA 0006            #          LEA      @1,A2
065C| 6000 FEF6            #          BRA      LOTONE
0660|                #@1
0660|
0662| 303C A55A            MOVE      #PATRN2,D0      ;set pattern for usess
0666| 3080                @2      MOVE      D0,(A0)
0668| 3210                MOVE      (A0),D1
066A| 60FA                BRA.S     @2            ;loop with random display on screen
066C|
066C|                .ELSE
066C|                .ENDC                ;{DIAGS}
066C|
066C|                ; Now attempt to initialize status areas and save results
066C|
066C| 307C 0180            INITMEM MOVEA     #STATUS,A0      ;get ptr to start of status area                RM000
0670| 707F                MOVEQ      #127,D0          ;set count
0672| 4298                @2      CLR.L     (A0)+        ;clear it
0674| 51C8 FFFC            DBF       D0,@2            ;until done
0678|
0678| 31C3 0186            MOVE      D3,MEMRSLT      ;save test results
067C| 31CE 0184            MOVE      A6,SIZRSLT      ;save sizing results
0680| 21CA 02A4            MOVE.L    A2,MINMEM        ;save min memory address
0684| 21CB 0294            MOVE.L    A3,MAXMEM        ;save max memory address
0688| 97CA                SUBA.L     A2,A3            ;compute total memory
068A| 21CB 02A8            MOVE.L    A3,TOTMEM        ;and save also
068E|

```

```

068E| 207C 0000 8000          MOVE.L  #HEX32K,A0      ;compute base address for screen
0694| 97C8                    SUBA.L   A0,A3
0696| 21CB 0110          MOVE.L   A3,SCRNBASE      ;and save
069A|
069A| 31C2 01B0          MOVE     D2,MMURSLT      ;and save MMU results also
069E|
069E| 21FC 0000 02B0 0260      MOVE.L   #KBDQ,KBDQPTR      ;init COPS buffer pointer for later use
06A6|
06A6|          .PAGE
06A6|          ;-----
06A6|          ; Initialize exception and trap vectors to catch unexpected errors
06A6|          ;-----
06A6|
06A6| 6104          INITVCT BSR.S   SETVCTRS      ;init vectors
06A8| 6000 00CA          BRA      SCCSET      ;continue testing
06AC|
06AC|          ; Subroutine to set up default vectors
06AC|
06AC|          SETVCTRS
06AC| 41FA 003E          LEA      MISC,A0
06B0| 93C9          SUBA.L   A1,A1
06B2| 7040          MOVEQ    #64,D0
06B4| 22C8          @1      MOVE.L   A0,(A1)+      ; fill with unknown ones
06B6| 5340          SUBQ     #1,D0
06B8| 6EFA          BGT      @1
06BA| 6126          BSR.S   SETBUSVCT      ; then, with special ones
06BC| 41FA 0090          LEA      AERR,A0
06C0| 21C8 000C          MOVE.L   A0,ADRVCTR
06C4| 41FA 0032          LEA      IERR,A0
06C8| 21C8 0010          MOVE.L   A0,ILLVCTR
06CC| 41FA 0036          LEA      NMI,A0
06D0| 21C8 007C          MOVE.L   A0,NMIVCT
06D4| 41FA 0060          LEA      TRPERR,A0      ; same routine for line 1010 and 1111
06D8| 21C8 0028          MOVE.L   A0,L10VCTR
06DC| 21C8 002C          MOVE.L   A0,L11VCTR
06E0|
06E0|          .IF  USERINT = 0
06E0|          .ENDC
06E0|
06E0| 4E75          RTS
06E2|          ;-----
06E2|          ; Subroutine to setup bus error vector
06E2|          ;-----
06E2|
06E2|          SETBUSVCT
06E2| 47FA 005E          LEA      BERR,A3      ;setup default vector
06E6| 21CB 0008          MOVE.L   A3,BUSVCTR      ;

```

```

06EA| 4E75                RTS                ;                RM000
06EC|
06EC|                ;-----
06EC|                ; Exception Handler routines
06EC|                ;-----
06EC|
06EC| 21C7 01AC            MISC    MOVE.L  D7,D7SAV        ;save incoming value
06F0| 7E00                MOVEQ   #0,D7
06F2| 08C7 0007            BSET    #MISEXCP,D7        ;set error indicator
06F6| 606C                BRA.S    EXCP1
06F8|
06F8| 21C7 01AC            IERR    MOVE.L  D7,D7SAV        ;save incoming value
06FC| 7E00                MOVEQ   #0,D7
06FE| 08C7 0008            BSET    #ILLEXCP,D7        ;set error indicator
0702| 6060                BRA.S    EXCP1
0704|
0704| 21C7 01AC            NMI      MOVE.L  D7,D7SAV        ;save incoming value
0708| 7E00                MOVEQ   #0,D7
070A| 6100 085C            BSR      TSTSTAT        ;check status reg for parity error
070E| 6620                BNE.S    NOTPE        ;skip if not
0710|
0710| 08C7 0016                BSET    #MPAR,D7        ;set error indicator
0714| 6100 08DA            BSR      GETPADDD        ;get and save error address
0718| 4A39 00FC E01C        TST.B    PAROFF        ;toggle to clear error bit
071E| 0801 0005            BTST     #5,D1        ;video error?
0722| 6706                BEQ.S    @1        ;skip if not
0724| 0281 FFFF 8000        ANDI.L  #VMSK,D1        ;mask if yes
072A| 21C1 01A6            @1      MOVE.L  D1,PEADDR        ;save converted error address
072E| 6034                BRA.S    EXCP1        ;go to exit
0730|
0730| 08C7 0004            NOTPE    BSET    #CPUINTR,D7        ;else set NMI code
0734| 602E                BRA.S    EXCP1        ; and exit
0736|
0736| 21C7 01AC            TRPERR   MOVE.L  D7,D7SAV        ;save incoming value
073A| 7E00                MOVEQ   #0,D7
073C| 08C7 0009            BSET    #TRPEXCP,D7        ;set error indicator
0740| 6022                BRA.S    EXCP1
0742|
0742| 21C7 01AC            BERR      MOVE.L  D7,D7SAV        ;save incoming value
0746| 7E00                MOVEQ   #0,D7
0748| 08C7 0005            BSET    #BUSEXCP,D7        ;set error indicator
074C| 600A                BRA.S    EXCP0
074E| 21C7 01AC            AERR      MOVE.L  D7,D7SAV        ;save incoming value
0752| 7E00                MOVEQ   #0,D7
0754| 08C7 0006            BSET    #ADREXCP,D7        ;set error indicator
0758|
0758|                EXCP0                ; GROUP 0 EXCEPTIONS HERE

```

```

0758| 31DF 0280          MOVE    (SP)+,EXCFC      ; SAVE THE EXTRA DATA
075C| 21DF 0282          MOVE.L  (SP)+,EXCADR
0760| 31DF 0286          MOVE    (SP)+,EXCIR
0764|
0764|          EXCP1          ; GROUP 1 EXCEPTIONS HERE
0764| 31DF 0288          MOVE    (SP)+,EXCSR      ; SAVE COMMON INFO
0768| 21DF 028A          MOVE.L  (SP)+,EXCPC
076C| 31C0 028E          MOVE    D0,EXCTYPE      ; save error type
0770| 6000 0C28          BRA      TSTCHK          ; and go display error
0774|
0774|          .PAGE
0774|          ;-----
0774|          ; Initialize SCC chip for Applebus use.                      CHG027
0774|          ; Bus error vector setup in case of problems.
0774|          ;-----
0774|          SCCSET
0774| 47FA 03E4          LEA      NOIO,A3          ;set bus error vector in case no IO board  CHG027
0778| 21CB 0008          MOVE.L  A3,BUSVCTR      ;                      CHG027
077C| 6100 0952          BSR      RSTSCC          ;go do setup                      CHG027
0780|
0780|          ;-----
0780|          ; Now test VIA for parallel port and contrast latch.
0780|          ; A read/write test on the timer 1 latches is done, then contrast
0780|          ; is set if OK.
0780|          ;-----
0780|          VIA2TST
0780|          .IF  DIAGS = 1
0780|          VIA2CHK
0780|          .IF  ROM16K = 1
0780| 47FA 002C          LEA      VIA2VCT,A3          ;OK - set up special bus vector
0784| 21CB 0008          MOVE.L  A3,BUSVCTR
0788|
0788| 207C 00FC D931      MOVE.L  #<VIA2BASE+T1LL2>,A0 ;set base address of timer low latch
078E| 7008              MOVEQ    #8,D0              ;set offset to high latch
0790|              BSRS6    VIATST          ;and go do test
0790| 4DFA 0004          #          LEA      @1,A6
0794| 6022              #          BRA.S    VIATST
0796|              #@1
0796| 670A              BEQ.S    @2              ;if OK, continue
0798| 08C7 000B          BSET     #VIA2,D7          ;else set error bit
079C| 4A87              TST.L    D7              ;check if in loop mode
079E| 6BE0              BMI.S    VIA2CHK          ;restart if yes
07A0| 600A              BRA.S    @3              ;else skip contrast setting
07A2|              .ENDC
07A2|

```

```

07A2| 4A87          @2      TST.L   D7              ;in loop mode?
07A4| 6BDA          BMI.S   VIA2CHK          ;restart if yes
07A6|              BSRS4    CONOFF          ;go turn off contrast
07A6| 49FA 0004      #        LEA      @1,A4
07AA| 6054          #        BRA.S    CONOFF
07AC|              #@1
07AC|
07AC|              .ENDC                      ;{DIAGS}
07AC|
07AC| 6074          @3      BRA.S    SCRNTST          ;else skip to next test
07AE|
07AE|              .IF   DIAGS = 1
07AE|              ;-----
07AE|              ;   Bus error handler for VIA #2 use
07AE|              ;-----
07AE|
07AE| 7033          VIA2VCT MOVEQ    #EVIA2,D0          ;SET ERROR CODE
07B0| 08C7 000B      BSET     #VIA2,D7          ;set indicator
07B4| 6000 0162      BRA      IOVCT          ;AND GO HANDLE I/O EXCEPTION
07B8|              .ENDC                      ;{DIAGS}
07B8|
07B8|              .IF   ROM16K = 1
07B8|              .PAGE
07B8|              ;-----
07B8|              ;   Subroutine to do VIA testing
07B8|              ;   A0 = address of first timer latch
07B8|              ;   D0 = offset to other latch
07B8|              ;-----
07B8|
07B8| 2248          VIATST  MOVE.L   A0,A1              ;set up address for second latch
07BA| D3C0          ADDA.L   D0,A1
07BC| 7000          MOVEQ    #0,D0              ;for error use
07BE| 4202          CLR.B    D2              ;clear old data value
07C0| 4210          MOVE.B   #0,(A0)          ;and the timer latches
07C2| 4211          MOVE.B   #0,(A1)
07C4|
07C4| 363C 00FF      MOVE     #$FF,D3          ;set up start value
07C8|              BSRS4    VIARW          ;go do read/write test
07C8| 49FA 0004      #        LEA      @1,A4
07CC| 6002          #        BRA.S    VIARW
07CE|              #@1
07CE|              RTS6              ;and return
07CE| 4ED6          #        JMP      (A6)
07D0|
07D0|              ;   Subroutine to do read/write test - loops thru all 256 values
07D0|
07D0|  B410          VIARW  CMP.B    (A0),D2          ;check for old values first

```

```

07D2| 6620          BNE.S   VIAFAIL
07D4| B411          CMP.B   (A1),D2
07D6| 661C          BNE.S   VIAFAIL
07D8|
07D8| 1083          MOVE.B   D3,(A0)          ;set new value in low timer latch
07DA| B411          CMP.B   (A1),D2          ;ensure high latch not affected
07DC| 6616          BNE.S   VIAFAIL
07DE| B610          CMP.B   (A0),D3          ;verify new low latch setting
07E0| 6612          BNE.S   VIAFAIL
07E2|
07E2| 1283          MOVE.B   D3,(A1)          ;set new value in high timer latch
07E4| B610          CMP.B   (A0),D3          ;ensure low latch not affected
07E6| 660C          BNE.S   VIAFAIL
07E8| B611          CMP.B   (A1),D3          ;verify new high latch setting
07EA| 6608          BNE.S   VIAFAIL
07EC|
07EC| 1403          MOVE.B   D3,D2          ;the new value becomes the old
07EE| 51CB FFE0      DBF     D3,VIARW        ;loop thru all 256 values
07F2| 6002          BRA.S   VIARWEND
07F4|
07F4| 5240          VIAFAIL ADDQ   #1,D0          ;set for error
07F6|
07F6| 4A40          VIARWEND TST   D0          ;set zero bit indicator
07F8|              RTS4
07F8| 4ED4          #        JMP     (A4)
07FA|              .ENDC
07FA|              .PAGE
07FA|
07FA| ;-----
07FA| ; Subroutine to set contrast latch - sets for default, off or value in D0
07FA| ;-----
07FA|
07FA| 103C 0080      CONSET MOVE.B   #$80,D0          ;set mid range value default
07FE| 6002          BRA.S   CONSET2
0800|
0800| 70FF          CONOFF MOVEQ   #-1,D0          ;set for contrast off
0802|          CONSET2
0802| 207C 00FC D901      MOVE.L   #VIA2BASE,A0          ;for external entry
0808| 117C 0084 0010      MOVE.B   #$84,DDRB2(A0)          ;GET 6522 BASE ADDR
080E| 10BC 0004          MOVE.B   #4,ORB2(A0)          ;ENSURE NO STRAY DATA TO CONTRAST
0812| 117C 00FF 0018      MOVE.B   #$FF,DDRA2(A0)          ; LATCH BY DISABLING DRIVERS
0818| 1140 0008          MOVE.B   D0,ORA2(A0)          ;NOW SET PORT A AS OUTPUTS
081C| 08D0 0007          MOVE.B   D0,ORA2(A0)          ;set contrast value
0820|          BSET     #7,ORB2(A0)          ;AND STROBE IT
0820|
0820|          RTS4          ;RETURN TO CALLER
0820| 4ED4          #        JMP     (A4)
0822|
0822|

```

```

0822|          .PAGE
0822|          ;-----
0822|          ; Test memory to be used for screen. The screen can then be
0822|          ; used for test icon display. Default choice is last 32K of memory.
0822|          ; If this is invalid, do backwards scan through memory until a valid
0822|          ; area is found.
0822|          ; Assumes: location SCRNBASE = base address of default screen (set by
0822|          ; sizing routine)
0822|          ;-----
0822|
0822|          SCRNTST
0822|          .IF  DIAGS = 1
0822|
0822|          2078 0110          MOVE.L  SCRNBASE,A0          ;get base address of screen
0826|  2278 02A8          MOVE.L  TOTLMEM,A1          ;set end address
082A|          BSR4    RAMTEST          ;and go do test
082A|  49FA 0006          #          LEA      @1,A4
082E|  6000 0680          #          BRA      RAMTEST
0832|          #@1
0832|  676E          BEQ.S  INVTST          ;continue if no error
0834|  08C7 0015          BSET    #MEM,D7          ;else set memory read/write error
0838|
0838|          ; save error results and then start search for good video page
0838|
0838|  6100 064A          BSR      TSTINIT          ;initialize for further testing
083C|  6136          BSR.S  SCRNSAV          ;save results
083E|
083E|  2278 0110          MOVE.L  SCRNBASE,A1          ;set new search end address
0842|  2049          MOVE.L  A1,A0
0844|  247C 0000 8000          MOVE.L  #HEX32K,A2          ;screen size is 32K
084A|  91CA          SUBA.L  A2,A0          ;set new start addr (end-32K)
084C|
084C|  48E7 00E0          @1    MOVEM.L  A0-A2,-(SP)          ;save search addresses
0850|          BSR4    RAMTEST          ;go test
0850|  49FA 0006          #          LEA      @1,A4
0854|  6000 065A          #          BRA      RAMTEST
0858|          #@1
0858|  4CDF 0700          MOVEM.L  (SP)+,A0-A2          ;restore addresses (no effect on CCR)
085C|  670E          BEQ.S  SCRNOK          ;skip if OK (non-zero CCR if error)
085E|  2808          MOVE.L  A0,D4          ;else go save results
0860|  6112          BSR.S  SCRNSAV
0862|  2248          MOVE.L  A0,A1          ;set next end
0864|  91CA          SUBA.L  A2,A0          ; and start addresses
0866|  2008          MOVE.L  A0,D0          ;continue thru all of memory if necessary
0868|  6EE2          BGT.S  @1
086A|
086A|  6036          SCRNNERR BRA.S  INVTST          ;continue testing, leave screen at default

```



```

086C|
086C| 21C8 0110      SCRNOK  MOVE.L  A0,SCRNBASE      ;save new screen base
0870| 6114          BSR.S   SETVLTCH      ;and go set video latch
0872| 602E          BRA.S   INVTST       ;and exit to next test
0874|
0874| ;-----
0874| ; Subroutine to save error results from screen test routine
0874| ; Inputs:
0874| ;     A3 = ptr to base of save result area
0874| ;     D4 = base address of test area
0874| ; Outputs:
0874| ;     None
0874| ; Side Effects:
0874| ;     D1/D3-D4 trashed
0874| ;-----
0874|
0874| 7211      SCRNSAV  MOVEQ   #17,D1          ;divide base address by 128K
0876| E2AC          LSR.L   D1,D4
0878| D844          ADD     D4,D4          ;double for word index to save area
087A| 2203          MOVE.L  D3,D1          ;combine error results
087C| 4843          SWAP    D3
087E| 8641          OR      D1,D3
0880| 8773 4000      OR      D3,0(A3,D4)      ;save and exit
0884| 4E75          RTS
0886|
0886| ;-----
0886| ; Subroutine to set the video latch
0886| ; Inputs:
0886| ;     Location SCRNBASE = logical base address for screen
0886| ; Outputs:
0886| ;     None
0886| ; Side Effects:
0886| ;     D0-D1 trashed
0886| ;-----
0886|
0886| SETVLTCH
0886| 2438 0110      MOVE.L  SCRNBASE,D2      ;get logical screen base address
088A| 2238 02A8      MOVE.L  TOTLMEM,D1      ;get physical amount of memory
088E| 9282          SUB.L   D2,D1          ;compute screen base offset
0890| 2038 0294      MOVE.L  MAXMEM,D0       ;get max physical address
0894| 9081          SUB.L   D1,D0          ;compute physical screen base address
0896| E088          LSR.L   #8,D0          ;convert to page value
0898| EE88          LSR.L   #7,D0
089A| 13C0 00FC E800 MOVE.B  D0,VIDLTCH      ;set latch
08A0| 4E75          RTS                   ;and exit
08A2|
08A2| .PAGE

```

```

08A2| ;-----
08A2| ; Now check state of INVID bit to see if inverse video is installed.
08A2| ; If yes, rewrite last 4 words of screen page to avoid retrace line.
08A2| ;-----
08A2|
08A2| INVTST
08A2| .IF INVERTCK = 1 ; CHG013
08A2| .ENDC ;{INVERTCK} CHG013
08A2| .ENDC ;{DIAGS}
08A2|
08A2| ;-----
08A2| ; Continue testing by now doing COPS VIA test
08A2| ;-----
08A2|
08A2| VIA1TST
08A2| .IF USERINT = 1
08A2|
08A2| ; Draw desktop on screen for test icon display
08A2|
08A2| 6100 2832 BSR DRAWDESK ;draw the desk
08A6| 6134 BSR.S DSPCPURM ;and CPU ROM id CHG001
08A8|
08A8| .ENDC
08A8|
08A8| BSR4 CONSET ;set default contrast
08A8| 49FA 0006 # LEA @1,A4
08AC| 6000 FF4C # BRA CONSET
08B0| #@1
08B0|
08B0| .IF ROM16K = 1
08B0| VIA1CHK LEA VIA1VCT,A3 ;first set up bus error vector
08B4| 21CB 0008 MOVE.L A3,BUSVCTR
08B8| 207C 00FC DD8D MOVE.L #<VIA1BASE+T1LL1>,A0 ;set base address of timer low latch
08BE| 7002 MOVEQ #2,D0 ;set offset to high latch
08C0| BSR6 VIATST ;go test
08C0| 4DFA 0006 # LEA @1,A6
08C4| 6000 FEF2 # BRA VIATST
08C8| #@1
08C8| 670C BEQ.S @2 ;skip if OK
08CA| 08C7 000A BSET #VIA1,D7 ;else set error bit
08CE| 4A87 TST.L D7 ;loop?
08D0| 6BDE BMI.S VIA1CHK ;yes - test again
08D2| 6000 0AC6 BRA TSTCHK ;else abort further testing
08D6|
08D6| 4A87 @2 TST.L D7 ;check for loop mode
08D8| 6BD6 BMI.S VIA1CHK
08DA| 600E BRA.S COPSENBL ;else go test COPS

```

```

08DC|
08DC| ;-----
08DC| ; Subroutine to display CPU ROM id
08DC| ;-----
08DC|
08DC| DSPCPURM
08DC| 103A 371F      MOVE.B  REV,D0          ;read ROM rev          CHG001
08E0| 7A03         MOVEQ   #ROMIDROW,D5      ;setup cursor ptrs      CHG001
08E2| 7C50         MOVEQ   #ROMIDCOL,D6      ;                      CHG001
08E4| 6100 2E54    BSR     DSPVAL          ;do display          CHG001
08E8| 4E75         RTS                      ;                      CHG001
08EA|
08EA| .ENDC
08EA| .PAGE
08EA| ;-----
08EA| ; Try turning COPS on so that keyboard commands can be received
08EA| ;-----
08EA|
08EA| COPSENBL
08EA| 47FA 002A      LEA      COPSVCT,A3        ;set up bus error vector first
08EE| 21CB 0008      MOVE.L  A3,BUSVCTR
08F2| 612C         BSR.S   CPSINIT          ;enable COPS
08F4| 6514         BCS.S   COPSBAD          ;skip if error
08F6| 4A87         TST.L   D7              ;looping desired?
08F8| 6BF0         BMI.S   COPSENBL         ;go repeat test
08FA|
08FA| 2007         MOVE.L  D7,D0              ;get error indicator
08FC| 0280 001F FFFF ANDI.L  #CPIOMSK,D0      ;mask off don't care bits
0902| 6700 00BE      BEQ     RSTSCAN          ;continue if OK to do reset scan
0906| 6000 0A92      BRA     TSTCHK          ;else go report error
090A|
090A| 08C7 000C      COPSBAD BSET   #IOCOPS,D7      ;else set COPS error
090E| 4A87         TST.L   D7              ;looping desired?
0910| 6BD8         BMI.S   COPSENBL         ;go repeat test
0912| 6000 0A86      BRA     TSTCHK          ;else abort further testing
0916|
0916| ;-----
0916| ; Bus error handler for COPS testing with entry point for other I/O tests
0916| ;-----
0916|
0916| COPSVCT MOVEQ   #EIOCOP,D0          ;SET ERROR CODE
0918| 08C7 0012      IOVCT  BSET   #IOEXCP,D7      ;SET I/O EXCEPTION ERROR
091C| 6000 FE3A      BRA     EXCP0          ;AND GO HANDLE EXCEPTION
0920|
0920| ;-----
0920| ; Subroutine to initialize COPS interface for use
0920| ;-----

```

```

0920| 207C 00FC DD81      CPSINIT MOVEA.L #VIA1BASE,A0      ;GET VIA BASE ADDRESS
0926| 117C 0001 0016      MOVE.B #01,ACR1(A0)      ;SET PORT A LATCH ENABLE
092C| 0028 0009 0018      OR.B #09,PCR1(A0)      ;SET HANDSHAKE ENABLE
0932| 117C 007F 001C      MOVE.B #07F,IER1(A0)      ;CLEAR ALL INTRPT ENABLES
0938| 117C 007F 001A      MOVE.B #07F,IFR1(A0)      ;AND CLEAR FLAGS
093E|
093E|      ; Now turn COPS on, disabling mouse and NMI key
093E|
093E| 4240      TURNON CLR D0      ;SET FOR PORT ON CMD
0940| 6114      BSR.S COPSCMD      ;SEND TO COPS
0942| 6510      BCS.S @1      ;EXIT IF TIMEOUT ERROR
0944|
0944| 7070      MOVEQ #070,D0      ;DISABLE MOUSE
0946| 610E      BSR.S COPSCMD
0948| 650A      BCS.S @1
094A|
094A|      .IF DEBUG = 0
094A| 7050      MOVEQ #050,D0      ;disable NMI key
094C| 6108      BSR.S COPSCMD
094E| 6504      BCS.S @1
0950| 7060      MOVEQ #060,D0
0952| 6102      BSR.S COPSCMD
0954|      .ENDC
0954|
0954| 4E75      @1 RTS      ;AND EXIT
0956|
0956|      .PAGE
0956|
0956|      ;-----
0956|      ; Subroutine to send cmd to COPS
0956|      ; Assumes registers:
0956|      ; D0 = cmd value
0956|      ; If COPS does not respond, timeout error indicated by setting carry bit.
0956|      ;-----
0956|
0956|      COPSCMD
0956| 48E7 F8E0      MOVEM.L D0-D4/A0-A2,-(SP) ;save regs
095A|      DISABLE      ;disable all interrupts
095A| 40E7      # MOVE SR,-(SP)
095C| 007C 0700      # ORI #0700,SR
0960| 207C 00FC DD81      MOVEA.L #VIA1BASE,A0      ;set COPS VIA interface ptr
0966| 2248      MOVEA.L A0,A1      ;save for use as port B output reg address
0968| 2448      MOVEA.L A0,A2
096A| D4FC 0006      ADDA #DDRA1,A2      ;compute address for port A data direction reg
096E| 7440      MOVEQ #040,D2      ;set up constants for later use
0970| 76FF      MOVEQ #-1,D3
0972| 7806      MOVEQ #6,D4
0974|

```

```

0974| 1140 001E          MOVE.B D0,PORTA1(A0)    ;set cmd in data reg (no handshake)
0978|
0978|          ;-----
0978|          ; First find a ready state (CRDY low)
0978|          ; Each of the following loops take about 32 machine cycles = 6.4 us plus
0978|          ; a variable amount of time for sync with 6522 (max = 2us)
0978|          ;-----
0978|
0978| 323C 061A          MOVE    #$061A,D1          ;set timeout for about 10 ms
097C| 5341          @3      SUBQ    #1,D1
097E| 6732          BEQ.S    @1          ;exit if timeout
0980| 0911          BTST     D4,(A1)          ;else wait for "ready" (bit 6 = CRDY)
0982| 66F8          BNE.S    @3
0984|
0984|          ; Now find the next ready state to insure enough time available for data
0984|
0984| C0FC 0001          MULU     #1,D0          ;kill some time (about 15.2 us) to get
0988|          ; out of previous CRDY
0988| 323C 061A          MOVE    #$061A,D1          ;reinit timeout count
098C| 5341          @4      SUBQ    #1,D1
098E| 6722          BEQ.S    @1          ;exit if timeout
0990| 0911          BTST     D4,(A1)          ;wait for another "ready"
0992| 66F8          BNE.S    @4
0994|
0994| 1483          MOVE.B    D3,(A2)          ; ok, jam out the data
0996|
0996|          ; Now wait for CRDY high and then hold data for COPS to read
0996|
0996| 323C 061A          MOVE    #$061A,D1          ;set timeout for about 10 ms
099A| 5341          @5      SUBQ    #1,D1
099C| 6714          BEQ.S    @1          ;exit if timeout
099E| 0911          BTST     D4,(A1)          ;wait for "not-ready"
09A0| 67F8          BEQ.S    @5
09A2|
09A2| 700A          @6      MOVEQ    #$A,D0          ; force about a 40 ms
09A4| 5340          SUBQ    #1,D0          ; delay for COPS hold time
09A6| 6EFC          BGT.S    @6
09A8|
09A8| 4212          CLR.B     (A2)          ; reset direction reg now
09AA| 117C 0082 001C  MOVE.B    #$82,IER1(A0)    ; and, enable CA1
09B0| 6008          BRA.S    @2          ; go to normal exit
09B2|
09B2|          ; Timeout occurred - set error indicator
09B2| @1      ENABLE          ;reenable
09B2| 46DF          #      MOVE     (SP)+,SR
09B4| 003C 0001          ORI.B    #$01,CCR          ;set carry bit
09B8| 6002          BRA.S    @9          ;skip to exit

```

```

09BA|
09BA|      @2      ENABLE                      ;restore interrupt levels
09BA| 46DF      #      MOVE      (SP)+,SR
09BC| 4CDF 071F      @9      MOVEM.L (SP)+,D0-D4/A0-A2 ;restore regs
09C0| 4E75      RTS                      ;and return to caller
09C2|
09C2|      .PAGE
09C2|      ;-----
09C2|      ; Scan COPS for proper reset codes. Delay added for normal COPS power-up
09C2|      ; time of about 1.7 seconds.
09C2|      ;
09C2|      ; Send reset signal and then scan keyboard/mouse interface. First "clears"
09C2|      ; COPS of any pending codes, and then issues reset. Works via
09C2|      ; a "state machine" that checks codes received and sets flags as follows:
09C2|      ;
09C2|      ;      D1 = 0 - reset signal in place
09C2|      ;      = 1 - reset signal removed
09C2|      ;      D3 = 0 - no keyboard codes received => keyboard disconnected
09C2|      ;      = 1 - keyboard disconnect code ($80/$FD) received
09C2|      ;      => ignore, may be old keyboard
09C2|      ;      = 2 - keyboard disconnect/connect codes ($80/$FD/$80/id) received
09C2|      ;      => keyboard connected
09C2|      ;
09C2|      ;      D4 = 0 - no mouse codes received => mouse connected
09C2|      ;      = 1 - only mouse connect code ($87) received => ignore, may be old sys
09C2|      ;      = 2 - mouse connect/disconnect ($87/$07) codes received
09C2|      ;      => mouse disconnected
09C2|      ;-----
09C2|
09C2|      RSTSCAN
09C2| 2278 0260      MOVE.L KBDQPTR,A1      ;setup buffer ptrs
09C6| 347C 02C0      MOVEA #QEND,A2
09CA| 616C      @1      BSR.S GETJMP      ;clear COPS queue, saving data
09CC| 64FC      BCC.S @1
09CE|
09CE| 6100 00DA      BSR      RSTKBD      ;do reset of keyboard/mouse interfaces
09D2|
09D2|      .IF ROM4K = 0
09D2| 4281      CLR.L D1      ;init some flags
09D4| 4283      CLR.L D3
09D6| 4284      CLR.L D4
09D8| 615E      BSR.S GETJMP      ;check for data
09DA| 6560      BCS.S RSTXIT      ;exit if none (may be old keyboard)
09DC|
09DC|      ; State 0 - waiting for reset flag or mouse connect code
09DC| 0C00 0080      RST0      CMPI.B #RSTCODE,D0      ;reset flag?
09E0| 6724      BEQ.S RST1      ;skip if yes to state 1

```

```

09E2| 0C00 0087          CMPI.B  #MSPLG,D0          ;mouse connect code?
09E6| 670E              BEQ.S    RST2              ;skip if yes to state 2
09E8| 0C00 0007          CMPI.B  #MSUNPLG,D0        ;mouse disconnect code only?
09EC| 6602              BNE.S    GET0
09EE| 7802              MOVEQ    #2,D4              ;set flag for disconnect state
09F0| 6146              GET0      BSR.S    GETJMP        ;go get next code
09F2| 6548              BCS.S    RSTXIT        ;exit if no more codes
09F4| 60E6              BRA.S    RST0          ;else continue scan loop
09F6|
09F6|          ; State 2 - waiting for mouse unplugged code
09F6|
09F6| 7801              RST2      MOVEQ    #1,D4              ;set flag for mouse connect received
09F8| 613E              BSR.S    GETJMP        ;go get next byte
09FA| 6540              BCS.S    RSTXIT        ;exit if none or queue full
09FC| 0C00 0007          CMPI.B  #MSUNPLG,D0        ;mouse disconnect code?
0A00| 66DA              BNE.S    RST0          ;no - return to state 0
0A02| 7802              MOVEQ    #2,D4              ;yes - set flag
0A04| 60EA              BRA.S    GET0          ;and return to state 0
0A06|
0A06|          ; State 1 - waiting for reset code
0A06|
0A06| 6130              RST1      BSR.S    GETJMP        ;go get next byte
0A08| 6532              BCS.S    RSTXIT        ;exit if no more
0A0A| 0C00 00FD          CMPI.B  #KUNPLG,D0        ;keyboard unplugged code?
0A0E| 6604              BNE.S    @1            ;skip if not
0A10| 7601              MOVEQ    #1,D3              ;else set flag
0A12| 60DC              BRA.S    GET0          ;and return to state 0
0A14|
0A14| 0C00 00DF          @1      CMPI.B  #$DF,D0          ;id code?
0A18| 6208              BHI.S    @2            ;skip if not
0A1A| 11C0 01B2          MOVE.B  D0,KEYID          ;else save for later use
0A1E| 7602              MOVEQ    #2,D3              ;update flag
0A20| 60CE              BRA.S    GET0          ;and return to state 0
0A22| 0C00 00FF          @2      CMPI.B  #KCERR,D0        ;Keyboard COPS error?
0A26| 6604              BNE.S    @3            ;skip if not
0A28| 08C7 000D          BSET     #KBD COPS,D7      ;else set error indicator
0A2C|
0A2C| 0C00 00FE          @3      CMPI.B  #ICERR,D0        ;I/O COPS error code?
0A30| 6604              BNE.S    @4            ;skip if not
0A32| 08C7 000C          BSET     #IOCOPS,D7      ;else set error indicator
0A36|
0A36| 60B8              @4      BRA.S    GET0          ;continue scan from state 0
0A38|
0A38|          ; Insert to save code space
0A38|
0A38| 6144              GETJMP   BSR.S    GETDATA        ;go get COPS data
0A3A| 4E75              RTS              ;and return to caller

```

```

0A3C|           ; Reset exit - analyze results
0A3C|
0A3C| 4A01      RSTXIT  TST.B   D1           ;reset signal lifted?
0A3E| 660A      BNE.S   @1           ;skip if yes
0A40| 6100 0082  BSR     CLRRST        ;else remove reset signal
0A44| 7201      MOVEQ   #1,D1        ;set "removed flag"
0A46| 6136      BSR.S   GETDATA       ;any data?
0A48| 6492      BCC.S   RST0         ;go decode if yes
0A4A|
0A4A|
0A4A| @1
0A4A|           .IF  FINKBD = 1
0A4A| 4A03      TST.B   D3           ;any keyboard data detected?
0A4C| 6604      BNE.S   MSCHK        ;skip if yes - assume keybd connected
0A4E| 08C7 0017  BSET    #KBDOUT,D7    ;if none, keyboard is disconnected
0A52|
0A52| MSCHK
0A52| 4A04      TST.B   D4           ;any mouse data?
0A54| 6714      BEQ.S   SCANXIT       ;skip if none - mouse connected
0A56|
0A56| 5344      @1     SUBQ   #1,D4        ;flag = 1?
0A58| 6710      BEQ.S   SCANXIT       ;ignore if yes
0A5A|
0A5A| 08C7 0018  BSET    #MOUSOUT,D7    ;else mouse disconnected
0A5E| 600A      BRA.S   SCANXIT       ;and go to exit
0A60|
0A60|           .ELSE
0A60|           .ENDC                ;{FINKBD}
0A60|
0A60|           ; Error exits - set appropriate indicator
0A60|
0A60| 08C7 0014  SCANERR BSET    #IOKBD,D7    ;I/O or keyboard failure
0A64| 6004      BRA.S   SCANXIT
0A66|
0A66| 08C7 000C  IOCERR  BSET    #IOCOPS,D7    ;I/O COPS error
0A6A|
0A6A|           .ELSE
0A6A|           .ENDC                ;{ROM4K}
0A6A|
0A6A| 21C9 0260  SCANXIT MOVE.L  A1,KBDQPTR    ;save queue ptr for later use
0A6E|
0A6E|           .IF  ROM4K = 0
0A6E| 2007      MOVE.L  D7,D0           ;check error codes
0A70| 0280 0018 3000  ANDI.L  #SCANMSK,D0    ; for scan errors
0A76| 4A80      TST.L   D0           ;any found?
0A78| 6600 0920  BNE     TSTCHK        ;skip if yes
0A7C|           .ENDC
0A7C|

```



```

0A7C| 606A          BRA.S   BEEP          ;else continue testing
0A7E|
0A7E|              ;-----
0A7E|              ; Subroutine to get COPS data
0A7E|              ; Assumes registers:
0A7E|              ;     D0 - scratch use          A0 - VIA address
0A7E|              ;     D1 - unused              A1 - Ptr to data save area
0A7E|              ;     D2 - scratch use          A2 - Ptr to end of data area
0A7E|              ; Puts data in save area and also leaves in register D0.
0A7E|              ; Carry bit set if timeout error or keyboard queue full.
0A7E|              ;-----
0A7E|
0A7E| B5C9          GETDATA CMPA.L  A1,A2          ;check if at end of queue
0A80| 671A          BEQ.S   @2          ;exit if yes
0A82| 243C 0000 01FF      MOVE.L  #$1FF,D2      ;else set timeout for about 5 ms
0A88| 207C 00FC DD81      MOVEA.L #VIA1BASE,A0    ;set COPS VIA interface ptr
0A8E| 1028 001A          @1  MOVE.B  IFR1(A0),D0    ;check if data avail
0A92| 0800 0001          BTST   #1,D0
0A96| 660A          BNE.S   GETIT          ;skip if yes
0A98| 5342          SUBQ   #1,D2
0A9A| 66F2          BNE.S   @1          ;else continue
0A9C| 003C 0001          @2  ORI.B   #$01,CCR      ;set timeout error
0AA0| 4E75          RTS
0AA2|
0AA2| 1028 0002          GETIT  MOVE.B  ORA1(A0),D0    ;read data
0AA6| 12C0          MOVE.B  D0,(A1)+          ;save it
0AA8| 4E75          RTS          ;and exit with results
0AAA|
0AAA|              .PAGE
0AAA|
0AAA|              ;-----
0AAA|              ; Subroutine to do reset of keyboard and mouse interfaces
0AAA|              ; Inputs:
0AAA|              ;     None
0AAA|              ; Outputs:
0AAA|              ;     None
0AAA|              ; Side Effects:
0AAA|              ;     D0/A0 trashed
0AAA|              ;-----
0AAA|
0AAA| 207C 00FC DD81      RSTKBD MOVEA.L #VIA1BASE,A0    ;set VIA ptr
0AB0| 0890 0000          BCLR   #0,ORB1(A0)      ;set reset signal
0AB4| 0028 0001 0004      ORI.B   #$01,DDRB1(A0)    ;send it
0ABA| 203C 0000 0BB8      MOVE.L  #3000,D0        ;do delay for 12 ms
0AC0| 6120          BSR.S   DELAY
0AC2| 4E75          RTS
0AC4|
0AC4|              ;-----

```

```

0AC4|          ; Subroutine to remove reset signal for keyboard and mouse interfaces
0AC4|          ; Inputs:
0AC4|          ;   A0 = ptr to parallel port VIA (set in RSTKBD routine)
0AC4|          ; Outputs:
0AC4|          ;   None
0AC4|          ; Side Effects:
0AC4|          ;   D0 trashed
0AC4|          ;-----
0AC4|
0AC4| 08D0 0000      CLRRST  BSET    #0,ORB1(A0)      ;remove reset signal
0AC8| 6112          BSR.S   KBDDELAY      ;delay for keyboard reset time
0ACA| 4E75          RTS
0ACC|
0ACC|          ;-----
0ACC|          ; Subroutine to delay for count in D0 (each count = 4 us).  Additional
0ACC|          ; entry points set up for fixed delays.
0ACC|          ;-----
0ACC|
0ACC|          .IF ROM4K = 0
0ACC| 203C 0000 61A8  DELAY_1 MOVE.L  #TNTHSEC,D0      ;.1 second delay
0AD2| 600E          BRA.S   DELAY
0AD4|
0AD4| 203C 0013 12D0  DELAY5  MOVE.L  #FIVESEC,D0      ;5 second delay
0ADA| 6006          BRA.S   DELAY
0ADC|
0ADC|          KBDDELAY
0ADC| 203C 0006 7C28      MOVE.L  #KBDDLY,D0      ;delay for COPS debounce loop
0AE2|          .ENDC
0AE2|
0AE2| 5380          DELAY  SUBQ.L  #1,D0      ;loop until count = 0
0AE4| 66FC          BNE.S   DELAY
0AE6| 4E75          RTS
0AE8|
0AE8|          .PAGE
0AE8|          ;-----
0AE8|          ; Sound starting "tone"
0AE8|          ;-----
0AE8|
0AE8|          BEEP
0AE8| 6104          BSR.S   CLICK      ;go click speaker
0AEA| 6000 00AA      BRA     VIDTST      ;then go do video test
0AEE|
0AEE|          ;-----
0AEE|          ; Subroutine to set parms for speaker "click"
0AEE|          ;-----
0AEE|
0AEE| 103C 00A0      CLICK  MOVE.B  #$A0,D0      ;set frequency

```

```

0AF2| 7200          MOVEQ    #0,D1          ;and duration
0AF4| 7408          MOVEQ    #8,D2          ;and volume (medium)          RM000
0AF6|                                     ;then fall thru to tone routine          RM000
0AF6|
0AF6|          ;-----
0AF6|          ; Routine to beep the speaker
0AF6|          ; Assumes regs set up as
0AF6|          ;   D0 = desired frequency ($00 - $AA)
0AF6|          ;   D1 = duration (0 = .5 msec)
0AF6|          ;   D2 = volume (0,2,4,...,E)
0AF6|          ;-----
0AF6|
0AF6| 48E7 1088      TONE      MOVEM.L A0/A4/D3,-(SP) ;save regs
0AFA|          BSR4      TONE2          ;go do tone
0AFA| 49FA 0004      #          LEA      @1,A4
0AFE| 6006          #          BRA.S    TONE2
0B00|          #@1
0B00| 4CDF 1108      MOVEM.L (SP)+,A0/A4/D3 ;restore and exit
0B04| 4E75          RTS
0B06|
0B06|          ; separate entry point for call without memory usage
0B06|
0B06| 207C 00FC DD81  TONE2   MOVEA.L #VIA1BASE,A0 ;set VIA ptr
0B0C| 0028 000E 0004  ORI.B   #$0E,DDRB1(A0) ;set volume bits for output
0B12| 0210 00F1      ANDI.B   #$F1,ORB1(A0) ;clear and then
0B16| 8510          OR.B     D2,ORB1(A0) ; set volume bits
0B18| 0228 00E3 0016  ANDI.B   #$E3,ACR1(A0) ;clear shift mode bits
0B1E| 0028 0010 0016  ORI.B   #$10,ACR1(A0) ;set shift reg for continuous rotate
0B24|
0B24|          ; check system type
0B24|
0B24| 4A39 00FC C031      TST.B   DISKROM          ;test for Lisa 1 board          CHG014
0B2A| 6A10          BPL.S    @3          ;no changes if yes          CHG014
0B2C| 0839 0005 00FC C031  BTST   #SLOTMR,DISKROM ;else check if slow timers          CHG029
0B34| 6606          BNE.S    @3          ;skip if yes          CHG029
0B36| 1600          MOVE.B   D0,D3          ;else adjust input parm          CHG014
0B38| E40B          LSR.B    #2,D3          ; by factor of .25          CHG014
0B3A| D003          ADD.B    D3,D0          ;          CHG014
0B3C|
0B3C| 1140 0010      @3      MOVE.B   D0,T2CL1(A0) ;set frequency
0B40| 117C 000F 0014      MOVE.B   #$0F,SHR1(A0) ;set for square wave and trigger
0B46|
0B46|          ; Do time delay - enter with count in D1 (about .5 msec per count)
0B46|
0B46| 363C 00D0      @1      MOVE.W   #$00D0,D3          ;set delay constant
0B4A| 51CB FFFE      @2      DBF      D3,@2
0B4E| 51C9 FFF6      DBF      D1,@1

```

```

0B52|
0B52| 0228 00E3 0016      SILENCE ANDI.B  #$E3,ACR1(A0)  ;disable tone
0B58|                      RTS4                      ;and return
0B58| 4ED4                #      JMP      (A4)
0B5A|
0B5A|
0B5A|                      .IF  DIAGS = 1
0B5A|                      ;-----
0B5A|                      ;  Routine to handle I/O board selection errors.  Does check for access
0B5A|                      ;  of other I/O board devices to try to pinpoint error.
0B5A|                      ;-----
0B5A|
0B5A| 08C7 0010      NOIO    BSET    #RS232B,D7      ;set SCC port B access error      CHG027
0B5E| 3E7C 0480      MOVE    #STKBASE,SP      ;restore stack pointer
0B62|                      ;  try access of other I/O board devices
0B62|
0B62| 47FA 0012      LEA      NOIO2,A3      ;set up new bus error vector
0B66| 21CB 0008      MOVE.L  A3,BUSVCTR
0B6A| 207C 00FC D901    MOVE.L  #VIA2BASE,A0      ;set base address      CHG027
0B70| 4A10      TST.B  (A0)      ;try access
0B72| 6000 FC0C      BRA      VIA2TST      ;return to testing if OK      CHG027
0B76|
0B76| 08C7 000B      NOIO2    BSET    #VIA2,D7      ;set VIA #2 error also      CHG027
0B7A| 47FA 0012      LEA      NOIO3,A3      ;try final access to VIA #1      CHG027
0B7E| 21CB 0008      MOVE.L  A3,BUSVCTR
0B82| 207C 00FC DD81    MOVE.L  #VIA1BASE,A0      ;set base address      CHG027
0B88| 4A10      TST.B  (A0)
0B8A| 6000 FC96      BRA.S   SCRNTST      ;exit if OK      CHG027
0B8E|
0B8E| 08C7 0001      NOIO3    BSET    #CPUSEL,D7      ;most likely CPU board error
0B92|
0B92| 6000 0806      BRA      TSTCHK      ;go report errors      CHG027
0B96|
0B96|                      .ENDC
0B96|
0B96|                      .INCLUDE RM248.S.TEXT
0B96|
0B96|                      .PAGE
0B96|                      ;-----
0B96|                      ;  VIDEO CIRCUITRY TEST
0B96|                      ;  The following test checks the vertical retrace signal of the
0B96|                      ;  video circuitry to verify it is toggling.
0B96|                      ;  Register usage:
0B96|                      ;      D0 = timeout count      A0 = unused
0B96|                      ;      D1 = unused      A1 = unused
0B96|                      ;      D2 = bit pointer      A2 = unused
0B96|                      ;      D3 = unused      A3 = address to disable VTIR

```

```

0B96|      ;      D4 = unused      A4 = address to enable VTIR
0B96|      ;      D5 = unused      A5 = address of bus status register
0B96|      ;-----
0B96|
0B96|      VIDTST
0B96|          .IF      ROM4K = 0
0B96|
0B96|          .IF      USERINT = 1
0B96| 6100 25E8      BSR      MAKETEST      ;display test icons
0B9A| 327C 1DF6      MOVEA  #CPUSTRT,A1    ;hilite CPU board icon
0B9E| 6100 29D4      BSR      INVICON
0BA2|          .ENDC
0BA2|      VIDCHK
0BA2| 267C 00FC E018  MOVEA.L #VTIRDIS,A3    ;ADDRESS FOR DISABLING VTIR
0BA8| 287C 00FC E01A  MOVEA.L #VTIRENB,A4    ;ADDRESS FOR VTIR ENABLE
0BAE| 2A7C 00FC F801  MOVEA.L #STATREG,A5    ;STATUS REGISTER LOCATION FOR BYTE OPS
0BB4|
0BB4| 303C 0DF4      MOVE     #$0DF4,D0      ;SET TIMEOUT COUNT FOR ABOUT 20 MS
0BB8| 7402          MOVEQ    #VRBIT,D2      ;VR BIT LOCATION
0BBA|
0BBA| 4A53          TST      (A3)      ;RESET THEN
0BBC| 4A54          TST      (A4)      ; ENABLE VTIR
0BBE| 0515      @1    BTST     D2,(A5)      ;WAIT FOR LOW
0BC0| 6706          BEQ.S    @2          ;EXIT IF YES
0BC2| 51C8 FFFA      DBF     D0,@1        ;ELSE LOOP (ABOUT 5.6 MS PER LOOP)
0BC6| 600C          BRA.S    VIDERR      ;AND SET ERROR IF TIMEOUT
0BC8|
0BC8| 4A53      @2    TST      (A3)      ;RESET VTIR
0BCA| 4A54          TST      (A4)      ;THEN RENABLE
0BCC| 0515          BTST     D2,(A5)      ;SHOULD BE HIGH BY NOW
0BCE| 6704          BEQ.S    VIDERR      ;GO TO ERROR EXIT IF NOT
0BD0| 4A53          TST      (A3)      ;DISABLE VTIR
0BD2| 600C          BRA.S    VIDXIT      ;and go to exit
0BD4|
0BD4|      ; Error exit
0BD4|
0BD4| 08C7 0002      VIDERR BSET     #VID,D7    ;SET ERROR INDICATOR
0BD8| 4A87          TST.L     D7          ;in loop mode?
0BDA| 6BC6          BMI.S    VIDCHK      ;restart test if yes
0BDC| 6000 07BC      BRA     TSTCHK      ;else go to error msg routine
0BE0|
0BE0|      ; Normal exit
0BE0|
0BE0|      VIDXIT
0BE0|
0BE0|      ;-----
0BE0|      ; Now, try reading of system serial number

```

```

OBE0|                                     ;-----
OBE0|
OBE0| 307C 0240          MOVEA    #SERNUM,A0      ;ptr for save of serial #
OBE4| 6110              BSR.S     RDSEEN         ;go do read
OBE6| 64EC              BCC.S     VIDERR         ;exit if error
OBE8| 4A79 00FC E018    TST       VTIRDIS        ;else disable vertical retrace
OBE6| 4A87              TST.L     D7             ;check for loop mode
OBF0| 6BB0              BMI.S     VIDCHK         ;if not, fall thru to next test
OBF2| 6000 0168        BRA        PARTST         ;and go on to next test
OBF6|
OBF6|          .PAGE
OBF6|                                     ;-----
OBF6|                                     ;
OBF6|                                     ; Routine to read system serial # from video prom.
OBF6|                                     ; Written by Ken Schmal and Ron Hochsprung.
OBF6|                                     ;
OBF6|                                     ; Register Usage:
OBF6|                                     ;
OBF6|                                     ; temporary and iterative          D0
OBF6|                                     ; temporary and iterative          D1
OBF6|                                     ; temporary and iterative          D2
OBF6|                                     ; temporary and iterative          D3
OBF6|                                     ; boolean FOUND to be returned      D4
OBF6|                                     ; pointer to save area for serial #    A0
OBF6|                                     ; SN1 & SN2 pointer                A1
OBF6|                                     ; STATUS REGISTER pointer          A2
OBF6|                                     ; SCRACH array pointer              A3
OBF6|                                     ; SCRACH END pointer                A4
OBF6|                                     ; Tag const                      A5
OBF6|                                     ; static link and stack frame
OBF6|                                     ; base pointer register            A6
OBF6|                                     ;
OBF6|                                     ; Returns with carry bit set if all OK.
OBF6|                                     ; All registers except D7 and A0 trashed.
OBF6|                                     ;
OBF6|                                     ;-----
OBF6|
OBF6| RDSEEN
OBF6|
OBF6| 48E7 0180          MOVEM.L     D7/A0,-(SP)      ;save regs
OBF6|
OBF6|                                     ; turn off all interrupts
OBF6| 40E7              move        SR, -(sp)         ;save the present status register
OBF6| 007C 0700          ori.w       #$0700, SR       ;set interrupt to level 7
OC00|
OC00|                                     ;-----
OC00|                                     ; now set up registers for the algorithm

```

```

0C00|                                     ;-----
0C00|
0C00| 227C 00FE 8000          move.l      #Snum, a1          ;location in MMU of SN1 & SN2
0C06| 247C 00FC F801          move.l      #Statreg,a2        ;Status Register pointer
0C0C| 4E56 FF18              link         a6, #dStack        ;make room for SCRACH
0C10| 47EE FF18              lea          dScrach(a6), a3      ;get pointer for SCRACH
0C14| 49FA 0142              lea          Tag,a4
0C18| 2D48 FFF8              move.l      a0,dSavArray(a6)
0C1C|
0C1C|                                     ;-----
0C1C|                                     ;      first we get the block out of the vertical half
0C1C|                                     ;-----
0C1C|                                     ;
0C1C|                                     ;      sync up to the vertical retrace bit
0C1C|                                     ;
0C1C|
0C1C| GetBits1:
0C1C|
0C1C| 7202                    moveq        #2, d1              ;vertical retrace is bit #2
0C1E| 2D7C 0000 0007 FFFC    move.l      #BytesPerRead,dLcnt(a6) ;read this many bytes
0C26| 4279 00FC E018          clr         VTIRDIS          ;clear vertical retrace bit
0C2C| 4279 00FC E01A          clr         VTIRENB          ;set vertical retrace interrupt
0C32| 0312                    @1:    btst         d1, (a2)        ;wait until it's true
0C34| 66FC                    bne.s       @1
0C36|
0C36|                                     ;
0C36|                                     ;----- read the first block -----
0C36|                                     ;
0C36|
0C36| @3:    movem              (a1), d0-d7
0C3A| 4893 00FF              movem              d0-d7, (a3)
0C3E| 508B                    addq.l      #8, a3
0C40| 508B                    addq.l      #8, a3
0C42| 4E71                    nop
0C44| 7008                    moveq        #dlycnst-1, d0
0C46| 53AE FFFC              subq.l      #1, dLcnt(a6)
0C4A| 5FC8 FFFE              @4:    dble         d0, @4
0C4E| 6EE6                    bgt.s       @3
0C50|
0C50|                                     ;-----
0C50|                                     ;      then we get the block out of the horizontal half
0C50|                                     ;-----
0C50|                                     ;
0C50|                                     ;      kill time until we're near the last vertical retrace line
0C50|                                     ;
0C50|
0C50| GetBits2:

```

```

0C50| 2D7C 0000 0007 FFFC      move.l      #BytesPerRead, dLcnt(a6);get the last few bytes
0C58| 303C 00AB                move.w      #TKiller-1, d0      ;time killer constant
0C5C| 51C8 FFFE                @1:   dbra         d0, @1                ;loop
0C60|
0C60|                          ;
0C60|                          ;----- read the second or last block -----
0C60|                          ;
0C60|
0C60| 4C91 00FF                @2:   movem     (a1), d0-d7
0C64| 4893 00FF                movem     d0-d7, (a3)
0C68| 508B                    addq.l     #8, a3
0C6A| 508B                    addq.l     #8, a3
0C6C| 4E71                    nop
0C6E| 7008                    moveq     #dlycnst-1, d0
0C70| 53AE FFFC                subq.l     #1, dLcnt(a6)
0C74| 5FC8 FFFE                @3:   dble      d0, @3
0C78| 6EE6                    bgt.s     @2
0C7A|
0C7A|                          ;-----
0C7A|                          ;       now we have to find sync bytes and extract the bit stream
0C7A|                          ;-----
0C7A|
0C7A| 4279 00FC E018          clr         VTIRDIS          ;turn off vertical retrace
0C80| 7801                    moveq     #1, d4              ;initialize FOUND to true
0C82|
0C82|      GetBytes:
0C82| 47EE FF18              lea         dScrach(a6), a3      ;pointer to 1/2 Scrach Array pointer
0C86| 284B                    move.l     a3, a4
0C88| D8FC 0070              adda       #HalfSize, a4        ;pointer to end of 1/2 Scrach Array      RM000
0C8C|                          ;
0C8C|                          ;       find the first sync byte
0C8C|                          ;
0C8C| 6100 007C              bsr         FindSync
0C90| 4A44                    tst.w     d4
0C92| 6764                    beq.s     Exit                ;exit if no sync byte found
0C94|                          ;
0C94|                          ;       now pull out the first block from the bit stream
0C94|                          ;
0C94| 6100 009E              bsr         GetNibbles
0C98|                          ;       here we look for the second sync byte.
0C98|                          ;
0C98| 47EE FF18              lea         dScrach(a6), a3
0C9C| D6FC 0070              adda       #HalfSize, a3        ;pointer to 2/2 Scrach Array pointer      RM000
0CA0| 284B                    move.l     a3,a4
0CA2| D8FC 0070              adda       #HalfSize,a4        ;pointer to end of 2/2 Scrach Array      RM000
0CA6|                          ;
0CA6| 6100 0062              bsr         FindSync

```



```

0CAA| 4A44          tst.w          d4
0CAE|              ;      beq.s          Exit          ;again, exit if no sync byte found
0CAE|              ;      now pull out second block from the bit stream
0CAE|              ;
0CAE| 6100 0084          bsr          GetNibbles
0CB2|
0CB2|              ;-----
0CB2|              ;      Check the checksum of the read data
0CB2|              ;-----
0CB2|
0CB2|      CheckSum:
0CB2| 206E FFF8          move.l      dSavArray(a6),a0
0CB6| 4240          clr.w          d0
0CB8|
0CB8| 1028 0018          move.b      24(a0),d0
0CBC| 343C 0064          move.w      #100,d2
0CC0| C0C2          mulu          d2,d0
0CC2|
0CC2| 1228 0019          move.b      25(a0),d1
0CC6| 343C 000A          move.w      #10,d2
0CCA| C2C2          mulu          d2,d1
0CCC| D041          add.w          d1,d0
0CCE|
0CCE| 1228 001A          move.b      26(a0),d1
0CD2| D041          add.w          d1,d0
0CD4|
0CD4| 4241          clr.w          d1
0CD6| 4242          clr.w          d2
0CD8| 4243          clr.w          d3
0CDA| 1630 1000          @2:      move.b      0(a0,d1),d3
0CDE| D443          add.w          d3,d2
0CE0| 5241          addq.w          #1,d1
0CE2| 0C41 0018          cmpi.w      #24,d1
0CE6| 66F2          bne.s          @2
0CE8|
0CE8| 1628 001B          move.b      27(a0), d3
0CEC| D443          add.w          d3,d2
0CEE| 0442 003C          subi.w      #4 * $F, d2
0CF2| B440          cmp.w          d0,d2
0CF4| 6702          beq.s          @3
0CF6| 4244          clr.w          d4
0CF8|      @3:
0CF8|              ;-----
0CF8|              ;      job well done, lets go home
0CF8|              ;-----
0CF8|
0CF8|      Exit:

```

```

OCF8| 4E5E                unlk          a6
OCFA| 46DF                move          (sp)+, SR          ;restore status reg
OCFC| 4CDF 0180           MOVEM.L      (SP)+,D7/A0        ;and regs
OD00| 4279 00FC E01A      clr          VTIRENB          ;re-enable interrupts
OD06| E24C                LSR          #1,D4              ;shift to set/reset error indicator
OD08| 4E75                @1      RTS                    ; and exit
OD0A|
OD0A|                .PAGE
OD0A|                ;-----
OD0A|                ;      subroutine to find a sync byte
OD0A|                ;-----
OD0A|
OD0A|      FindSync:
OD0A| 4280                clr.l          d0
OD0C| 7202                moveq         #2, d1              ;two passes to find the sync byte
OD0E| 341B                @1:    move.w      (a3)+, d2        ;
OD10| E34A                lsl.w          #1, d2              ;
OD12| E310                roxl.b         #1, d0              ;get SN1
OD14| B9CB                cmpa.l         a3, a4              ;assure the buffer's circular
OD16| 660A                bne.s          @2                  ;
OD18| D7FC FFFF FF90      adda.l         #-HalfSize, a3      ;if it's at the end then
OD1E| 5341                subq          #1, d1              ; check if it's the second try
OD20| 670E                beq.s          @3                  ; and exit if so
OD22| 0C00 00FF          @2:    cmpi.b         #$0ff, d0      ;test here if it's a sync byte
OD26| 66E6                bne.s          @1                  ;no: loop again
OD28| E948                lsl.w          #4, d0              ;yes: adjust the byte
OD2A| E808                lsr.b          #4, d0              ;
OD2C| 30C0                move.w         d0, (a0)+          ;save it
OD2E| 4E75                rts                    ;and return
OD30|
OD30| 4244                @3:    clr.w          d4              ;uh, oh. No sync byte.
OD32| 4E75                rts                    ;clear FOUND and return
OD34|                ;-----
OD34|                ;      subroutine to pull out a 14 nibble block from the bit stream
OD34|                ;-----
OD34|
OD34|      GetNibbles:
OD34| 7406                moveq         #BytesPerRead-1, d2    ;
OD36| 7208                @1:    moveq         #8, d1          ;8 bits/byte
OD38| 4280                clr.l          d0                  ;
OD3A| E3DB                @2:    lsl          (a3)+          ;get SN1 in the next scrach word
OD3C| E310                roxl.b         #1, d0              ;shift it into the save buffer
OD3E| B9CB                cmpa.l         a3, a4              ;assure a circular bufer
OD40| 6606                bne.s          @3                  ;
OD42| D7FC FFFF FF90      adda.l         #-HalfSize, a3      ;
OD48| 5341                @3      subq          #1, d1          ;decrement bit/byte counter
OD4A| 66EE                bne.s          @2                  ;loop again if still in byte

```

```

0D4C| E948          lsl.w          #4, d0          ;separate the nibbles
0D4E| E808          lsr.b          #4, d0          ;
0D50| 30C0          move          d0, (a0)+        ;save these nibbles
0D52| 5342          subq          #1, d2          ;decrement byte/SN counter
0D54| 66E0          bne.s          @1             ;loop again if still more to go
0D56| 4E75          rts
0D58|
0D58| 4B41 5300      Tag          .word          $4b41,$5300
0D5C|
0D5C|
0D5C|              .PAGE
0D5C|              ;-----
0D5C|              ; PARITY CIRCUITRY TEST
0D5C|              ; The purpose of this test is to verify the operation of the parity checking
0D5C|              ; logic by forcing a parity error and ensuring it is caught.
0D5C|              ; Register usage:
0D5C|              ;     D0 = pattern written          A0 = logical address used for test
0D5C|              ;     D1 = read results            A1 = corresponding physical address
0D5C|              ;     D2 = NMI indicator           A2 = save for NMI vector
0D5C|              ;     D3 = save of memory contents A3 = scratch
0D5C|              ;     D4 = save of error addr latch A4 = unused
0D5C|              ;     D5 = unused                  A5 = address of bus status register
0D5C|              ;     D6 = unused                  A6 = unused
0D5C|              ;-----
0D5C|
0D5C| PARTST
0D5C|              .ENDC
0D5C|              .IF ROM16K = 1
0D5C|
0D5C| MOVE.L 2478 007C  NMIVCT,A2          ;SAVE STANDARD NMI VECTOR
0D60| 47FA 0092      LEA          WWPERR,A3        ;THEN SET UP NEW PARITY ERROR (NMI) VECTOR
0D64| 21CB 007C      MOVE.L  A3,NMIVCT
0D68| 2A7C 00FC F801 MOVEA.L #STATREG,A5        ;setup status reg ptr for byte ops
0D6E| 4A39 00FC E01C TST.B  PAROFF          ;disable parity initially
0D74| 4282          CLR.L  D2          ;clear regs for result use
0D76| 4284          CLR.L  D4
0D78| 303C 01FF      MOVE          #$01FF,D0        ;SET UP PATTERN FOR WRITE
0D7C| 307C 0300      MOVEA     #$300,A0          ;SET UP ADDRESS FOR USE (in already verified mem)      RM000
0D80| 3610          MOVE          (A0),D3          ;SAVE ITS CONTENTS
0D82| 2248          MOVEA.L  A0,A1          ;COMPUTE CORRESPONDING
0D84| D3F8 02A4      ADDA.L  MINMEM,A1          ; PHYSICAL ADDRESS
0D88|
0D88| 4A39 00FC E006      TST.B  DG2ON          ;ENABLE WRITE WRONG PARITY FUNCTION
0D8E| 3080          MOVE          D0,(A0)          ;DO WRITE TO CREATE BAD PARITY
0D90| 4A39 00FC E004      TST.B  DG2OFF         ;DISABLE WWP
0D96|
0D96| 4A39 00FC E01E      TST.B  PARON          ;ENABLE PARITY ERROR DETECTION

```

```

0D9C| 4A42          TST      D2          ;SHOULDN'T HAVE INTERRUPT YET
0D9E| 6632          BNE.S    PARERR       ;EXIT IF ERROR
0DA0|
0DA0| 3210          MOVE     (A0),D1        ;DO READ - PARITY ERROR SHOULD OCCUR
0DA2| 4E71          NOP              ;GIVE A LITTLE EXTRA TIME
0DA4| 4A42          TST      D2          ;NMI RECEIVED?
0DA6| 672A          BEQ.S    PARERR       ;ERROR IF NO
0DA8|              ; Check that parity error and failing address correctly caught
0DA8| 0815 0001      BTST     #PBIT,(A5)   ;PARITY ERROR BIT SET?
0DAC| 6624          BNE.S    PARERR       ;EXIT IF NOT
0DAE| 3839 00FC F000 MOVE     MEALTCB,D4    ;GET ERROR ADDRESS
0DB4| 4A39 00FC E01C TST.B    PAROFF      ;disable parity to clear error bit
0DBA| EB8C          LSL.L    #5,D4        ;NORMALIZE THE ADDRESS
0DBC| B3C4          CMPA.L   D4,A1        ;SAME ADDRESS AS WRITTEN TO?
0DBE| 6612          BNE.S    PARERR       ;EXIT IF ERROR
0DC0| 21CA 007C      MOVE.L   A2,NMIVCT   ;ELSE RESTORE NMI VECTOR
0DC4| 4240          CLR      D0
0DC6| 4640          NOT      D0
0DC8| 3080          MOVE     D0,(A0)      ;"clear" bad parity
0DCA| 4A39 00FC E01E TST.B    PARON       ;reenable parity
0DD0| 6016          BRA.S    PARXIT      ;and skip to exit
0DD2|
0DD2|              ; Error exit
0DD2| 08C7 0003      PARERR  BSET     #PAR,D7    ;SET INDICATOR
0DD6| 4A87          TST.L    D7          ;in loop mode?
0DD8| 6B82          BMI.S    PARTST      ;restart if yes
0DDA| 4A39 00FC E01C TST.B    PAROFF      ;else ensure parity disabled
0DE0| 21CA 007C      MOVE.L   A2,NMIVCT   ;RESTORE NMI VECTOR
0DE4| 6000 05B4      BRA      TSTCHK      ;AND ABORT FURTHER TESTING
0DE8|
0DE8|              ; Normal exit
0DE8|
0DE8| 4A87          PARXIT  TST.L    D7          ;check for loop mode
0DEA| 6B00 FF70      BMI.S    PARTST      ;restart test if yes
0DEE| 6100 27C6      BSR      CHKCPU      ;place check over CPU (all tests OK)
0DF2| 600E          BRA.S    MEMTST2     ;else go do memory test
0DF4|
0DF4|              ; NMI routine for parity error checking
0DF4|
0DF4| 7401          WWPERR  MOVEQ    #1,D2      ;SET INDICATOR
0DF6| 4E73          RTE              ;AND RETURN
0DF8|
0DF8|              ;-----
0DF8|              ; Bus error handler for VIA #1 use
0DF8|              ;-----
0DF8|
0DF8| 7032          VIA1VCT MOVEQ    #EVIA1,D0    ;SET ERROR CODE

```



```

0DFA| 08C7 000A          BSET    #VIA1,D7          ;set indicator
0DFE| 6000 FB18          BRA      IOVCT             ;AND GO HANDLE I/O EXCEPTION
0E02|
0E02|          .ENDC                                ; (ROM16K)
0E02|          .PAGE
0E02|          ;-----
0E02|          ; Now do full memory test with and without parity enabled.  If parameter
0E02|          ; memory bit set for extended memory testing, memory tests executed in
0E02|          ; twice.  If warm-start, execute only one pass with parity enabled.
0E02|          ; Uses registers:
0E02|          ;   A0 = starting address to test      D0 = used to consolidate test results
0E02|          ;   A1 = ending address to test        D1 = scratch
0E02|          ;   A2 = unused                          D2 = address increment
0E02|          ;   A3 = save address for results      D3 = test results for each 128K
0E02|          ;   A4 = return address                D4 = max test address
0E02|          ;   A5 = unused                          D5 = pass count
0E02|          ;-----
0E02|
0E02|          MEMTST2
0E02|          .IF ROM4K = 0
0E02|
0E02|          .IF USERINT = 1
0E02| 327C 1E04          MOVEA    #MEMSTRT,A1          ;hilite memory board test icon
0E06| 6100 276C          BSR      INVICON
0E0A|          .ENDC
0E0A|
0E0A| 6100 F8D6          BSR      SETBUSVCT           ;restore normal bus error vector      RM000
0E0E|          MEMLOOP
0E0E| 43FA 0104          LEA      PRTYINT1,A1          ;setup up vector for parity intrpt      CHG015
0E12| 21C9 007C          MOVE.L   A1,NMIVCT           ;                                          CHG015
0E16|
0E16|          .IF ROM16K = 1
0E16|          ; First check if this is a warm-start      CHG006
0E16|
0E16| 0807 001E          BTST     #WRMSTRT,D7          ;warm-start?      CHG006
0E1A| 6704          BEQ.S      @0                      ;skip if not      CHG015
0E1C| 7A01          MOVEQ     #1,D5                    ;else set count for one pass      CHG015
0E1E| 6020          BRA.S      @3                      ;skip to do it      CHG015
0E20|
0E20|          ; Next check parameter memory to see if extended testing desired
0E20|
0E20| 6100 0A14          @0      BSR      CHKPM           ;go check parameter memory
0E24| 650E          BCS.S      @1                      ;skip if not valid to do only one pass
0E26| 0839 0006 00FC C18D      BTST     #6,MEMCODE     ;else check extended memory test indicator
0E2E| 6704          BEQ.S      @1                      ;exit if not set
0E30|
0E30| 7A02          MOVEQ     #2,D5                    ;run two passes for extended mode      CHG015

```

```

0E32| 6002          BRA.S  @2          ;go do it          CHG015
0E34| 7A01          @1      MOVEQ  #1,D5  ;run one pass for normal mode  CHG015
0E36|
0E36|          ; Run the memory tests
0E36|
0E36| 4A39 00FC E01C  @2      TST.B  PAROFF      ;first run with parity off      CHG015
0E3C| 612C          BSR.S  RUNTESTS      ;run test pass      CHG015
0E3E| 660E          BNE.S  TSTDONE      ;skip if error      CHG015
0E40| 4A39 00FC E01E  @3      TST.B  PARON      ;then run pass with parity on  CHG015
0E46| 6122          BSR.S  RUNTESTS      ;run test pass      CHG015
0E48| 6604          BNE.S  TSTDONE      ;exit if error      CHG015
0E4A| 5345          SUBQ   #1,D5      ;decr pass count      CHG015
0E4C| 66E8          BNE.S  @2          ;continue testing until done  CHG015
0E4E|
0E4E| 4A87          TSTDONE TST.L  D7      ;in loop mode?
0E50| 6BBC          BMI.S  MEMLOOP      ;restart if yes
0E52| 0807 0015      BTST   #MEM,D7      ;memory error?
0E56| 6600 0542      BNE    TSTCHK      ;abort if yes
0E5A| 6100 2762      BSR    CHKMBRD      ;else signal memory OK
0E5E| 47FA F8A4      LEA    NMI,A3      ;restore NMI vector      CHG015
0E62| 21CB 007C      MOVE.L  A3,NMIVCT    ;      CHG015
0E66| 6000 0198      BRA     IOTST      ;go on to next test
0E6A|
0E6A|
0E6A|          ;-----
0E6A|          ; Subroutine to run the memory tests - saves results as test proceeds
0E6A|          ; Zero condition code bit set if no errors.
0E6A|          ;-----
0E6A|
0E6A| RUNTESTS
0E6A|
0E6A|          ; Do the basic test
0E6A|
0E6A| BASICTEST
0E6A| 6118          BSR.S  TSTINIT      ;init for new test
0E6C|          CALL3
0E6C|          BSR4    RAMTEST
0E6C| 49FA 0006      #      LEA     @1,A4
0E70| 6000 003E      #      BRA     RAMTEST
0E74|          #@1
0E74| 6704          BEQ.S  @1          ;skip if no errors
0E76| 08C7 0015      BSET   #MEM,D7      ;else set error indicator
0E7A| 611C          @1      BSR.S  SAVRSLT      ;save results
0E7C| 66EE          BNE.S  CALL3      ;loop until done      CHG021
0E7E| 0807 0015      BTST   #MEM,D7      ;set condition code      CHG015
0E82| 4E75          RTS          ;and exit
0E84|

```

```

OE84|          .ELSE
OE84|          .ENDC          ;{ROM16K}
OE84|
OE84|          .PAGE
OE84|          ;-----
OE84|          ; Subroutine to do initialization for memory tests
OE84|          ;-----
OE84|
OE84| 7402          MOVEQ    #2,D2          ;test in 128K increments          RM000
OE86| 4842          SWAP     D2          ; (sets D2 = $20000)          RM000
OE88| 2838 0110      MOVE.L   SCRNBASE,D4      ;get max test address (base of screen)
OE8C| 307C 0800      MOVEA   #LOMEM,A0        ;set initial start
OE90| 2242          MOVE.L   D2,A1          ; and ending address
OE92| 367C 0186      MOVEA   #MEMRSLT,A3      ;set address of result area          RM000
OE96| 4E75          RTS
OE98|
OE98|          ;-----
OE98|          ; Subroutine to save results and update ptrs.
OE98|          ;-----
OE98|
OE98| 3003      SAVRSLT MOVE    D3,D0          ;get low results
OE9A| 4843          SWAP     D3          ;get high results
OE9C| 8640          OR       D0,D3          ;combine
OE9E| 875B          OR       D3,(A3)+        ; and save
OEA0| B889          CMP.L    A1,D4          ;at max test address?
OEA2| 670A          BEQ.S    @1          ;exit if yes
OEA4| 2049          MOVEA.L  A1,A0          ;else set new addresses
OEA6| D3C2          ADDA.L   D2,A1          ; to check next row of memory
OEA8| B889          CMP.L    A1,D4          ;in last segment?
OEAA| 6C02          BGE.S    @1
OEAC| 2244          MOVE.L   D4,A1          ;set at base of video page
OEAE| 4E75          @1      RTS
OEB0|
OEB0|          .PAGE
OEB0|          ;-----
OEB0|          ; BASIC MEMORY TEST - writes pattern and its complement in memory location,
OEB0|          ; then verifies by reading. Also does second scan as
OEB0|          ; addressing check. Uses long word operations for speed.
OEB0|          ; Inputs:
OEB0|          ;     A0 - Starting address to test
OEB0|          ;     A1 - Ending address
OEB0|          ;     A4 - Return address
OEB0|          ; Outputs:
OEB0|          ;     CCR zero bit set if no error
OEB0|          ;     D3 = OR mask of errors
OEB0|          ; Uses registers:
OEB0|          ;     A0 = current test address          D0 = current test pattern

```

```

OEB0|          ;      A1 = ending test address      D1 = scratch
OEB0|          ;      A2 = unused                    D2 = unused
OEB0|          ;      A3 = unused                    D3 = OR mask of errors
OEB0|          ;      A4 = return address            D4 = unused
OEB0|          ;      A5 = saved start address        D5 = unused
OEB0|          ;      A6 = used for return address    D6 = unused
OEB0|          ;-----
OEB0|
OEB0| 2A48      RAMTEST MOVE.L  A0,A5                ;save start address
OEB2| 203C AA55 A55A      MOVE.L  #PATRN,D0          ;get pattern
OEB8| 4680      NOT.L    D0                        ;use complement first
OEBA| 7600      MOVEQ   #0,D3                      ;clear for result use
OECB| 007C 0010      ORI     #$0010,SR              ;set extend bit for use with pattern rotate
OEC0|
OEC0| 2080      RAMRW   MOVE.L  D0,(A0)              ;do write
OEC2| B090      CMP.L   (A0),D0                    ;verify
OEC4| 6706      BEQ.S   RAMCHK2                    ;skip if OK
OEC6|          BSR6    RDERR                        ;else save error bits
OEC6| 4DFA 0004      #      LEA      @1,A6
OECA| 6040      #      BRA.S    RDERR
OECB|          #@1
OECB|
OECB| 4680      RAMCHK2 NOT.L  D0                    ;now use inverse
OECB| 2080      MOVE.L  D0,(A0)                    ;write to check for stuck bits
OED0| B098      CMP.L   (A0)+,D0                  ;verify and bump address
OED2| 670A      BEQ.S   RAMNXT                      ;skip if OK
OED4| 5988      SUBQ.L  #4,A0                      ;else get error address
OED6|          BSR6    RDERR                        ;go save error bits
OED6| 4DFA 0004      #      LEA      @1,A6
OEDA| 6030      #      BRA.S    RDERR
OEDC|          #@1
OEDC| 5888      ADDQ.L  #4,A0                      ;and restore next test address
OEDE|
OEDE| E390      RAMNXT ROXL.L  #1,D0                ;create new pattern
OEE0| 4680      NOT.L   D0                        ;invert for test
OEE2| B3C8      CMPA.L  A0,A1                      ;done?
OEE4| 66DA      BNE.S   RAMRW                      ;loop if not
OEE6|
OEE6|          ; Now do address check - writes memory as all F's during scan
OEE6| 203C AA55 A55A      ADRTST MOVE.L  #PATRN,D0          ;reinitialize
OEEC| 204D      MOVE.L  A5,A0                      ;get start address
OEEE| 7200      MOVEQ   #0,D1                      ;
OEF0| 4681      NOT.L   D1                        ;final pattern for write
OEF2| 007C 0010      ORI     #$0010,SR              ;set extend
OEF6|
OEF6| B090      ADRCHK  CMP.L   (A0),D0              ;check contents
OEF8| 6706      BEQ.S   ADRCLR                      ;skip if OK

```



```

0EFA|          BSR.S  RDERR          ;else save errors
0EFA| 4DFA 0004      #          LEA          @1,A6
0EFE| 600C          #          BRA.S      RDERR
0F00|          #@1
0F00| 20C1          ADRCLR MOVE.L  D1,(A0)+      ;'clear' and go to next location
0F02| E390          ROXL.L  #1,D0              ;create next pattern
0F04| B3C8          CMPA.L  A0,A1              ;done?
0F06| 66EE          BNE.S   ADRCHK              ;loop if not
0F08|
0F08|          ; Check results
0F08|
0F08| 4A83          TST.L    D3                  ;set condition codes
0F0A|          RTS4
0F0A| 4ED4          #          JMP          (A4)
0F0C|
0F0C|          ; Failure routine - save results and continue testing
0F0C|
0F0C| 2210          RDERR MOVE.L  (A0),D1          ;do read again
0F0E| B181          EOR.L    D0,D1              ;isolate bad bits
0F10| 8681          OR.L     D1,D3              ;save result
0F12|          RTS6                  ;and return
0F12| 4ED6          #          JMP          (A6)
0F14|
0F14|          .PAGE
0F14|          ;-----
0F14|          ; Phase 1 Parity error handler for memory tests. Objective for handler is to
0F14|          ; isolate parity error to chip level.
0F14|          ; Assumes:
0F14|          ;     D0 = expected data pattern
0F14|          ;     A0 = error address or address + 4
0F14|          ; Uses registers:
0F14|          ;     D1 = parity error address
0F14|          ;     D2 = search size for byte in error
0F14|          ;     D3 = low memory address
0F14|          ;     A1 = search address
0F14|          ;-----
0F14|          PRTYINT1
0F14| 6152          BSR.S   TSTSTAT          ;check if parity error          CHG015
0F16| 6600 F7EC          BNE     NMI          ;skip if not          CHG015
0F1A| 08C7 0016          BSET    #MPAR,D7      ;set error indicator          CHG015
0F1E| 21C0 026C          MOVE.L  D0,XPCDATA      ;save data and address          CHG015
0F22| 21C8 0268          MOVE.L  A0,XPCTADDR      ;          CHG015
0F26| 2638 02A4          MOVE.L  MINMEM,D3        ;get low memory address          CHG015
0F2A| 6100 00C4          BSR     GETPADDR        ;read and convert parity address  CHG015
0F2E| 0801 0005          BTST    #5,D1          ;main mem error?          CHG015
0F32| 6604          BNE.S   @1              ;skip if not          CHG015

```

```

0F34|
0F34| 743F          MOVEQ  #MSRCHSZ-1,D2  ;setup up search size for main mem  CHG015
0F36| 600A          BRA.S   @2             ;skip to do it                    CHG015
0F38| 343C 7FFF      @1    MOVE  #VSRCHSZ-1,D2 ;setup for video memory search      CHG015
0F3C| 0281 FFFF 8000      ANDI.L #VMSK,D1    ;mask off undefined info          CHG015
0F42| 21C1 01A6      @2    MOVE.L D1,PEADDR    ;save error address                CHG015
0F46|
0F46|          ; Reset NMI vector and start search for exact address          CHG015
0F46|
0F46| 43FA 002A      LEA     PRTYINT2,A1      ;setup new vector                CHG015
0F4A| 21C9 007C      MOVE.L A1,NMIVCT        ;                               CHG015
0F4E| 9283          SUB.L   D3,D1            ;convert to logical address        CHG015
0F50| 2241          MOVE.L   D1,A1            ;setup for use                    CHG015
0F52| 4A39 00FC E01C   TST.B  PAROFF         ;clear parity bit                CHG015
0F58| 4A39 00FC E01E   TST.B  PARON          ;                               CHG015
0F5E| 4284          CLR.L   D4              ;clear for use                    CHG015
0F60|
0F60| 1819          @3    MOVE.B (A1)+,D4      ;search for parity error by byte    CHG015
0F62| 51CA FFFC      DBRA    D2,@3            ;loop until found                 CHG015
0F66|
0F66|          ; Error did not repeat                                          CHG015
0F66| 605E          BRA.S   PRXIT             ;go save error info and exit       CHG015
0F68|
0F68|          ;-----
0F68|          ; Subroutine to check for parity error
0F68|          ;-----
0F68|
0F68| 0839 0001 00FC F801 TSTSTAT BTST  #1,STATREG ;check for parity error          CHG015
0F70| 4E75          RTS                     ;return with condition code set    CHG015
0F72|
0F72|          ;-----
0F72|          ; Parity error handler, phase 2.
0F72|          ; Assumes:
0F72|          ;     A1 = error address + 1
0F72|          ;     D0 = expected data (long)
0F72|          ;     D4 = error data (byte)
0F72|          ; Uses registers:
0F72|          ;     D1 = error address
0F72|          ;     D2 = scratch
0F72|          ;-----
0F72|
0F72|          PRTYINT2
0F72| 61F4          BSR.S   TSTSTAT           ;parity error?                    CHG015
0F74| 6600 F78E      BNE     NMI              ;skip if not to handle NMI        CHG015
0F78| 6100 0076      BSR     GETPADDR         ;get error address                CHG015
0F7C| 21C1 0278      MOVE.L  D1,PEADR2        ;save it                          CHG015
0F80| 93FC 0000 0001   SUBA.L #1,A1           ;get actual address                CHG015

```

```

0F86| 21C9 0270          MOVE.L  A1,ACTADDR      ;save address and data      CHG015
0F8A| 21C4 0274          MOVE.L  D4,ACTDATA      ;                          CHG015
0F8E| 0801 0005          BTST      #5,D1          ;video error?          CHG015
0F92| 6632                BNE.S    PRIXIT        ;skip if yes              CHG015
0F94|
0F94| 2209                MOVE.L  A1,D1          ;get error address        CHG015
0F96| 0281 0000 0003      ANDI.L  #ADRMASK,D1      ;setup up rotate count    CHG015
0F9C| 2401                MOVE.L  D1,D2          ;save it                  CHG015
0F9E| 6706                BEQ.S    @2          ;skip if pre-rotate not needed CHG015
0FA0|
0FA0| E188                @1      LSL.L    #8,D0          ;shift expected data to high byte CHG015
0FA2| 5341                SUBQ   #1,D1          ;                          CHG015
0FA4| 66FA                BNE.S    @1          ;                          CHG015
0FA6|
0FA6| E198                @2      ROL.L    #8,D0          ;shift to low byte        CHG015
0FA8| 0280 0000 00FF      ANDI.L  #$FF,D0        ;strip unneeded info      CHG015
0FAE| B900                EOR.B   D4,D0        ;isolate bad bits         CHG015
0FB0| 671E                BEQ.S    PCERR        ;skip if no data error    CHG015
0FB2| 0802 0000          BTST      #0,D2          ;check if high or low byte error CHG015
0FB6| 6602                BNE.S    @3          ;skip if low byte        CHG015
0FB8| E148                LSL     #8,D0        ;else shift to high byte   CHG015
0FBA|
0FBA| 367C 0186          @3      MOVEA   #MEMRSLT,A3      ;set ptr to save area      CHG015
0FBE| 2809                MOVE.L  A1,D4          ;set error address        CHG015
0FC0| 2600                MOVE.L  D0,D3          ;and error bits           CHG015
0FC2| 6100 F8B0          BSR     SCRNSAV      ;then go save data        CHG015
0FC6|
0FC6| 4A39 00FC E01C      PRIXIT  TST.B    PAROFF      ;disable parity            CHG015
0FCC| 6000 F796          BRA     EXCP1        ;and go to exit            CHG015
0FD0|
0FD0|                      ; no data error - must be parity chip failure; decode to chip id CHG015
0FD0|
0FD0| 2209                PCERR   MOVE.L  A1,D1          ;get error address        CHG015
0FD2| 0801 0000          BTST      #0,D1          ;check if odd or even      CHG015
0FD6| 6708                BEQ.S    @1          ;skip if even              CHG015
0FD8| 11FC 0014 027D      MOVE.B   #$14,PCHIP      ;bad parity chip in low word CHG015
0FDE| 6006                BRA.S    @2          ;                          CHG015
0FE0| 11FC 0009 027D      @1      MOVE.B   #9,PCHIP      ;bad chip in high word     CHG015
0FE6|
0FE6| 7411                @2      MOVEQ   #17,D2        ;calculate row address     CHG015
0FE8| E4A9                LSR.L    D2,D1          ; for parity error         CHG015
0FEA| 11C1 027C          MOVE.B   D1,PCHPROW      ;save row info             CHG015
0FEE| 60D6                BRA.S    PRIXIT        ;and exit                  CHG015
0FF0|
0FF0|                      ;-----
0FF0|                      ; Subroutine to get parity error address
0FF0|                      ; Returns D1 = error address

```

```

OFF0|                                     ;-----
OFF0|
OFF0| GETPADDR
OFF0| 4281          CLR.L   D1          ;clear for use          CHG015
OFF2| 3239 00FC F000 MOVE    MEALTCH,D1      ;read error latch      CHG015
OFF8| 31C1 01AA     MOVE    D1,ADRLTCH      ;save it                CHG015
OFFC| EB89          LSL.L   #5,D1          ;convert to physical address CHG015
OFFE| 4E75          RTS                      ;                CHG015
1000|
1000|          .PAGE
1000|
1000|                                     ;-----
1000| ; Continue with I/O board testing
1000|                                     ;-----
1000|
1000| IOTST
1000|          .ELSE          ;{ROM4K}
1000|          .ENDC          ;{ROM4K}
1000|
1000|          .IF USERINT = 1
1000| 327C 1E12     MOVEA    #IOSTRT,A1      ;hilite I/O board test icon
1004| 6100 256E     BSR      INVICON
1008|          .ENDC
1008|
1008|          .IF FULLSCC = 1
1008|                                     ;-----;
1008| ; SCC Test (Checks RS232 port controller)
1008| ;
1008| ; The SCC interrupt vector is written and read with all 8 bit patterns
1008| ; to check SCC addressing. An internal loopback test is then done on
1008| ; channel B. RM014
1008| ;
1008| ; The chip is always left in an initial state as follows:
1008| ; both channels are reset
1008| ; master interrupt enable is reset
1008| ; DTR, RTS outputs set high on channel B CHG011
1008| ;
1008| ; Runs with interrupts off, uses stack. Uses registers:
1008| ;
1008| ; A0 = SCC address          D0 = error indicator
1008| ; A2 = scratch              D1 = scratch
1008| ;                          D2 = scratch
1008| ;                          D3 = scratch
1008| ;
1008| ; Errors saved in D0 and stored in low memory as follows:
1008| ;
1008| ; 0000 0001 -> SCC vector read/write error (accessed via channel A) RM014
1008| ; 0000 0010 -> channel B transmit buffer empty timeout RM014

```

```

1008|          ;          0000 0100 -> channel B receive buffer full timeout          RM014
1008|          ;          0000 1000 -> channel B data compare error                  RM014
1008|          ;
1008|          ;-----;
1008|
1008| 47FA 00E4      SCCTEST LEA      SCCVCT,A3          ;set up bus error vector
100C| 21CB 0008          MOVE.L   A3,BUSVCTR
1010| 6100 00BE          BSR      RSTSCC          ;reset and set up A0 for SCC
1014| 5488          ADDQ.L   #ACTL,A0          ;adjust SCC address for channel A
1016| 7200          MOVEQ    #0,D1          ;SCC interrupt vector starts out 0
1018| 7000          MOVEQ    #0,D0          ;no errors
101A|
101A|          VECTLOOP
101A| 10BC 0002          MOVE.B   #2,(A0)          ;test scc write register 2 (interrupt vector)
101E|          ; via channel A
101E| 3E97          MOVE      (SP),(SP)          ;delay
1020| 1410          MOVE.B   (A0),D2          ;read unmodified vector
1022| B401          CMP.B    D1,D2          ;ok?
1024| 6704          BEQ.S    @1          ;branch if so
1026| 7001          MOVEQ    #1,D0          ;otherwise set error code
1028| 6064          BRA.S    SCCEXIT          ;and exit
102A| 3E97          @1      MOVE      (SP),(SP)
102C| 10BC 0002          MOVE.B   #2,(A0)          ;write next vector value
1030| 5281          ADDQ.L   #1,D1          ;increment and delay
1032| 1081          MOVE.B   D1,(A0)          ;write it
1034| 66E4          BNE.S    VECTLOOP          ;go through 256 values
1036| 6010          BRA.S    SETSCC          ;now go do loopback init
1038|
1038|          ;-----;
1038|          ; Now init channel B for max baud rate and internal loopback.
1038|          ; External transmit is inhibited by setting DTR low.
1038|          ;-----;
1038|
1038|          ; Initialization data for SCC: max baud RS-232 async communication
1038|
1038|          b96data:
1038| 09 00          .byte     9,$00          ;disable all interrupts          RM014
103A| 04 4D          .byte     4,$4D          ;x16 clk, 2 stop bits, odd parity
103C| 0B 50          .byte    11,$50          ;baud rate gen clk to receiver, transmitter
103E| 0C 00          .byte    12,$00          ;set baud rate to max
1040| 0D 00          .byte    13,$00
1042| 0E 13          .byte    14,$13          ;enable baud rate gen, BR=PCLK, loopback
1044| 03 C1          .byte     3,$C1          ;8 bits/char recv, enable receiver
1046| 05 EA          .byte     5,$EA          ;DTR low, 8 bits/char xmit, enable xmit, CRC          RM014
1048| 0000 0010      b96lth .equ     *-b96data
1048|
1048| 45FA FFEE      SETSCC LEA      B96DATA,A2          ;setup channel B          RM014

```

```

104C| 323C 0010          MOVE.W #B96LTH,D1
1050| 5588          SUBQ.L #ACTL,A0          ;set address for channel B
1052| 616A          BSR.S WRITESCC          ;
1054|          ; do the loopback test
1054|          ;
1054| 7200          LPTEST MOVEQ #0,D1          ;go thru 256 bytes
1056| 76FF          MOVEQ #-1,D3          ;set up timeout count
1058|          SCCLOOP
1058| 0810 0002          BTST #TXBE,(A0)          ;wait for transmit buffer empty
105C| 6608          BNE.S SCCOUT
105E| 51CB FFF8          DBRA D3,SCCLOOP
1062| 5440          ADDQ #2,D0
1064| 6024          BRA.S SCCLXIT          ;report timeout error
1066| 3E97          SCCOUT MOVE (SP),(SP)
1068| 1141 0004          MOVE.B D1,SCCDATA(A0)
106C|
106C|          SCCLOOP2
106C| 0810 0000          BTST #RXBF,(A0)          ;wait for data byte to come in
1070| 6608          BNE.S SCCIN
1072| 51CB FFF8          DBRA D3,SCCLOOP2
1076| 5840          ADDQ #4,D0
1078| 6010          BRA.S SCCLXIT
107A| 3E97          SCCIN MOVE (SP),(SP)
107C| 1428 0004          MOVE.B SCCDATA(A0),D2
1080| B401          CMP.B D1,D2
1082| 6608          BNE.S SCCLERR
1084| 76FF          MOVEQ #-1,D3          ;update timeout count
1086| 5201          ADDQ.B #1,D1          ;increment data
1088| 66CE          BNE.S SCCLOOP          ;just do it 256 times
108A|
108A| 6002          SCCLXIT BRA.S SCCEXIT
108C|
108C| 5040          SCCLERR ADDQ #8,D0
108E|
108E|          ; exit, saving errors
108E|
108E| 11C0 02AC          SCCEXIT MOVE.B D0,SCCRSLT          ;save results
1092| 6720          BEQ.S @3          ;continue if OK
1094| 0800 0000          BTST #0,D0          ;check for chan A error
1098| 6704          BEQ.S @1
109A| 08C7 000F          BSET #RS232A,D7
109E| E248          @1 LSR #1,D0          ;check for chan B error
10A0| 4A00          TST.B D0
10A2| 6704          BEQ.S @2
10A4| 08C7 0010          BSET #RS232B,D7
10A8| 6126          @2 BSR.S RSTSCC          ;leave SCC at initial condition
10AA| 4A87          TST.L D7          ;looping required?

```

```

10AC| 6B00 FF5A          BMI.S  SCCTEST      ;restart test if yes
10B0| 6000 02E8          BRA     TSTCHK      ;else go report error
10B4|
10B4| 611A              @3    BSR.S  RSTSCC      ;leave SCC at initial condition
10B6| 4A87              TST.L   D7           ;in loop mode?
10B8| 6B00 FF4E          BMI.S  SCCTEST      ;restart test if yes
10BC| 604E              BRA.S   DSKTST      ;else continue to next test
10BE|
10BE|                  .PAGE
10BE|
10BE| ;-----
10BE| ; WRITESCC: used to initialize a series of SCC registers.
10BE| ;
10BE| ;     A0 = SCC address for channel to be initialized
10BE| ;     A2 = pointer to an initialization data block as above
10BE| ;     A4 = return address
10BE| ;     D1 = initialization data block size in bytes
10BE| ;
10BE| ;     A2, D1, D2 are modified.
10BE| ;
10BE| ;-----
10BE| WRITESCC
10BE|          MOVE.B  (A0),D2      ;read to make sure SCC is sync'ed up
10C0|          BRA.S  @2          ;delay for timing, too
10C2|          @1    MOVE.B  (A2)+,(A0)
10C4|          @2    DBRA    D1,@1
10C8|          RTS
10CA|
10CA| ;-----
10CA| ; Subroutine to initialize SCC.  Does reset and zeroes interrupt vector.
10CA| ;-----
10CA| INITBDATA
10CA| 02 00          .BYTE  2,$00      ;zero interrupt vector
10CC| 09 C0          .BYTE  9,$C0      ;reset both channels
10CE| 0000 0004      INITBLTH .EQU   4          ;
10CE|
10CE| 05 82          INITB2  .BYTE  5,$82      ;set DTR, RTS high for Applebus
10D0| 0000 0002      INITB2L .EQU   2          ;
10D0|
10D0| RSTSCC
10D0| 207C 00FC D241      MOVE.L  #SCCBCTL,A0      ;point to SCC base address (chan B)
10D6| 45FA FFF2          LEA     INITBDATA,A2      ;point to channel B init data
10DA| 7204          MOVEQ    #INITBLTH,D1      ; and set up the length
10DC| 61E0          BSR.S   WRITESCC      ;then init channel B
10DE| 700C          MOVEQ    #12,D0          ;delay for SCC reset
10E0| 6100 FA00          BSR     DELAY

```

RM014

```

10E4|
10E4| 45FA FFE8          LEA     INITB2,A2          ;setup DTR, RTS outputs          CHG011
10E8| 7202              MOVEQ   #INITB2L,D1        ;                      CHG011
10EA| 61D2              BSR.S   WRITESCC          ;                      CHG011
10EC| 4E75              RTS                      ;and return
10EE|
10EE|          ;-----
10EE|          ;  Bus error routine for SCC testing
10EE|          ;-----
10EE|
10EE| B1FC 00FC D241      SCCVCT  CMPA.L  #SCCBCTL,A0      ;accessing channel B?
10F4| 6604              BNE.S   @1                  ;skip if no
10F6| 7038              MOVEQ   #ERS232B,D0          ;set error code for chan B
10F8| 6002              BRA.S   @2
10FA| 7037              @1    MOVEQ   #ERS232A,D0      ;set error code for chan A
10FC| 4A87              @2    TST.L   D7                ;check if in loop mode
10FE| 6A08              BPL.S   @3                  ;skip if not
1100| 3E7C 0480          MOVEA   #STKBASE,SP          ;else restore stack ptr          RM000
1104| 6000 FF02          BRA     SCCTEST              ;and restart test
1108| 6000 F80E          @3    BRA     IOVCT              ;and go handle I/O card bus error
110C|
110C|
110C|          .ENDC
110C|          .PAGE
110C|
110C|          ;-----
110C|          ;  Test of disk interface - ensure R/W capability to shared RAM, then
110C|          ;  try disable interrupts command. This test will also verify
110C|          ;  the results of the disk controller's own self-test (ROM and RAM test).
110C|          ;-----
110C|
110C|          DSKTST
110C|          .IF  DIAGS = 1
110C|
110C| 47FA 0078          LEA     DSKVCT,A3          ;set up vector in case of bus timeout
1110| 21CB 0008          MOVE.L  A3,BUSVCTR
1114| 207C 00FC C001      MOVE.L  #DISKMEM,A0          ;set ptr for shared memory
111A|
111A|          ;  Display ROM id                      CHG001
111A|
111A| 7A03              MOVEQ   #ROMIDROW,D5          ;set cursor ptrs          CHG001
111C| 3C3C 0051          MOVE    #ROMIDCOL+1,D6        ;                      CHG001
1120| 702F              MOVEQ   #'/',D0              ;preced with / char          CHG001
1122| 6100 2616          BSR     DSPVAL              ;display it                  CHG001
1126| 1028 0030          MOVE.B  ROMV(A0),D0          ;read id                    CHG001
112A| 11C0 02A1          MOVE.B  D0,IOROM              ;save in low memory          CHG010
112E| 7202              MOVEQ   #2,D1                ;                      CHG001
1130| 6100 0546          BSR     OUTCH              ;                      CHG001

```



```

1134|
1134|           ; Read system type                                CHG009
1134|
1134| 6162           BSR.S   SETTYPE           ;determine system type      CHG029
1136|
1136|           ; Check disk alive indicator
1136|
1136| 4282           CLR.L   D2               ;clear for use                CHG022
1138| 227C 00FC D901  MOVE.L   #VIA2BASE,A1       ;set ptr to parallel port 6522
113E| 0229 00BF 0010  ANDI.B   #$BF,DDRB2(A1)    ;ensure bit 6 is input
1144| 203C 001C 8000  MOVE.L   #DSKTMOUT,D0       ;set up timeout count for 15 secs
114A| 0811 0006      @2      BTST   #DSKDIAG,IRB2(A1) ;check indicator
114E| 6606           BNE.S   @3               ;skip if set
1150| 5380           SUBQ.L   #1,D0             ;else loop until timeout (about 8 us per loop)
1152| 66F6           BNE.S   @2               ;
1154| 7439           MOVEQ    #EDISK,D2         ;error if not set          CHG022
1156|
1156|           ; Try read operation and check results of self-test
1156|
1156|           @3
1156|           .IF  DIAGS = 1
1156| 11E8 0016 02AE  MOVE.B   STST(A0),DSKRSLT ;get results of disk self-test  CHG022
115C| 6616           BNE.S   INTERR           ;exit if error              CHG022
115E|
115E|           @4      TST.B   D2             ;previous error?            CHG022
1160| 6612           BNE.S   INTERR           ;exit if yes                CHG022
1162|
1162|           ; Then try simple write operation to shared RAM
1162|
1162| 7055           MOVEQ    #$55,D0          ;set up pattern              RM000
1164| 1140 0002  MOVE.B   D0,CMD(A0)          ;try write
1168| B028 0002  CMP.B    CMD(A0),D0          ;verify
116C| 6606           BNE.S   INTERR           ;exit if error
116E|
116E|           ; Finally try a command to disable interrupts
116E|
116E| 6100 0BD6      BSR      DSABLDISK        ;go issue disable cmd
1172| 640C      BCC.S   DSKXIT                ;skip if OK
1174|           .ELSE
1174|           .ENDC
1174|
1174| 08C7 0011  INTERR  BSET   #DISK,D7       ;else set disk error
1178| 4A87      TST.L   D7                   ;restart if in loop mode
117A| 6B90      BMI.S   DSKTST                ;
117C| 6000 021C      BRA     TSTCHK           ;and abort further testing
1180|
1180| 4A87      DSKXIT  TST.L   D7             ;restart if in loop mode

```

```

1182| 6B88          BMI.S   DSKTST
1184|              .ENDC
1184|
1184| 603A          BRA.S   COPSCHK          ;else go to next test
1186|              ;-----
1186|              ;  Bus error routine for disk testing
1186|              ;-----
1186| 7039          DSKVCT  MOVEQ  #EDISK,D0          ;SET ERROR CODE
1188|
1188|              .IF  ROM4K = 0
1188| 4A87          TST.L   D7              ;check if in loop mode
118A| 6A08          BPL.S   @3              ;skip if not
118C|              .ENDC
118C|
118C| 3E7C 0480          MOVEA  #STKBASE,SP          ;else restore stack ptr          RM000
1190| 6000 FF7A          BRA    DSKTST          ;and restart test
1194| 6000 F782          @3    BRA    IOVCT          ;GO HANDLE I/O CARD BUS ERROR
1198|
1198|              ;-----
1198|              ;  Subroutine for determining system type
1198|              ;  Returns type value in D0 and sets SYSTYPE location in memory
1198|              ;  D0 = 0 - Lisa 1
1198|              ;      1 - Lisa 2/external disk with slow timers
1198|              ;      2 - Lisa 2/external disk with fast timers
1198|              ;      3 - Lisa 2/internal disk (Pepsi) with fast timers
1198|              ;-----
1198|
1198| 4280          SETTYPE CLR.L   D0              ;clear for type usage          CHG029
119A| 1239 00FC C031      MOVE.B  DISKROM,D1          ;read disk id          CHG029
11A0| 4A01          TST.B   D1              ;check for Lisa 1          CHG029
11A2| 6A16          BPL.S   @9              ;skip if yes          CHG029
11A4| 0801 0005          BTST  #SLOTMR,D1          ;Lisa 2 with slow timers?          CHG029
11A8| 6704          BEQ.S   @1              ;skip if not          CHG029
11AA| 7001          MOVEQ   #1,D0          ;else set type          CHG029
11AC| 600C          BRA.S   @9              ;          CHG029
11AE| 0801 0006          @1    BTST  #FASTMR,D1          ;Lisa 2 with fast timers?          CHG029
11B2| 6704          BEQ.S   @2              ;skip if not          CHG029
11B4| 7002          MOVEQ   #2,D0          ;else set type          CHG029
11B6| 6002          BRA.S   @9              ;          CHG029
11B8| 7003          @2    MOVEQ   #3,D0          ;else must be Pepsi with fast timers          CHG029
11BA| 11C0 02AF          @9    MOVE.B  D0,SYSTYPE          ;save system type          CHG029
11BE| 4E75          RTS              ;          CHG029
11C0|
11C0|              .PAGE
11C0|
11C0|              ;-----
11C0|              ;  Scan the keyboard for user commands. Click speaker first to alert user.
11C0|              ;-----

```

```

11C0|
11C0| 47FA F754          COPSCHK LEA      COPSVCT,A3          ;set up bus error vector
11C4| 21CB 0008          MOVE.L   A3,BUSVCTR
11C8| 6100 F924          BSR       CLICK          ;notify user that keyboard about to be scanned
11CC| 6100 F8FE          BSR       DELAY_1        ;delay for 1/10 sec
11D0| 207C 00FC DD81    MOVEA.L   #VIA1BASE,A0        ;set up VIA address
11D6| 117C 00C9 0018    MOVE.B    #$C9,PCR1(A0)      ;set intrpt control for later use
11DC|                                     ; also causes second "click"
11DC|
11DC| 6104              BSR.S     SCANCPS        ;go check for keyboard input
11DE| 6000 00AC          BRA       CLKTST          ;and continue on
11E2|
11E2| ;-----
11E2| ; Subroutine to do scan of keyboard COPS
11E2| ;-----
11E2|
11E2| SCANCPS
11E2| 2278 0260          MOVE.L   KBDQPTR,A1        ;set up queue ptrs
11E6| 347C 02C0          MOVEA    #QEND,A2
11EA|
11EA| ; Scan for keyboard data
11EA|
11EA| 6100 F892          KEYSCAN BSR      GETDATA          ;go check for keyboard input
11EE| 6568              BCS.S     @9              ;exit if no data or queue full
11F0| 0C00 00FF          CMPI.B   #CMDKEY,D0        ;is it the command key?
11F4| 6624              BNE.S     @1              ;skip if no
11F6| 6100 F886          BSR      GETDATA          ;yes - get next char to see if boot cmd
11FA| 655C              BCS.S     @9              ;exit if queue full or no more data
11FC|
11FC| 0C00 00FE          CMPI.B   #SHFTKEY,D0        ;check for shift key
1200| 6636              BNE.S     @2              ;skip if no - go save as boot code
1202| 6100 F87A          BSR      GETDATA          ;else keep checking for command sequence
1206| 6550              BCS.S     @9              ;skip if Q full or no data
1208| 0C00 00C4          CMPI.B   #PKEY,D0         ;'P' key for power-cycling
120C| 660C              BNE.S     @1              ;skip if not
120E| 11FC 000F 01B3    MOVE.B    #PC,BOOTDVCE        ;set for power-cycle mode
1214| 08C7 001C          BSET     #ALTB00T,D7        ;set alternate boot
1218| 60D0              BRA.S     KEYSCAN          ;and continue scan
121A|
121A| @1
121A| .IF USERINT = 1
121A|
121A| ; do test for downstroke or mouse button (used for burnin cycling)
121A|
121A| 4A00              TST.B     D0              ;check keycode
121C| 6A1E              BPL.S     @4              ;skip if not downstroke
121E| 0C00 00FD          CMPI.B   #ALPHKEY,D0        ;ignore alpha lock key

```

```

1222| 67C6          BEQ.S   KEYSCAN
1224|              .ENDC
1224|
1224|              .IF  BURNIN = 1
1224| 1224| 0C00 0086    CMP.B   #MOUSDWN,D0      ;mouse button?
1228| 6608          BNE.S   @3              ;skip if not
122A| 08F8 0002 02A2 BSET    #MSBUTN,STATFLGS ;else set flag for later use
1230| 60B8          BRA.S   KEYSCAN          ;and continue scan
1232|              .ENDC
1232|
1232|              @3
1232| 1232| 08C7 001D    BSET    #BTMENU,D7      ;set indicator for boot menu
1236|              .ENDC
1236|
1236| 60B2          BRA.S   KEYSCAN          ;and continue scan
1238|
1238|              ; Save code as possible boot id and set indicator
1238|
1238| 6124          @2      BSR.S   XLATE          ;translate to boot id code and save
123A| 60AE          BRA.S   KEYSCAN          ;and continue keyboard scan
123C|
123C|              ; Check if release of mouse or COMMAND key (in case continuing after error)
123C|
123C| 123C| 0C00 0006    @4      CMP.B   #MOUSUP,D0      ;mouse release?
1240| 6608          BNE.S   @5
1242| 08B8 0004 02A2 BCLR    #MOUSE,STATFLGS ;clear marker if yes
1248| 60A0          BRA.S   KEYSCAN          ;and continue scan
124A|
124A| 124A| 0C00 007F    @5      CMP.B   #CMDUP,D0      ;Left CMD key release?
124E| 6606          BNE.S   @6
1250| 08B8 0003 02A2 BCLR    #CMDFLG,STATFLGS ;clear marker if yes
1256|
1256| 6092          @6      BRA.S   KEYSCAN          ;continue scan
1258|
1258| 21C9 0260    @9      MOVE.L  A1,KBDQPTR      ;save buffer ptr
125C| 4E75          RTS              ;and return to caller
125E|
125E|              ;-----
125E|              ; Subroutine to translate keycodes to boot device codes. Returns
125E|              ; with boot code in D2 if match found, else D2 = $F for no match.
125E|              ; Also saves boot id in memory, and sets alternate boot indicator.
125E|              ; Destroys A3 and D2.
125E|              ;-----
125E|
125E| 125E| 47FA 001A    XLATE   LEA     KEYTBL,A3      ;get ptr to keycode table
1262| 4282          CLR.L   D2              ;clear for counter

```



```

1264| B01B          @1      CMP.B    (A3)+,D0      ;do search until match
1266| 6708          BEQ.S    @2              ;skip if match
1268| 5242          ADDQ     #1,D2              ;else bump cntr
126A| 4A13          TST.B    (A3)              ;at end?
126C| 66F6          BNE.S    @1              ;if not continue scan
126E| 747F          MOVEQ    #$7F,D2          ;else set for invalid code
1270|
1270| 11C2 01B3      @2      MOVE.B    D2,BOOTDVCE    ;save as boot device code
1274| 08C7 001C      BSET     #ALTBOOT,D7        ;set indicator
1278| 4E75          RTS              ;and exit
127A|
127A| F4 F1 F2      KEYTBL .BYTE    KEY1,KEY2,KEY3      ;1,2,3
127D| F3 E4 01      .BYTE    KEY4,KEY5,01              ;4,5,reserved (01 is invalid keycode)
1280| E1 E2 01      .BYTE    KEY6,KEY7,01              ;6,7,reserved
1283| E3 D0 01      .BYTE    KEY8,KEY9,01              ;8,9,reserved
1286| 01 01 01      .BYTE    01,01,01              ;reserved
1289| 01            .BYTE    01              ;reserved for power-cycle mode
128A| AF            .BYTE    ENTRKEY              ;Enter on numeric key pad
128B|
128B| 00            TBLEND .BYTE    0              ; (for Monitor access)
128C|
128C|              .PAGE
128C|
128C| ;-----
128C| ; Try initial clock read and save data for later use
128C| ;-----
128C|
128C|
128C| CLKTST
128C|      .IF      NEWLISA = 1
128C|      .IF      DIAGS = 1
128C| 6112      BSR.S    READCLK      ;go read clock
128E| 4A87      TST.L    D7              ;restart if in loop mode
1290| 6BFA      BMI.S    CLKTST
1292| 0807 000E      BTST    #CLK,D7      ;any errors?
1296| 6600 0102      BNE     TSTCHK      ;abort if yes
129A| 6100 232A      BSR     CHKIOBRD    ;else mark I/O board OK
129E| 604E          BRA.S    CONFIG      ;and exit to next test
12A0|
12A0|      ; Subroutine to read clock - destroys regs A0-A2, D0-D1
12A0|
12A0|
12A0| READCLK DISABLE      ;disable all interrupts
12A0| #      MOVE      SR,-(SP)
12A2| #      ORI      #$0700,SR
12A6|      MOVEQ     #$02,D0      ;set up read clock cmd
12A8| 6100 F6AC      BSR     COPSCMD      ;and send to COPS
12AC| 6534          BCS.S    CLKERR      ;exit if error
12AE| 347C 01C0      MOVEA    #DATARG$A2    ;set ptr to end of save area      RM000
12B2| 327C 01B9      RDCLK0 MOVEA    #CLKDATA-1,A1    ;set ptr to start of save area      RM000

```



```

12B6| 6100 F7C6          BSR.S  GETDATA      ;go get clock reset code
12BA| 6526              BCS.S  CLKERR      ;exit if timeout error
12BC| 0C00 0080          CMP.B  #$80,D0     ;is it the reset code?
12C0| 66F0              BNE.S  RDCLK0      ;skip if no to continue wait
12C2| 6100 F7BA          BSR.S  GETDATA      ;go check if clock data
12C6| 651A              BCS.S  CLKERR
12C8| 0200 00F0          ANDI.B  #$F0,D0     ;mask to check if clock flag
12CC| 0C00 00E0          CMP.B  #$E0,D0     ;clock data?
12D0| 66E0              BNE.S  RDCLK0      ;continue wait if no
12D2|
12D2| 7205              MOVEQ   #5,D1        ;set expected byte count
12D4| 6100 F7A8          RDCLK1  BSR  GETDATA  ;go read clock data
12D8| 6508              BCS.S  CLKERR      ;exit if error
12DA| 5341              SUBQ    #1,D1        ;else loop until all data received
12DC| 66F6              BNE.S  RDCLK1
12DE|                  ENABLE          ;restore interrupt mask
12DE| 46DF              #          MOVE      (SP)+,SR
12E0| 4E75              RTS
12E2|
12E2|                  ; Error exit - set indicator and return
12E2|
12E2| 08C7 000E          CLKERR  BSET   #CLK,D7
12E6|                  ENABLE          ;restore interrupt mask
12E6| 46DF              #          MOVE      (SP)+,SR
12E8| 003C 0001          ORI.B  #$01,CCR     ;leave carry bit set
12EC| 4E75              RTS
12EE|
12EE|                  .ENDC
12EE|                  .ENDC
12EE|                  .PAGE
12EE|
12EE|                  ;-----
12EE|                  ; Scan I/O slots to determine what cards, if any, are installed and save
12EE|                  ; id's of installed cards.
12EE|                  ;-----
12EE|
12EE|                  CONFIG
12EE|                  .IF  USERINT = 1
12EE| 327C 1E20          MOVEA   #XCRDSTRT,A1    ;hilite I/O slot test icon
12F2| 6100 2280          BSR    INVICON
12F6|                  .ENDC
12F6|
12F6| 7801              CONFIG2 MOVEQ   #1,D4        ;set flag for status check
12F8| 610C              BSR.S  RDSLOTS          ; and go scan the slots
12FA|
12FA|                  .IF  DIAGS = 1
12FA| 4A87              TST.L   D7              ;restart if in loop mode
12FC| 6BF8              BMI.S  CONFIG2

```



```

12FE|                                     .ENDC
12FE|
12FE| 6100 22CE          BSR      CHKXCRD      ;mark I/O slots OK
1302| 6000 0096          BRA      TSTCHK      ;exit to check overall results
1306|
1306| ;-----
1306| ; Subroutine to scan I/O expansion slots
1306| ; Inputs:
1306| ;     D4 = non-zero if status check to be done, else 0 for no check
1306| ; Outputs:
1306| ;     Saves card id's in locations $298-$29C
1306| ;     Error bits set in D7 if slot card errors encountered
1306| ;     Error code saved in location BOOTDATA+1
1306| ; Side Effects:
1306| ;     A5,A6 trashed
1306| ;-----
1306|
1306| 48E7 4070          RDSLOTS  MOVEM.L D1/A1-A3,-(SP) ;save regs
130A| 2C4F              MOVE.L  SP,A6      ;save stack ptr
130C| 4281              CLR.L   D1        ;for result use
130E| 327C 0298          MOVEA  #IO1ID,A1   ;get ptr to id save area          RM000
1312|
1312| 247C 00FC 0001      MOVE.L  #SLOT1L,A2   ;get slot 1 address
1318|
1318| 2A78 0008          MOVE.L  BUSVCTR,A5    ;save current bus vector value
131C| 47FA 0014          LEA     NOCRD1,A3    ;init bus error vector
1320| 21CB 0008          MOVE.L  A3,BUSVCTR   ; in case no card installed
1324| 030A 0000          MOVEP   (A2),D1      ;read id for slot 1
1328| 6156              BSR.S   CHKID        ;go check id
132A| 6408              BCC.S   SLOT2        ;skip if OK
132C| 08C7 0019          BSET    #IO1ERR,D7  ;else set error indicator
1330| 6002              BRA.S   SLOT2        ;and continue
1332|
1332| 4259              NOCRD1  CLR      (A1)+ ;set id for no card
1334|
1334| 247C 00FC 4001      SLOT2   MOVE.L  #SLOT2L,A2 ;do same for slot 2
133A| 47FA 0014          LEA     NOCRD2,A3
133E| 21CB 0008          MOVE.L  A3,BUSVCTR
1342| 030A 0000          MOVEP   (A2),D1      ;read and check id
1346| 6138              BSR.S   CHKID
1348| 6408              BCC.S   SLOT3        ;skip if OK
134A| 08C7 001A          BSET    #IO2ERR,D7  ;else set error indicator
134E| 6002              BRA.S   SLOT3        ;and continue
1350|
1350| 4259              NOCRD2  CLR      (A1)+ ;set id for no card
1352|
1352| 247C 00FC 8001      SLOT3   MOVE.L  #SLOT3L,A2 ;and finally for slot 3

```



```

1358| 47FA 0014          LEA      NOCRD3,A3
135C| 21CB 0008          MOVE.L  A3,BUSVCTR
1360| 030A 0000          MOVEP   (A2),D1          ;read and check id
1364| 611A              BSR.S    CHKID
1366| 6408              BCC.S    CFGEXIT          ;skip if OK
1368| 08C7 001B          BSET     #IO3ERR,D7      ;else set error indicator
136C| 6002              BRA.S    CFGEXIT          ;go to exit
136E|
136E| 4259          NOCRD3 CLR      (A1)+          ;set id for no card
1370|
1370|          ; Restore default bus error vector and SP and continue
1370|
1370| 007C 0700          CFGEXIT ORI      #$0700,SR          ;ensure interrupts off
1374| 21CD 0008          MOVE.L  A5,BUSVCTR          ;restore from previous saves
1378| 2E4E              MOVE.L  A6,SP
137A| 4CDF 0E02          MOVEM.L (SP)+,D1/A1-A3      ;and restore regs
137E| 4E75              RTS              ;then exit
1380|
1380|          ;-----
1380|          ; Subroutine to do I/O slot card id check.
1380|          ; Requires D1 = card id
1380|          ;-----
1380|
1380|          CHKID
1380| 0C41 FFFF          CMP      #$FFFF,D1          ;check for prototype card
1384| 6710              BEQ.S    @9              ;skip if not - treat as no card
1386| 32C1              MOVE     D1,(A1)+          ;else save id
1388| 6B06              BMI.S    @7              ;if bootable go do check
138A| 0801 000E          BTST     #STBIT,D1          ; or do if status routine exists
138E| 6704              BEQ.S    @8              ;skip if not
1390| 6100 0E3A          @7      BSR      RDIOSLT          ;else go check for good board
1394| 4E75              @8      RTS
1396|
1396| 4259          @9      CLR      (A1)+          ;set id for no card
1398| 4E75              RTS
139A|
139A|          .PAGE
139A|          ;-----
139A|          ; Check test results by checking error indicators in reg D7.
139A|          ; Output greeting message if system contains memory and all is OK.
139A|          ; Else output appropriate error messages.
139A|          ;-----
139A|
139A| 6100 EC9E          TSTCHK BSR      SAVEREGS          ;save regs first
139E|
139E| 47FA 000A          LEA      TST2,A3          ;setup bus error vector for type check  CHG032
13A2| 21CB 0008          MOVE.L  A3,BUSVCTR          ;                      CHG032

```



Computer Boot ROM 2.48 Listing • 105 of 133

```

13F4|
13F4|           ; Continue check for specific error
13F4|
13F4| 0807 0000          BTST    #MMU,D7          ;CHECK IF MMU ERROR
13F8| 6704              BEQ.S    @2              ;SKIP IF NO
13FA| 7028              MOVEQ   #EMMU,D0         ;ELSE GET ERROR CODE
13FC| 600C              BRA.S    @9              ;and go output it
13FE|
13FE|           @2
13FE|           .IF      NEWLISA = 1
13FE|           .IF      ROM16K = 1
13FE| 0807 0002          BTST    #VID,D7          ;CHECK IF VIDEO ERROR
1402| 6704              BEQ.S    @3              ;SKIP IF NO
1404| 702A              MOVEQ   #EVID,D0         ;ELSE GET ERROR CODE
1406| 6002              BRA.S    @9              ;and go output it
1408|
1408| 702B              @3      MOVEQ   #ECPAR,D0         ;else must be parity ckt error
140A|
140A|           .ENDC                      ;{ROM16K}
140A|           .ENDC                      ;{NEWLISA}
140A|           .ENDC                      ;{DIAGS}
140A|
140A| 6000 01B4          @9      BRA      TSTXIT         ;go to exit
140E|
140E|           .ENDC                      ;{ROM4K}
140E|
140E|           ;-----
140E|           ; Scan for exception errors
140E|           ;-----
140E|
140E| 2007              EXCHK    MOVE.L  D7,D0          ;mask off non-exception errors
1410| 0280 0000 03F0      ANDI.L  #EXMSK,D0
1416| 4A80              TST.L    D0              ;OK?
1418| 6744              BEQ.S    IOCHK            ;skip if yes to next check
141A|
141A|           .IF  USERINT = 0
141A|           .ELSE
141A|
141A|           ; Sound error tones
141A|
141A| 6100 02B0          BSR      LOPTCH            ;general logic failure causes lo,hi tones
141E| 6100 02A8          BSR      HIPTCH
1422| 45FA 2930          LEA      LISA,A2          ;set ptr for general LISA error
1426|           .ENDC
1426|           ; Scan for details on exception errors
1426|
1426| 0807 0004          BTST    #CPUINTR,D7        ;NMI?

```



```

142A| 6704                BEQ.S    @1
142C| 702C                MOVEQ    #ECPUNTR,D0    ;set error code
142E| 602A                BRA.S    @9            ;and go display
1430|
1430| 0807 0005            @1      BTST    #BUSEXCP,D7    ;bus error?
1434| 6704                BEQ.S    @2
1436| 702D                MOVEQ    #EBUSEXCP,D0    ;set error code
1438| 6020                BRA.S    @9
143A|
143A| 0807 0006            @2      BTST    #ADREXCP,D7    ;address error?
143E| 6704                BEQ.S    @3
1440| 702E                MOVEQ    #EADREXCP,D0    ;set error code
1442| 6016                BRA.S    @9
1444|
1444| 0807 0007            @3      BTST    #MISEXCP,D7    ;miscellaneous error?
1448| 6704                BEQ.S    @4
144A| 702F                MOVEQ    #EMISEXCP,D0    ;set error code
144C| 600C                BRA.S    @9
144E|
144E| 0807 0008            @4      BTST    #ILLEXCP,D7    ;illegal instruction error?
1452| 6704                BEQ.S    @5
1454| 7030                MOVEQ    #EILLEXCP,D0    ;set error code
1456| 6002                BRA.S    @9
1458|
1458| 7031                @5      MOVEQ    #ETRPEXCP,D0    ;must be a trap error
145A|
145A| 6000 0164            @9      BRA      TSTXIT        ;and go to exit
145E|
145E|                      ;-----
145E|                      ;  Check for I/O errors
145E|                      ;-----
145E|
145E| 2007                IOCHK    MOVE.L    D7,D0        ;GET ERRORS
1460| 0280 001F DC00        ANDI.L    #IOMSK,D0        ;MASK OFF NON-IO ERRORS
1466| 4A80                TST.L     D0                ;OK?
1468| 6700 008E            BEQ      KBDCHK        ;SKIP IF YES TO NEXT CHECK
146C|
146C|                      .IF  USERINT = 0
146C|                      .ELSE
146C| 45FA 24DD            LEA      IOBRD,A2        ;set ptr for I/O board icon
1470|                      .ENDC
1470|
1470|                      .IF  ROM4K = 0
1470|                      ;  Scan for details on I/O errors
1470|
1470|                      .IF      ROM16K = 1
1470| 0807 000A            BTST     #VIA1,D7        ;check for keyboard VIA errors

```



```

1474| 6708                BEQ.S   @1           ;skip if OK
1476| 7032                MOVEQ   #EVIA1,D0      ;else set error code
1478| 6170                BSR.S   ERRDISP        ;display the error
147A| 6000 F434           BRA     VIA1CHK        ;and loop on VIA #1 test
147E|
147E|                    ; Sound error tones if not VIA #1 error (controls the speaker)
147E| 6100 024C           @1     BSR     LOPTCH          ;I/O errors cause lo,hi,lo tones
1482| 6100 0244           BSR     HIPTCH
1486| 6100 0244           BSR     LOPTCH
148A|                    ; Continue scan for detailed errors
148A|
148A| 0807 000B           BTST     #VIA2,D7        ;parallel port VIA error?
148E| 6704                BEQ.S   @2
1490| 7033                MOVEQ   #EVIA2,D0      ;set error code
1492| 6052                BRA.S   @19
1494|                .ENDC
1494|
1494| 0807 000C           @2     BTST     #IOCOPS,D7
1498| 6708                BEQ.S   @3
149A| 7034                MOVEQ   #EIOCOP,D0      ;get error code
149C| 614C                BSR.S   ERRDISP        ;display error
149E| 6000 F44A           BRA     COPSENBL        ;and go do loop on COPS test
14A2|
14A2|                @3
14A2|                .IF   DIAGS = 1
14A2| 0807 000E           BTST     #CLK,D7
14A6| 6704                BEQ.S   @4
14A8| 7036                MOVEQ   #ECLK,D0        ;ELSE GET ERROR CODE
14AA| 603A                BRA.S   @19
14AC|                .ENDC
14AC|
14AC|                @4
14AC|                .IF   FULLSCC = 1
14AC| 0807 000F           BTST     #RS232A,D7
14B0| 6704                BEQ.S   @6
14B2| 7037                MOVEQ   #ERS232A,D0      ;ELSE GET ERROR CODE
14B4| 6030                BRA.S   @19
14B6|
14B6| 0807 0010           @6     BTST     #RS232B,D7
14BA| 6704                BEQ.S   @7
14BC| 7038                MOVEQ   #ERS232B,D0      ;ELSE GET ERROR CODE
14BE| 6026                BRA.S   @19
14C0|                .ENDC
14C0|
14C0| 0807 0011           @7     BTST     #DISK,D7
14C4| 6704                BEQ.S   @8
14C6| 7039                MOVEQ   #EDISK,D0        ;ELSE GET ERROR CODE

```



```

14C8| 601C          BRA.S   @19
14CA|
14CA| 0807 0012      @8     BTST   #IOEXCP,D7
14CE| 6704          BEQ.S   @9
14D0| 703A          MOVEQ  #EIOEXCP,D0      ;ELSE GET ERROR CODE
14D2| 6012          BRA.S   @19
14D4|
14D4| 0807 0013      @9     BTST   #IOCOPS2,D7      ;COPS code error?
14D8| 6704          BEQ.S   @10
14DA| 703B          MOVEQ  #EIOCOP2,D0      ;get error code
14DC| 6008          BRA.S   @19
14DE|
14DE| 0807 0014      @10    BTST   #IOKBD,D7      ;I/O or keyboard error?
14E2| 6702          BEQ.S   @19
14E4| 703C          MOVEQ  #EIOKBD,D0      ;get error code
14E6|
14E6|              .ENDC              ;{ROM4K}
14E6|
14E6| 6000 00D8      @19     BRA     TSTXIT
14EA|
14EA|              ;-----
14EA|              ; Subroutine to do display for fatal errors
14EA|              ;-----
14EA|
14EA| ERRDISP
14EA|              .IF USERINT = 1
14EA| 6100 1FEE      BSR      DSPERRICON      ;display error
14EE|              .ENDC
14EE|
14EE|              BSR      DSPCODE      ;output error code
14F2| 08C7 001F      BSET    #LOOP,D7      ;set for looping operation
14F6| 4E75          RTS
14F8|
14F8|              ;-----
14F8|              ; Check for keyboard error
14F8|              ;-----
14F8|
14F8| 0807 000D      KBDCHK  BTST   #KBDOPS,D7      ;Keyboard error?
14FC| 6716          BEQ.S   MEMCHK      ;skip to next check if not
14FE|
14FE|              .IF USERINT = 1
14FE|
14FE|              ; Sound error tones
14FE|
14FE| 6100 01C8      BSR      HIPTCH      ;Keyboard error causes hi,lo,hi tones
1502| 6100 01C8      BSR      LOPTCH
1506| 6100 01C0      BSR      HIPTCH

```



```

150A| 45FA 2699          LEA      KEYBDOUT,A2      ;set ptr for keyboard icon
150E|                  .ENDC
150E|
150E| 7035                MOVEQ    #EKBD COP,D0      ;set error code
1510| 6000 00AE          BRA      TSTXIT           ;and go to exit
1514|
1514|
1514|          ;-----
1514|          ; Check for memory errors
1514|          ;-----
1514|
1514|          MEMCHK
1514| 2007                MOVE.L    D7,D0              ;GET ERRORS
1516| 0280 0060 0000      ANDI.L    #MEMMSK,D0        ;MASK OFF NON-memory ERRORS
151C| 4A80                TST.L     D0                ;any errors?
151E| 6700 0070          BEQ       IOSCHK             ;skip if no - must be I/O slot error
1522|
1522|          .IF USERINT = 0
1522|          .ELSE
1522|
1522|          ; Sound memory error tones
1522|
1522| 6100 01A8          BSR       LOPTCH             ;memory error causes lo,hi,hi tones
1526| 6100 01A0          BSR       HIPTCH
152A| 6100 019C          BSR       HIPTCH
152E|
152E|          ; determine which memory card in error if more than one
152E|
152E| 0CB8 0008 0000 02A8  CMPI.L    #HEX512K,TOTLMEM ;more than 1 memory card?
1536| 6E14                BGT.S     SCNRSLTS         ;skip if yes
1538|
1538|          ; only one card - check memory addresses to determine slot
1538|
1538| 2038 02A4          MOVE.L    MINMEM,D0          ;get low physical address
153C| 0C80 0010 0000      CHKMADR  CMPI.L    #ONEMEG,D0 ;address in slot 1?
1542| 6D04                BLT.S     @2                ;skip if not
1544| 7201                MOVEQ    #1,D1             ;set board id for slot 1
1546| 6002                BRA.S     @3
1548| 7202                @2      MOVEQ    #2,D1             ;set board id for slot 2
154A| 6026                @3      BRA.S     MERRCHK        ;and go scan for details
154C|
154C|          ; more than one memory card - scan memory test results to determine which card
154C|
154C|          SCNRSLTS
154C| 0807 0016          BTST      #MPAR,D7           ;parity error?
1550| 6706                BEQ.S     @1                ;skip if not
1552| 2038 01A6          MOVE.L    PEADDR,D0          ;go get error address

```

CHG015



```

1556| 60E4          BRA.S  CHKMADR      ;and check it
1558|
1558|          ; Check for R/W error
1558|
1558| 307C 0186      @1    MOVEA  #MEMRSLT,A0      ;set ptr to OR masks      RM000
155C| 7008          MOVEQ  #8,D0      ;and set counter
155E| 4A58          @4    TST    (A0)+      ;check the rows
1560| 6604          BNE.S  @5      ;skip if error detected
1562| 5340          SUBQ   #1,D0      ;else check all masks
1564| 66F8          BNE.S  @4      ;until done
1566|
1566| 0C00 0004      @5    CMP.B  #4,D0      ;check where error found
156A| 6E04          BGT.S  @6      ;skip if low memory error
156C| 7201          MOVEQ  #1,D1      ;high memory on card 1
156E| 6002          BRA.S  MERRCHK
1570| 7202          @6    MOVEQ  #2,D1      ;low memory on card 2
1572|
1572|          .ENDC          ;{USERINT}
1572|
1572|          .IF  ROM4K = 0
1572|
1572|          ; scan for error details
1572|
1572| 0807 0015      MERRCHK BTST  #MEM,D7      ;check for main memory R/W error
1576| 6708          BEQ.S  @2      ;exit if not
1578| 7046          MOVEQ  #EMEM,D0      ;else display error code
157A| 11C1 02AD      MOVE.B  D1,MEMSLOT      ;save slot # for board in error
157E| 6002          BRA.S  MEMERR
1580|
1580| 7047          @2    MOVEQ  #EPAR,D0      ;must be parity error
1582|
1582|          .ENDC          ;{ROM4K}
1582|
1582|          MEMERR
1582|          .IF  USERINT = 1
1582|
1582| 21C7 0180      MOVE.L  D7,STATUS      ;save power-up status
1586| 45FA 2446      LEA     MEMBRD,A2      ;set ptr for memory board icon
158A| 6100 1EC4      BSR     DSPNUMICON      ;display icon and board slot #
158E| 6034          BRA.S  TSTXIT2      ;finally exit to monitor
1590|
1590|          .ELSE
1590|          .ENDC          ;{USERINT}
1590|
1590|          ;-----
1590|          ;-----
1590|          IOSCHK

```



```

1590|          .IF ROM4K = 0
1590|          .IF USERINT = 0
1590|          .ELSE
1590|          ; Sound error tones
1590|
1590| 6100 0136          BSR      HIPTCH          ;I/O slot error causes hi,lo,lo tones
1594| 6100 0136          BSR      LOPTCH
1598| 6100 0132          BSR      LOPTCH
159C| 45FA 2476          LEA      Xcard,A2          ;set ptr for I/O slot board icon
15A0|          .ENDC
15A0|
15A0| 0807 001B          @1      BTST      #IO3ERR,D7          ;check for slot 3 error
15A4| 6704              BEQ.S      @2          ;exit if not
15A6| 7203              MOVEQ     #3,D1          ;else set slot #
15A8| 600C              BRA.S      @4
15AA|
15AA| 0807 001A          @2      BTST      #IO2ERR,D7          ;slot 2 error?
15AE| 6704              BEQ.S      @3
15B0| 7202              MOVEQ     #2,D1          ;set slot #
15B2| 6002              BRA.S      @4
15B4|
15B4| 7201              @3      MOVEQ     #1,D1          ;must be slot 1 error
15B6|
15B6| 1038 01B4          @4      MOVE.B    BOOTDATA,D0          ;get error code
15BA| 6100 1E94          BSR      DSPNUMICON          ;display error icon and slot #
15BE| 6004              BRA.S      TSTXIT2          ;and exit to monitor
15C0|
15C0|          .ENDC          ;{ROM4K}
15C0|
15C0|          TSTXIT
15C0|          .IF USERINT = 0
15C0|          .ELSE
15C0| 6100 1F18          BSR      DSPERRICON          ;display error icon
15C4|
15C4|          TSTXIT2
15C4| 21C7 0180          MOVE.L    D7,STATUS          ;save status
15C8| 6100 0058          BSR      DSPCODE          ;display the error code
15CC|
15CC|          ; Save error data in special parameter memory area, then exit to monitor
15CC|
15CC|          ;*****
15CC|          ; Delete for LISA 2
15CC|          ;*****
15CC|
15CC|          ;          LEA      PMVCT,A3          ;setup bus error vector for PM
15CC|          ;          MOVE.L   A3,BUSVCTR          ;

```

CHG034

RM013

RM013




```

15CC|          ;      BSR.S   CHKSTATPM      ;check if error already saved
15CC|          ;      BCC.S   GOTOMON        ;skip if yes
15CC|          ;      MOVEA.L #STATSTRT,A0    ;set starting ptr
15CC|          ;      MOVE.B   D0,(A0)         ;save error code
15CC|          ;      MOVE     ADRLTCH,D0      ;save error address latch contents
15CC|          ;      MOVEP    D0,2(A0)        ;
15CC|          ;      MOVE.B   MEMSLOT,6(A0)    ;save memory slot #
15CC|          ;      MOVE.L   CLKDATA,D0      ;save clock data
15CC|          ;      MOVEP.L  D0,8(A0)        ;
15CC|          ;      MOVE     CLKDATA+4,D0    ;
15CC|          ;      MOVEP    D0,16(A0)       ;
15CC|          ;      CLR.L    D0              ;clear remaining area
15CC|          ;      MOVEP.L  D0,20(A0)       ;
15CC|          ;
15CC|          ;      MOVEQ    #STATWRDS-2,D0   ;validate save area
15CC|          ;      BSR      WRTSUM
15CC|
15CC| 6100 FC14      GOTOMON BSR      SCANCPS      ;clear COPS queue
15D0| 6100 F34E      BSR      CPSINIT        ;reinit interface
15D4| 6100 19F6      BSR      CURSORINIT     ;init cursor and mouse
15D8| 6000 0FC2      BRA      MONITOR        ;then jump to monitor
15DC|
15DC|          ;-----
15DC|          ;   Parameter memory bus error handler           RM013
15DC|          ;-----
15DC|
15DC| 3E7C 0480      PMVCT  MOVEA   #STKBASE,SP    ;reset stack                RM013
15E0| 6100 F100      BSR      SETBUSVCT        ;restore bus error vector      RM013
15E4| 60E6          BRA.S   GOTOMON            ;and exit to monitor          RM013
15E6|
15E6|          ;-----
15E6|          ;   Subroutine to check special parameter memory validity.
15E6|          ;   Verify checksum routine sets carry bit if checksum not valid.
15E6|          ;-----
15E6|
15E6|          ;CHKSTATPM                                     CHG034
15E6|          ;      MOVEM.L D0-D1/A0,-(SP)    ;save regs
15E6|          ;      MOVEA.L #STATSTRT,A0      ;set starting ptr
15E6|          ;      MOVEQ   #STATWRDS-1,D0    ;and # of words to check
15E6|          ;      MOVE    D0,D1             ;set for shared memory
15E6|          ;      BSR     VFYCHKSM          ;and go do checksum
15E6|          ;@1      MOVEM.L (SP)+,D0-D1/A0  ;restore regs
15E6|          ;      RTS
15E6|
15E6|          .ENDC
15E6|          ;-----
15E6|          ;   Scan for non-fatal errors

```



```

15E6| ;-----
15E6|
15E6| OTHER
15E6|     .IF ROM16K = 1
15E6| 2007 MOVE.L D7,D0 ;get errors
15E8| 0280 0180 0000 ANDI.L #OTHRMSK,D0 ;isolate to non-fatal errors
15EE| 4A80 TST.L D0 ;OK?
15F0| 672C BEQ.S @9 ;skip if no errors
15F2| 0807 0017 BTST #KBDOUT,D7 ;Keyboard disconnected?
15F6| 6706 BEQ.S @1 ;skip if no
15F8|
15F8|     .IF USERINT = 0
15F8|     .ELSE
15FC| 6014 BRA.S @2 ;with question markcon
15FE|     .ENDC
15FE|
15FE| @1 ;must be mouse
15FE|     .IF USERINT = 0
15FE|     .ELSE
15FE| 6100 0236 BSR CHKPM ;check parameter memory before notify
1602| ; of mouse disconnect
1602| 6516 BCS.S @8 ;ignore error if invalid
1604| 0839 0007 00FC C18D BTST #MOUSEON,MEMCODE ;check if should be installed
160C| 670C BEQ.S @8 ;skip if not
160E| 45FA 261B LEA MOUSEOUT,A2 ;else display mouse icon
1612| 6100 1F42 @2 BSR DSPQICON ;with question mark
1616|     .ENDC
1616|
1616| 6000 00A0 BRA NOTIFY ;alert user
161A|
161A|     .ENDC ;{ROM16K}
161A|
161A| 0887 0018 @8 BCLR #MOUSOUT,D7 ;ignore mouse disconnected error
161E|
161E| 6000 00BE @9 BRA SYSOK ;system must be OK
1622|     .PAGE
1622| ;-----
1622| ; Subroutine to output error code
1622| ;-----
1622|
1622| 48E7 F000 DSPCODE MOVEM.L D0-D3,-(SP) ;save regs
1626|
1626|     .IF USERINT = 0
1626|     .ENDC
1626|     .IF NEWTWIG = 0
1626|     .ENDC
1626|

```



```

1626|          .IF USERINT = 1
1626| 3A3C 0097      MOVE    #CODEROW,D5      ;set screen ptrs for display
162A| 3C3C 0012      MOVE    #CODECOL,D6
162E|              .ENDC
162E|
162E| 6004          BRA.S    GETDIG          ;go do display
1630|              .IF NEWTWIG = 1
1630|
1630|              ; Translate up to 4 digit hex error code to decimal
1630|              ; Second entry point for routine
1630|
1630|              DSPDEC
1630| 48E7 F000      MOVEM.L D0-D3,-(SP)      ;save regs
1634| 0280 0000 FFFF  GETDIG  ANDI.L  #$0FFFF,D0      ;clear other digits
163A| 7201          MOVEQ   #1,D1          ;display 1 char at a time
163C| 4A40          TST     D0            ;is it 0?
163E| 6726          BEQ.S   @9            ;exit if yes to display it
1640| 4282          CLR.L   D2            ;clear working regs
1642| 4283          CLR.L   D3
1644|
1644|              ; display all non-zero digits
1644|
1644| 80FC 000A      @1      DIVU    #$A,D0      ;converting to decimal
1648| 4840          SWAP    D0            ;get remainder
164A| 1400          MOVE.B  D0,D2          ;save for display
164C| E89A          ROR.L   #4,D2
164E| 5243          ADDQ    #1,D3          ;set count
1650|
1650| 4240          CLR     D0            ;clear remainder
1652| 4840          SWAP    D0            ;get new quotient
1654| 4A40          TST     D0            ;quit when =0
1656| 6702          BEQ.S   @2            ;skip to do display
1658| 60EA          BRA.S   @1
165A|
165A| E99A          @2      ROL.L   #4,D2      ;get char for output
165C| 1002          MOVE.B  D2,D0
165E| 5343          SUBQ    #1,D3          ;decr digit count
1660| 6704          BEQ.S   @9            ;skip to display last digit
1662| 6114          BSR.S   OUTCH        ;display a digit
1664| 60F4          BRA.S   @2            ;and loop until done
1666|
1666|              .ENDC
1666|
1666| 6106          @9      BSR.S   OUTCHR      ;do output and CR
1668|
1668| 4CDF 000F      DSPCKIT MOVEM.L (SP)+,D0-D3      ;restore regs
166C| 4E75          RTS              ;and return

```



```

166E|
166E| ;-----
166E| ; Subroutine to invoke code display routine, then do CR
166E| ;-----
166E|
166E| 6108      OUTCHR BSR.S   OUTCH          ;output digits
1670|
1670|          .IF USERINT = 0
1670|          .ELSE
1670| 0645 000A      ADD      #CHRSPC,D5      ;bump to next char row
1674|          .ENDC
1674|
1674| 7C01          MOVEQ    #1,D6            ;and do CR
1676| 4E75          RTS
1678|
1678|          .PAGE
1678| ;-----
1678| ; Subroutines to enable display of hex codes
1678| ; Requires D0 = value to display
1678| ;          D1 = # of digits to display
1678| ;-----
1678|
1678| 48E7 E000      OUTCH    MOVEM.L D0-D2,-(SP) ;save regs
167C| 7408          MOVEQ    #8,D2            ;set max digits to display
167E| B401          @1      CMP.B   D1,D2      ;check digits desired
1680| 6706          BEQ.S     @2              ;and skip if match
1682| E998          ROL.L     #4,D0            ;else skip over digit
1684| 5342          SUBQ      #1,D2            ;update count
1686| 60F6          BRA.S     @1              ;and loop until match
1688|
1688| E998          @2      ROL.L     #4,D0            ;rotate to next digit
168A| 610A          BSR.S     OUTNIB          ;go output one digit
168C| 5341          SUBQ      #1,D1            ;decr count
168E| 66F8          BNE.S     @2              ;loop until done
1690|
1690| 4CDF 0007      MOVEM.L (SP)+,D0-D2      ;restore and exit
1694| 4E75          RTS
1696|
1696| ; The following routine does conversion to ASCII to enable display
1696|
1696| 2F00          OUTNIB  MOVE.L   D0,-(SP)      ;SAVE REG
1698| 0240 000F      ANDI      #$000F,D0        ;ISOLATE DIGIT TO DISPLAY
169C| 0C00 0009      CMPI.B    #9,D0            ;CHECK IF NUMERIC
16A0| 6206          BHI.S     ALPHA            ;SKIP IF NOT
16A2| 0000 0030      ORI.B     #$30,D0          ;CONVERT TO ASCII
16A6| 6008          BRA.S     DSPCH            ;AND GO DISPLAY
16A8|

```



```

16A8| 0400 0009      ALPHA  SUBI.B  #9,D0          ;CONVERT FOR
16AC| 0000 0040      ORI.B  #$40,D0          ; ASCII
16B0|
16B0| 6100 2088      DSPCH  BSR      DSPVAL        ;OUTPUT IT
16B4| 201F          MOVE.L  (SP)+,D0          ;RESTORE REG
16B6| 4E75          RTS
16B8|
16B8|          .PAGE
16B8|
16B8|          ;-----
16B8|          ; Routine to notify user of non-fatal error.  Beep speaker and pause
16B8|          ; for 5 seconds.
16B8|          ;-----
16B8|
16B8|          NOTIFY
16B8|          .IF ROM16K = 1
16B8| 610E          BSR.S  HIPTCH          ;beep at high pitch twice
16BA| 610C          BSR.S  HIPTCH
16BC| 610E          BSR.S  LOPTCH        ;beep at low pitch
16BE| 6100 F414      BSR      DELAY5        ;delay 5 seconds
16C2| 6100 1A16      BSR      CLRDESK        ;clear desktop          CHG033
16C6| 601A          BRA.S  DOBOOT        ;then go attempt boot
16C8|
16C8|          ;-----
16C8|          ; Subroutine to beep speaker at high pitch
16C8|          ;-----
16C8| 7020      HIPTCH  MOVEQ  #$20,D0          ; set frequency
16CA| 6002          BRA.S  SETDUR          ; and go do it
16CC|
16CC|          ;-----
16CC|          ; Subroutine to beep speaker at low pitch
16CC|          ;-----
16CC|
16CC| 7060      LOPTCH  MOVEQ  #$60,D0          ; set frequency
16CE| 323C 00FA      SETDUR  MOVE  #250,D1        ; 1/8 sec duration
16D2| 7404          MOVEQ  #4,D2          ; low volume
16D4| 6100 F420      BSR      TONE          ; and go do it
16D8| 6100 F3F2      BSR      DELAY_1        ; delay for .1 sec
16DC| 4E75          RTS
16DE|
16DE|          .ENDC
16DE|          .PAGE
16DE|
16DE|          ;-----
16DE|          ; No errors detected - output greeting message
16DE|          ;-----
16DE|
16DE| 42B8 0180      SYSOK  CLR.L  STATUS        ;set status
16E2|          .IF ROM16K = 1

```



```

16E2|          ;      BSR      CHKSTATPM      ;check special save area
16E2|          ;      BCC.S    DOBOOT        ;skip if valid data saved
16E2|          ;      CLR.B    STATSAV        ;else set status to 0
16E2|          .ENDC
16E2|
16E2|          .IF ROM4K = 0
16E2|          .IF USERINT = 0
16E2|          .ENDC          ;{USERINT}
16E2|          .ENDC          ;{ROM4K}
16E2|          .IF ROM4K = 0
16E2|          .IF USERINT = 0
16E2|          .ENDC          ;{USERINT}
16E2|          .ENDC          ;{ROM4K}
16E2|
16E2|          .INCLUDE RM248.B.TEXT
16E2|
16E2|          .PAGE
16E2|          .LIST
16E2|          ;-----
16E2|          ; All is OK - check for boot device and then do bootstrap
16E2|          ;-----
16E2|
16E2| 6100 18E8      DOBOOT BSR      CURSORINIT      ;init cursor/mouse
16E6|
16E6|      BOOTCHK
16E6| 4280          CLR.L    D0          ;clear for use
16E8| 0807 001C      BTST     #ALTBOOT,D7      ;check if alternate boot command rcvd
16EC| 6710          BEQ.S    @1          ;skip if no
16EE| 1038 01B3      MOVE.B  BOOTDVCE,D0      ;get alternate boot code
16F2|
16F2|          .IF      BURNIN = 1
16F2| 0C00 000F      CMP.B    #PC,D0          ;power-cycle mode?
16F6| 665C          BNE.S    DVCECHK        ;if no, go determine device
16F8| 6100 0154      BSR      SAV2PM        ;if yes, save in parameter memory
16FC|          .ENDC
16FC|
16FC| 6056          BRA.S    DVCECHK        ;and go determine device
16FE|
16FE|          @1
16FE|          .IF USERINT = 1
16FE| 0807 001D      BTST     #BTMENU,D7      ;boot menu wanted?
1702| 6600 01F4      BNE      BOOTMENU      ;skip if yes
1706|          .ENDC
1706|
1706| 6100 012E      BSR      CHKPM          ;next step is to check parameter memory
170A|
170A|          .IF AAPL = 1

```



```

170A|          .ELSE
170A| 6416          BCC.S    @2          ;skip if valid
170C|
170C|          ; set default boot
170C|
170C| 4A38 02AF          TST.B    SYSTYPE          ;else check if Lisa 1          CHG030
1710| 670C          BEQ.S    @5          ;skip if yes          CHG030
1712| 7401          MOVEQ    #1,D2          ;else set wait flag          CHG030
1714| 6100 0320          BSR     CHKPROFILE          ;and go check if hard disk attached          CHG030
1718| 6704          BEQ.S    @5          ;skip if yes to do boot from hard disk          CHG030
171A| 7001          MOVEQ    #TWIG2,D0          ;else set for boot from floppy          CHG030
171C| 600E          BRA.S    @3          ;          CHG030
171E|
171E| 7002          @5      MOVEQ    #PROFILE,D0          ;else if not valid do default boot from Profile
1720| 600A          BRA.S    @3
1722|          .ENDC
1722|
1722| 1039 00FC C189          @2      MOVE.B  DVCCODE,D0          ;else read device code
1728| E808          LSR.B    #4,D0          ;rotate to low nibble
172A| 6028          BRA.S    DVCECHK
172C|
172C|          .IF  NEWTWIG = 0
172C|          .ENDC
172C|
172C|          @3
172C|          .IF  ROM16K = 1
172C|          ; Do special check for Applenet and I/O test cards
172C|
172C| 323C 1000          MOVE     #TSTCRD,D1          ;search first for a bootable test card
1730| 383C 1800          MOVE     #TSTQUAL,D4
1734| 6100 01A0          BSR     SEARCH          ;go do search
1738| 661A          BNE.S    DVCECHK          ;skip if not found
173A| 3602          MOVE     D2,D3          ;else save its id
173C| 323C 8001          MOVE     #APPLENET,D1          ;next search for an Applenet card
1740| 383C 9FFF          MOVE     #APPLQUAL,D4
1744| 6100 0190          BSR     SEARCH
1748| 6704          BEQ.S    @4          ;skip if found to boot from it (DMT need)
174A| 3403          MOVE     D3,D2          ;else do boot from I/O test card (DIAG need)
174C| 6A06          BPL.S    DVCECHK          ; unless card boot bit not set          CHG012
174E|          @4
174E| C4FC 0003          MULU     #3,D2          ;convert to boot id
1752| 3002          MOVE     D2,D0          ;and set boot id for appropriate slot
1754|          .ENDC
1754|
1754|          ; Alternate boot desired - check which
1754|
1758|          DVCECHK MOVE.B  D0,BOOTDVCE          ;save for later reference

```



```

1758|          .IF TWIGGY = 1
1758| 4A00      TST.B   D0          ;boot from upper drive?          CHG009
175A| 660E      BNE.S   @1          ;no - go to next check
175C| 4A38 02AF  TST.B   SYSTYPE    ;check system type          CHG009
1760| 6704      BEQ.S   @11         ;skip if Lisa 1.0          CHG009
1762| 6000 076A      @10    BRA     PROBOOT    ;else do Pepsii boot      CHG009
1766|
1766| 4200      @11    MOVE.B  #DRV1,D0    ;else set drive #          CHG009
1768| 600A      BRA.S   @2          ;and go do boot
176A|
176A| 0C00 0001      @1     CMP.B   #TWIG2,D0    ;boot from lower drive?    CHG009
176E| 6608      BNE.S   @3          ;skip if no
1770| 103C 0080      MOVE.B  #DRV2,D0    ;else set drive #
1774| 6000 0456      @2     BRA     TWGBOOT    ;and go boot
1778|          .ENDC
1778|
1778|          @3
1778|          .IF PROFILE = 1
1778| 0C00 0002      CMP.B   #PROFILE,D0    ;boot from Profile?
177C| 67E4      BEQ.S   @10         ;yes - go do it          CHG009
177E|          .ENDC
177E|
177E| 0C00 0004      CMP.B   #IO1PORT2,D0    ;boot from slot 1, ports 1-2?
1782| 6E08      BGT.S   @4          ;skip if not
1784| 227C 00FC 0001 MOVE.L   #SLOT1L,A1    ;set slot address
178A| 601A      BRA.S   @9          ;and go do boot
178C|
178C| 0C00 0007      @4     CMP.B   #IO2PORT2,D0    ;boot from slot 2, ports 1-2?
1790| 6E08      BGT.S   @5          ;skip if not
1792| 227C 00FC 4001 MOVE.L   #SLOT2L,A1    ;set slot address
1798| 600C      BRA.S   @9          ;and go do boot
179A|
179A| 0C00 000A      @5     CMP.B   #IO3PORT2,D0    ;boot from slot 3, ports 1-2?
179E| 6E0A      BGT.S   @6          ;skip if not
17A0| 227C 00FC 8001 MOVE.L   #SLOT3L,A1    ;set slot address
17A6|
17A6| 6000 09B6      @9     BRA     IOSBOOT    ;and go do boot
17AA|
17AA|          @6
17AA|          .IF      BURNIN = 1
17AA| 0C00 000F      CMP.B   #PC,D0          ;power-cycle?
17AE| 6700 0A8A      BEQ     CHKPASS    ;go do cycling
17B2|          .ENDC
17B2|
17B2|          .IF ROM4K = 0
17B2| 0C00 0010      @7     CMP.B   #MON,D0    ;abort boot?
17B6| 6676      BNE.S   LSTCHK    ;skip if no match

```




```

17B8|
17B8|          .IF USERINT = 1
17B8| 08B8 0000 02A2      BCLR    #NORSTRT,STATFLGS ;allow restart option but
17BE| 08F8 0001 02A2      BSET    #NOCONT,STATFLGS ; no CONTINUE button for direct to monitor jump
17C4| 6100 194E          BSR      CLRMENU          ;clear menu bar and
17C8| 6100 199A          BSR      MAKEPCALRT       ; open alert box for monitor or power cycling
17CC|          .ENDC
17CC|
17CC|          .IF BURNIN = 1
17CC|
17CC|          ; Check if completing power-cycling operation
17CC|
17CC|          .IF DEBUG = 0
17CC| 47FA 0052          LEA      PMERR,A3          ;set vector for parameter mem error
17D0| 21CB 0008          MOVE.L  A3,BUSVCTR
17D4|          .ENDC
17D4|
17D4| 6100 0060          BSR      CHKPM            ;check validity of parameter memory
17D8| 653E          BCS.S    PMEXIT              ;skip if not
17DA| 1039 00FC C189    MOVE.B  DVCCODE,D0      ;get boot code
17E0| E808          LSR.B    #4,D0                ;shift to lower nibble
17E2|
17E2|          .IF NEWTWIG = 0
17E2|          .ENDC
17E2|
17E2| 0C00 000F          CMPI.B  #PC,D0            ;exiting from power-cycling?
17E6| 6630          BNE.S    PMEXIT              ;skip if no
17E8| 4239 00FC C189    CLR.B    DVCCODE          ;else reset boot and
17EE| 08B9 0006 00FC C18D BCLR    #6,MEMCODE      ; memory test indicators
17F6|
17F6| 47FA 25FC          LEA      LOOPMSG,A3        ;display loop count
17FA| 6100 1F04          BSR      DSPMSG
17FE| 207C 00FC C195    MOVE.L  #LCNTHI,A0        ;set ptr to loop count
1804| 0108 0000          MOVEP    (A0),D0          ;get it
1808| 7204          MOVEQ    #4,D1                ;set # of digits to display
180A| 6100 FE62          BSR      OUTCHR           ;and do display
180E| 7C0C          MOVEQ    #PCCOL,D6            ;reset col for proper left margin
1810|
1810| 6100 0CB8          BSR      DSPCLK            ;display final clock value
1814| 6100 0CFA          BSR      TWGDSP           ;and display Twiggy error count
1818|
1818|          ; Normal exit
1818|          PMEXIT
1818|
1818|          .IF DEBUG = 0
1818| 6100 EEC8          BSR      SETBUSVCT         ;restore normal bus error vector      RM000
181C|          .ENDC          ;{debug}
181C|

```



```

181C|                                     .ENDC                                     ;{BURNIN}
181C|
181C| 6000 0D7E                         BRA      MONITOR                     ;and go to monitor
1820|
1820|                                     .IF BURNIN = 1
1820| ; Bus error handler for parameter memory error
1820|
1820| 47FA 25E1                         PMERR  LEA      PMMSG,A3             ;setup error message
1824| 45FA 2125                         LEA      IOBRD,A2             ;and icon
1828| 4280                             CLR.L    D0                     ;no error code
182A| 6000 0D08                         BRA      INITMON                    ;exit to monitor
182E|
182E|                                     .ENDC                                     ;{BURNIN}
182E|                                     .ENDC                                     ;{ROM4K}
182E|
182E| LSTCHK
182E|                                     .IF AAPL = 1
182E|                                     .ENDC
182E|
182E|                                     .IF USERINT = 0
182E|                                     .ELSE
182E| 6100 0EDC                         BSR      SQUAWK                    ;else sound error tone
1832| 6000 00C4                         BRA      BOOTMENU                  ;and go to boot menu
1836|                                     .ENDC
1836|
1836|                                     .PAGE
1836| ;-----
1836| ; Subroutine to check parameter memory validity. Calls generalized
1836| ; verify checksum routine.
1836| ;-----
1836|
1836| 48E7 C080                         CHKPM  MOVEM.L D0-D1/A0,-(SP)      ;save regs
183A| 207C 00FC C181                     MOVEA.L #PMSTRT,A0              ;set starting ptr
1840| 303C 001F                         MOVE    #PMWRDS-1,D0           ;and # of words to check
1844| 3200                             MOVE     D0,D1                    ;set for shared memory
1846| 6144                             BSR.S    VFYCHKSM                ;and go do checksum
1848| 4CDF 0103                         @1    MOVEM.L (SP)+,D0-D1/A0    ;restore regs
184C| 4E75                             RTS
184E|                                     .IF BURNIN = 1
184E| ;-----
184E| ; Subroutine to save boot device code to parameter memory.
184E| ;-----
184E|
184E| 48E7 C080                         SAV2PM MOVEM.L D0-D1/A0,-(SP)      ;save regs
1852|                                     .IF NEWTWIG = 0
1852|                                     .ENDC
1852| E908                             LSL.B    #4,D0                    ;rotate device code to upper nibble
1854| 1239 00FC C189                     MOVE.B   DVCCODE,D1             ;read current setting
185A| 0201 000F                         ANDI.B   #$0F,D1                ;clear device indicator

```



```

185E| 8001                OR.B   D1,D0          ;save other data
1860| 13C0 00FC C189      MOVE.B  D0,DVCCODE     ;and write new device code
1866|
1866|                    ; also set for full memory test
1866| 08F9 0006 00FC C18D      BSET   #6,MEMCODE     ;ensure memory test indicator set
186E|
186E|                    ; then compute new checksum
186E|
186E| 207C 00FC C181      MOVEA.L #PMSTRT,A0        ;compute new checksum
1874| 701E                MOVEQ   #PMWRDS-2,D0     ;leaving out checksum word
1876| 6106                BSR.S   WRTSUM
1878| 4CDF 0103          @2      MOVEM.L (SP)+,D0-D1/A0 ;restore regs
187C| 4E75                RTS
187E|
187E|                    ;-----
187E|                    ; Subroutine to write new checksum to parameter memory area
187E|                    ;-----
187E| 3200          WRTSUM  MOVE    D0,D1          ;set for shared memory
1880| 610A          BSR.S   VFYCHKSM
1882| 4643          NOT     D3                    ;compute 2's complement
1884| 5243          ADDQ    #1,D3
1886| 0788 0000      MOVEP   D3,(A0)             ;write as new checksum
188A| 4E75          RTS
188C|
188C|                    .ENDC
188C|
188C|                    ;-----
188C|                    ; Subroutine to verify 16 bit checksum validity for memory contents.
188C|                    ;
188C|                    ; Inputs Required:  A0 = starting address for verify
188C|                    ;                    D0 = # of words-1 to read
188C|                    ;                    D1 = 0 for regular memory (uses MOVE.W)
188C|                    ;                    = nonzero for shared memory (uses MOVEP)
188C|                    ;
188C|                    ; Outputs:           Carry bit set if computed checksum not 0.
188C|                    ;                    D2 = expected checksum (last word read)
188C|                    ;                    D3 = computed checksum
188C|                    ;-----
188C|
188C|          VFYCHKSM
188C| 4282          CLR.L   D2                    ;clear regs for use
188E| 4283          CLR.L   D3
1890| 4A41          CKLOOP  TST    D1             ;shared memory?
1892| 6708          BEQ.S   @1                    ;skip if no
1894| 0508 0000      MOVEP   (A0),D2             ;else read alternate bytes
1898| 5888          ADDQ.L   #4,A0                ;bump address
189A| 6002          BRA.S   @2                    ;skip to do checksum

```



```

189C| 3418      @1      MOVE      (A0)+,D2      ;read words
189E| D642      @2      ADD       D2,D3         ;add to computed checksum
18A0| E35B      ROL       #1,D3         ;rotate for better effectiveness
18A2| 51C8 FFEC      DBF       D0,CKLOOP      ;loop until done
18A6| 4A43      TST       D3             ;expected result = 0
18A8| 6704      BEQ.S     CKXIT
18AA| 003C 0001      ORI.B     #$01,CCR      ;else set error indicator
18AE| 4E75      CKXIT     RTS
18B0|
18B0|           .IF      NEWTWIG = 0
18B0|           .ENDC
18B0|           .IF      USERINT = 1
18B0|           ;-----
18B0|           ; Subroutine to expand boot id read from parameter memory to keycode
18B0|           ; Returns with keycode in D0.
18B0|           ;-----
18B0|
18B0| EXPAND  MOVEM.L D2/A2-A3,-(SP) ;save regs
18B4| 45FA F9C4      LEA       KEYTBL,A2      ;set ptrs to keycode table
18B8| 47FA F9D1      LEA       TBLEND,A3
18BC| 4282      CLR.L     D2             ;use for search id
18BE| B002      @1      CMP.B     D2,D0         ;check for match
18C0| 670C      BEQ.S     @2             ;skip if yes
18C2| 5242      ADDQ      #1,D2         ;incr search id
18C4| 528A      ADDQ.L     #1,A2         ;bump table ptr
18C6| B7CA      CMPA.L     A2,A3         ;at end?
18C8| 66F4      BNE.S     @1             ;loop if not
18CA| 7002      MOVEQ      #PROFILE,D0      ;else set for default boot
18CC| 6002      BRA.S     @3             ;and go to exit
18CE| 1012      @2      MOVE.B     (A2),D0      ;get keycode
18D0| 4CDF 0C04      @3      MOVEM.L (SP)+,D2/A2-A3 ;restore regs
18D4| 4E75      RTS
18D6|
18D6|           ;-----
18D6|           ; Routine to search I/O slots for specific card
18D6|           ; Expects D1 = card id to search for
18D6|           ;           D4 = qualifier for search (mask)
18D6|           ; Returns CC = 0 if card found and
18D6|           ;           D2 = slot #
18D6|           ;-----
18D6|
18D6| SEARCH  MOVEM.L D0/D3,-(SP) ;save regs
18DA| 7400      MOVEQ      #0,D2         ;setup as slot counter
18DC| 7602      MOVEQ      #2,D3         ;set # of slots - 1 to check
18DE| 41F8 0298      LEA       IO1ID,A0      ;get location of saved slot id's
18E2| 5242      @1      ADDQ      #1,D2         ;bump slot #
18E4| 3018      MOVE      (A0)+,D0      ;read slot id

```

```

18E6| C044          AND    D4,D0          ;mask it
18E8| B041          CMP     D1,D0          ;match?
18EA| 6706          BEQ.S   @2             ;skip if yes
18EC| 51CB FFF4      DBF     D3,@1         ;
18F0| 4A42          TST     D2             ;set nonzero status if no match
18F2| 4CDF 0009      @2    MOVEM.L (SP)+,D0/D3 ;restore regs
18F6| 4E75          RTS                     ;exit
18F8|
18F8|          .PAGE
18F8|
18F8|          ;-----
18F8|          ; Routines to display boot icon menu
18F8|          ;-----
18F8|
18F8|          BOOTMENU
18F8| 0238 000F 02A2      ANDI.B  #$0F,STATFLGS ;initialize flags
18FE| 4278 053A      CLR      RectCnt        ;clear active rectangle counter
1902| 7C01          MOVEQ    #1,D6            ;set min # of boot alternates      CHG009
1904|
1904|          ; i.e., at least have lower drive
1904|          ; to boot from
1904| 4A38 02AF      TST.B     SYSTYPE          ;check system type      CHG009
1908| 6602          BNE.S     @10             ;Skip if Lisa 2      CHG009
190A| 5246          ADDQ      #1,D6            ;else incr count for upper floppy drive CHG009
190C|
190C| 7401          @10    MOVEQ    #1,D2            ;set flag to do wait if needed      RM011
190E| 6100 0126      BSR      CHKPROFILE        ;go check for attached Profile
1912| 6602          BNE.S     @1             ;skip if not there
1914| 5246          ADDQ      #1,D6            ;else bump boot count
1916|
1916| 4284          @1      CLR.L     D4             ;set flag for no status check
1918| 6100 F9EC      BSR      RDSLOTS          ;go scan the I/O slots
191C| 41F8 0298      LEA       IO1ID,A0          ;check results
1920| 3018          MOVE      (A0)+,D0          ;read first ID
1922| 6A12          BPL.S     @2             ;skip if not bootable or not there
1924| 247C 00FC 0001  MOVE.L    #SLOT1L,A2        ;set slot address
192A| 6100 026E      BSR      RDSLT           ;go check if any icons
192E| 6506          BCS.S     @2             ;skip if any error
1930| 0243 0003      ANDI      #$03,D3          ;else clear don't care bits (max count = 2)
1934| DC43          ADD       D3,D6            ;and add to icon count
1936|
1936| 3018          @2      MOVE      (A0)+,D0        ;check slot 2
1938| 6A12          BPL.S     @3             ;skip if not bootable
193A| 247C 00FC 4001  MOVE.L    #SLOT2L,A2        ;set slot address
1940| 6100 0258      BSR      RDSLT           ;go check if any icons
1944| 6506          BCS.S     @3             ;skip if any error
1946| 0243 0003      ANDI      #$03,D3          ;else clear don't care bits (max count = 2)
194A| DC43          ADD       D3,D6            ;and add to icon count
194C|

```



```

194C| 3018          @3      MOVE    (A0)+,D0          ;check final slot
194E| 6A12          BPL.S    @4
1950| 247C 00FC 8001 MOVE.L    #SLOT3L,A2          ;set slot address
1956| 6100 0242      BSR      RDSLT          ;go check if any icons
195A| 6506          BCS.S    @4          ;skip if any error
195C| 0243 0003      ANDI     #$03,D3          ;else clear don't care bits (max count = 2)
1960| DC43          ADD      D3,D6          ;and add to icon count
1962|
1962|          ; set starting icon display address according to boot count
1962|
1962| 0C06 000A      @4      CMP.B    #10,D6          ;max of 10 icons in menu
1966| 6F02          BLE.S    @5          ;skip if OK
1968| 7C0A          MOVEQ    #10,D6          ;else set max count
196A|
196A|          ; now display blank boot icon menu
196A| 7012      @5      MOVEQ    #BMENUWIDTH,D0      ;set menu parameters
196C| 2206          MOVE.L    D6,D1          ;get count of entries
196E| C2FC 0022      MULU     #BMENULEN,D1          ;length depends on number of entries
1972| 47FA 2524      LEA      STRMSG,A3          ;set menu heading
1976| 6100 0E58      BSR      DSPMENUBOX          ;and go display the menu
197A| 31FC 05A2 0530 MOVE      #MENUSTRT,MenuBase ;setup base menu address
1980| 31FC 0658 0532 MOVE      #MENU1MSG,IconAddr ;and display pt for first entry
1986|
1986|          ; next fill in the menu entries with icons and alternate keycodes
1986|          ; D0 set with system type
1986|          ICONCHK
1986| 1038 02AF      MOVE.B    SYSTYPE,D0          ;read system type          CHG009/CHG029
198A| 6714          BEQ.S     @1          ;skip if Lisa 1.0          CHG009
198C| 0C00 0003      CMP.B     #3,D0          ;else check if internal disk      CHG009/CHG029
1990| 661C          BNE.S     @2          ;skip if not - no upper icon      CHG009/CHG029
1992|
1992| 4282          CLR.L     D2          ;else set for no wait          CHG009
1994| 6100 00A0      BSR      CHKPROFILE          ; and check if integral disk      CHG009
1998|          ; installed          CHG009
1998| 6614          BNE.S     @2          ;skip if not          CHG009
199A| 45FA 213E      LEA      UPPER,A2          ;set icon ptr for integral disk      CHG009
199E| 6006          BRA.S     @3          ;and go display          CHG009
19A0|
19A0| 45FA 217C      @1      LEA      DRIVEN,A2          ;set icon ptr for drive          CHG009
19A4| 7201          MOVEQ    #1,D1          ;set drive id #          CHG009
19A6| 7400          @3      MOVEQ    #TWIG1,D2          ;set id code          CHG009
19A8| 76FF          MOVEQ    #-1,D3          ;set compressed icon indicator
19AA| 6100 00E8      BSR      DSPMNTTRY          ;display the entry
19AE|
19AE| 45FA 216E      @2      LEA      DRIVEN,A2          ;set icon ptr for drive          CHG009
19B2| 7202          MOVEQ    #2,D1          ; and drive id #          CHG009
19B4| 7401          MOVEQ    #TWIG2,D2          ;set id code

```

```

19B6| 76FF          MOVEQ    #-1,D3          ;set compressed indicator
19B8| 6100 00DA      BSR      DSPMNTY      ;display the entry
19BC|
19BC| 4282          CLR.L    D2              ;set flag for no wait          RM011
19BE| 0C00 0003      CMP.B    #3,D0          ;skip check if internal disk    CHG009/CHG029
19C2| 6712          BEQ.S    SCNSLTS        ;                      CHG009/CHG029
19C4|
19C4| 6100 0070      BSR      CHKPROFILE    ;else check if external disk attached
19C8| 660C          BNE.S    SCNSLTS        ;skip if not
19CA| 45FA 20D3      LEA      PROICON,A2     ;else set icon ptr for Profile
19CE| 7402          MOVEQ    #PROFILE,D2     ;set id code
19D0| 76FF          MOVEQ    #-1,D3          ;set compressed indicator
19D2| 6100 00C0      BSR      DSPMNTY      ;display the entry
19D6|
19D6|                ; check for bootable devices in slots (what a pain!)
19D6|
19D6| 4280          SCNSLTS CLR.L    D0          ;clear for use
19D8| 41F8 0298      LEA      IO1ID,A0        ;set ptr for slot id
19DC| 3018          MOVE     (A0)+,D0        ;get id
19DE| 6A0E          BPL.S    CHKS2          ;skip if not bootable or not there
19E0| 247C 00FC 0001 MOVE.L    #SLOT1L,A2     ;else set slot address
19E6| 7201          MOVEQ    #1,D1          ;set slot # for generic display if no ROM icon
19E8| 7403          MOVEQ    #IO1PORT1,D2    ;set base boot id for slot
19EA| 6100 0158      BSR      CHKSLOT        ;go check slot and display icons
19EE|
19EE| 3018          CHKS2    MOVE     (A0)+,D0    ;read next id
19F0| 6A0E          BPL.S    CHKS3          ;skip if not bootable or not there
19F2| 247C 00FC 4001 MOVE.L    #SLOT2L,A2     ;else set slot address
19F8| 7202          MOVEQ    #2,D1          ;set slot # for generic display
19FA| 7406          MOVEQ    #IO2PORT1,D2    ;set base boot id for slot
19FC| 6100 0146      BSR      CHKSLOT        ;go check slot and display icons
1A00|
1A00| 3018          CHKS3    MOVE     (A0)+,D0    ;read slot 3 id
1A02| 6A10          BPL.S    WT4BOOT        ;skip if not bootable or not there
1A04| 247C 00FC 8001 MOVE.L    #SLOT3L,A2     ;else set slot address
1A0A| 7203          MOVEQ    #3,D1          ;set slot # for generic display
1A0C| 343C 0009      MOVE     #IO3PORT1,D2    ;set base boot id for slot
1A10| 6100 0132      BSR      CHKSLOT        ;go check slot 3 and display icons
1A14|
1A14|                ;-----
1A14|                ; Menu displayed - now wait for operator selection
1A14|                ;-----
1A14|                WT4BOOT
1A14| 6100 15F8          BSR      CursorDisplay ;display cursor on screen
1A18| 08F8 0005 02A2 BSET     #CHKCMD,STATFLGS ;set flag for CMD key check
1A1E| 6100 1226          BSR      GETINPUT     ;go wait for user input
1A22| 6500 0CC2          BCS      GETERR      ;skip if error

```



```

1A26| 6100 F836          BSR    XLATE          ;translate and save boot id
1A2A| 6100 15BE          BSR    CursorHide      ;remove cursor from screen
1A2E| 6100 16AA          BSR    CLRDESK        ;clear desktop
1A32| 6000 FCB2          BRA     BOOTCHK        ;and go start boot
1A36|
1A36|          .PAGE
1A36|          ;-----
1A36|          ; Routine to check for Profile attached to built-in parallel port.
1A36|          ; Checks for Profile connected (OCD) and tries an initial handshake to
1A36|          ; ensure the device is a Profile.
1A36|          ;
1A36|          ; Inputs: D2 = nonzero if full wait for Profile ready should be done
1A36|          ; Outputs: Zero condition code bit cleared if error
1A36|          ;-----
1A36|
1A36|          CHKPROFILE
1A36| 48E7 AA80          MOVEM.L A0/D0/D2/D4/D6,-(SP)      ;save regs          RM011
1A3A| 6100 05B4          BSR    PROINIT          ;init for Profile access
1A3E| 664E          BNE.S    @9              ;skip if not attached
1A40|
1A40| 6100 06F0          BSR    WFNBSY3          ;wait for not busy          RM000
1A44| 4A00          TST.B    D0              ;check return code
1A46| 671E          BEQ.S    @0              ;skip if OK
1A48| 4A02          TST.B    D2              ;do full wait?          RM011
1A4A| 6740          BEQ.S    @7              ;skip if not          RM011
1A4C|
1A4C| 4280          CLR.L    D0              ;else reset error code for retry
1A4E| 6100 0456          BSR    WAITALRT        ;output wait alert
1A52| 6100 06D6          BSR    WFNBSY2          ;try wait for normal profile boot time  CHG019
1A56| 48E7 8080          MOVEM.L A0/D0,-(SP)      ;save regs
1A5A| 6100 167E          BSR    CLRDESK        ;clear desktop
1A5E| 4CDF 0101          MOVEM.L (SP)+,A0/D0      ;restore regs
1A62| 4A00          TST.B    D0
1A64| 661E          BNE.S    @8              ;exit if still not ready
1A66|
1A66| 0210 00EF          @0    ANDI.B    #$EF,ORB2(A0)      ;set command = true
1A6A| 6100 0690          BSR    WFBSY          ;then get initial Profile response  RM016
1A6E| 660A          BNE.S    @1              ;skip if error
1A70|
1A70| 0C28 0001 0078          CMPI.B    #1,PORTA2(A0)      ;check for expected '01' response
1A76| 6702          BEQ.S    @1              ;skip if OK
1A78| 7052          MOVEQ    #BADRSP,D0          ;else set error code
1A7A|
1A7A| 7600          @1    MOVEQ    #0,D3              ;send '0' response to reset Profile
1A7C| 6100 0690          BSR    SENDRSP
1A80| 6100 06A0          BSR    WFNBSY          ;wait until command taken
1A84|

```




```

1A84| 4228 0018      @8      CLR.B   DDRA2(A0)           ;set port A bits to input
1A88| 0010 0018      ORI.B   #$18,ORB2(A0)         ;and set dir=in, cmd=false
1A8C|
1A8C| 4A40           @7      TST     D0              ;set return code
1A8E|
1A8E| 4CDF 0155      @9      MOVEM.L (SP)+,A0/D0/D2/D4/D6 ;restore                      RM011
1A92| 4E75          RTS                      ; and exit
1A94|
1A94|
1A94| ; -----
1A94| ; Subroutine to invoke display of boot menu entry
1A94| ; Inputs:
1A94| ;     D2 = boot id
1A94| ;     A2 = ptr to icon
1A94| ; Outputs:
1A94| ;     Location MenuBase updated with address for next menu "box"
1A94| ; Side Effects:
1A94| ;     None
1A94| ; -----
1A94|
1A94| DSPMNTRY
1A94| 48E7 8040      MOVEM.L D0/A1,-(SP)      ;save boot id and addr ptr
1A98| 3002          MOVE     D2,D0          ;get boot id
1A9A| 6100 FE14      BSR     EXPAND          ;go convert boot id to keycode
1A9E| 3278 0530      MOVE     MenuBase,A1    ;get address for display of entry
1AA2| 6106          BSR.S    ICONMENU       ;go display in menu
1AA4| 4CDF 0201      MOVEM.L (SP)+,D0/A1    ;restore boot id and addr ptr
1AA8| 4E75          RTS                      ;and exit
1AAA|
1AAA|
1AAA| ; -----
1AAA| ; Subroutine to display icon menu on screen.  Creates "active rectangle
1AAA| ; table" as entries are made.
1AAA| ; Inputs:
1AAA| ;     D0 = alternate keycode
1AAA| ;     D3 = compressed icon indicator
1AAA| ;     A1 = address for start of next menu "box"
1AAA| ;     A2 = ptr to icon
1AAA| ; Outputs:
1AAA| ;     A1 = ptr for display of next menu entry
1AAA| ; Side Effects:
1AAA| ;     None
1AAA| ; -----
1AAA|
1AAA| ICONMENU
1AAA| 48E7 F8A4      MOVEM.L D0-D4/A0/A2/A5,-(SP)
1AAE|
1AAE| ; first save icon coordinates in active rectangle table
1AAE|

```



```

1AAE| 41F8 053A      LEA      RectTable,A0      ;get ptr to active rect table
1AB2| 3410           MOVE      (A0),D2      ;get current count of rect's
1AB4| C4FC 0005      MULU      #5,D2      ;five entries per rect
1AB8| D442           ADD       D2,D2      ;double for word index
1ABA| 5258           ADDQ      #1,(A0)+    ;incr for new rect
1ABC| 3180 2000      MOVE      D0,0(A0,D2.W) ;save keycode id for new rect
1AC0| 6100 0C58      BSR       KeyToAscii ;convert keycode to Ascii
1AC4| 3800           MOVE      D0,D4      ;save for later display
1AC6|
1AC6|               ; compute X,Y pixel coordinates from starting address
1AC6|
1AC6| 6100 193E      BSR       GETROWCOL    ;get pixel row, byte col
1ACA| CCFC 0008      MULU      #8,D6      ;convert to pixel col
1ACE| 3186 2002      MOVE      D6,2(A0,D2.W) ;save upper left X
1AD2| 3185 2004      MOVE      D5,4(A0,D2.W) ; and Y coordinates
1AD6|
1AD6| 303C 0090      MOVE      #<MENUWIDTH*8>,D0 ;width in pixels of menu entry
1ADA| DC40           ADD       D0,D6      ;compute and save
1ADC| 3186 2006      MOVE      D6,6(A0,D2.W) ; lower Y coordinate
1AE0| 0645 0022      ADD       #<ICONHIGH+2>,D5 ;compute and save
1AE4| 3185 2008      MOVE      D5,8(A0,D2.W) ; lower X coordinate
1AE8|
1AE8|               ; now do icon display
1AE8|
1AE8| 9DCE           SUBA.L     A6,A6      ;clear for use
1AEA| 3C78 0532      MOVE      IconAddr,A6 ;get address for icon display
1AEE| 224E           MOVE.L    A6,A1      ;save for later use
1AF0| DDF8 0110      ADD.L      SCRNBASE,A6 ;convert to screen address
1AF4| 4A03           TST.B     D3      ;check for compressed icon
1AF6| 6A14           BPL.S     @1      ;skip if not
1AF8| 6100 1AE8      BSR       DSPICON    ;go do display
1AFC|
1AFC| 41FA 2020      LEA       DRIVEN,A0      ;displaying drive?
1B00| B5C8           CMPA.L     A0,A2      ;
1B02| 660C           BNE.S     @2      ;skip if not
1B04| 2A49           MOVE.L     A1,A5      ;else set icon display address
1B06| 6100 195C      BSR       DSPNUM     ; and display with id #
1B0A| 6004           BRA.S     @2      ;skip to continue
1B0C|
1B0C| 6100 1916      @1      BSR       DSPRGICON ;display an uncompressed icon
1B10|
1B10|               ; now display the alternate keycode
1B10|
1B10| D2FC 0445      @2      ADD       #ALTKYADDR,A1 ;set starting display pt
1B14| 6100 18F0      BSR       GETROWCOL    ;convert to row, col
1B18| 4240           CLR       D0      ;first display the apple
1B1A| 6100 1C1E      BSR       DSPVAL

```

```

1B1E| 3004                MOVE    D4,D0          ;get Ascii char
1B20| 6100 1C18          BSR      DSPVAL         ;and display it
1B24|
1B24|                ; finally compute the next menu entry and icon display address
1B24|
1B24| @3      MOVE    MenuBase,A1          ;get base address for the entry
1B28| D2FC 0BF4          ADD      #BMenuSpc,A1    ;space to next col
1B2C| 31C9 0530          MOVE     A1,MenuBase     ;and save for next entry
1B30|
1B30| 3278 0532          MOVE     IconAddr,A1     ;else get this icon's address
1B34| D2FC 0BF4          ADD      #BMenuSpc,A1    ;and bump to next spot in column
1B38| 31C9 0532          MOVE     A1,IconAddr     ;and do update
1B3C| 6000              BRA.S    @4
1B3E|
1B3E| 4CDF 251F          @4      MOVEM.L (SP)+,D0-D4/A0/A2/A5
1B42| 4E75              RTS
1B44|
1B44|                ;-----
1B44|                ; Routine to check slots for icons and do display or do generic display.
1B44|                ; Inputs:
1B44|                ;     D0 = card id
1B44|                ;     D1 = slot #
1B44|                ;     D2 = first boot id for slot
1B44|                ;     A1 = address for icon display
1B44|                ;     A2 = slot address
1B44|                ; Outputs:
1B44|                ;     Returns with carry bit set if error.
1B44|                ; Side Effects:
1B44|                ;     None
1B44|                ;-----
1B44|
1B44| 48E7 F8E0          CHKSLT MOVEM.L D0-D4/A0-A2,-(SP) ;save regs
1B48| 0800 000D          BTST     #ICBIT,D0        ;icon available?
1B4C| 661A              BNE.S    CHKICONS         ;skip if yes
1B4E|
1B4E|                ; no icons available - display slot # and display generic slot card icon
1B4E|                LEA      XCARD,A2            ;point to generic icon
1B52| 76FF              MOVEQ    #-1,D3          ;set compressed flag
1B54| 3278 0532          MOVE     IconAddr,A1     ;get display address for later use
1B58| 6100 FF3A          BSR      DSPMNTY        ;go display entry
1B5C| 45FA 1EB6          LEA      XCARD,A2        ;set icon ptr
1B60| 2A49              MOVE.L    A1,A5           ;get icon address
1B62| 6100 1900          BSR      DSPNUM         ;display slot #
1B66| 602C              BRA.S    CHKSXIT         ;and exit
1B68|
1B68|                ; Slot has icon - read ROM and get ptr to desired icon
1B68|
1B68|                CHKICONS

```



```

1B68| 6130          BSR.S   RDSLT      ;go read slot
1B6A| 6528          BCS.S   CHKSXIT   ;exit if error
1B6C| 227C 0001 FFFC MOVE.L   #ADR128K-4,A1 ;set base address of I/O slot ROM code
1B72| 2803          MOVE.L   D3,D4     ;save icon count
1B74| 0244 0003     ANDI     #$03,D4   ;isolate count (max = 2)
1B78| 204A          MOVE.L   A2,A0     ;get code ptr
1B7A| 3218          MOVE     (A0)+,D1   ;get icon offset
1B7C| 2449          MOVE.L   A1,A2     ;get base address
1B7E| D4C1          ADD      D1,A2     ;add offset to set up icon ptr
1B80| 6100 FF12     BSR      DSPMNTRY  ;display as menu entry
1B84|
1B84| 5344          SUBQ      #1,D4      ;more than one icon?
1B86| 670C          BEQ.S    CHKSXIT   ;skip if not
1B88| 3218          MOVE     (A0)+,D1   ;else get ptr to second icon
1B8A| 5242          ADDQ      #1,D2     ;bump boot id ptr for slot
1B8C| 2449          MOVE.L   A1,A2     ;restore base address
1B8E| D4C1          ADD      D1,A2     ;set up icon address
1B90| 6100 FF02     BSR      DSPMNTRY  ;display as menu entry
1B94|
1B94| 4CDF 071F     CHKSXIT MOVEM.L (SP)+,D0-D4/A0-A2 ;restore regs and exit
1B98| 4E75          RTS
1B9A|
1B9A| ;-----
1B9A| ; Routine to read I/O slot ROM's and get icon count if any.
1B9A| ;
1B9A| ; Expects D0 = boot id
1B9A| ;       A2 = slot address
1B9A| ;
1B9A| ; Returns D3 = icon count
1B9A| ;       A2 = address of ptr to first icon if more than one
1B9A| ;-----
1B9A|
1B9A| 48E7 A000     RDSLT    MOVEM.L D0/D2,-(SP) ;save boot id's
1B9E| 0800 000D     BTST     #ICBIT,D0     ;any icons stored in ROM?
1BA2| 6720          BEQ.S     @2           ;skip if none
1BA4| 4284          CLR.L    D4           ;set flag for no status check
1BA6| 6100 0624     BSR      RDIOSLT     ;and go read ROM on slot
1BAA| 651A          BCS.S     @1         ;skip if error
1BAC|
1BAC| 4283          CLR.L    D3           ;clear for use
1BAE| 3639 0002 0004 MOVE     ICONPTR,D3   ;get icon ptr
1BB4| 08C3 0000     BSET     #0,D3       ;must be odd address
1BB8| 247C 0001 FFFC MOVE.L   #ADR128K-4,A2 ;set base address
1BBE| D5C3          ADD.L    D3,A2       ;set actual address
1BC0| 161A          MOVE.B   (A2)+,D3   ;read icon count
1BC2| 6002          BRA.S    @1
1BC4|

```



```

1BC4| 7601          @2      MOVEQ    #1,D3          ;set default icon count
1BC6|
1BC6| 4CDF 0005      @1      MOVEM.L (SP)+,D0/D2      ;restore boot id's
1BCA| 4E75          RTS
1BCC|              .ENDC                          ;{USERINT}
1BCC|              .PAGE
1BCC|              ;-----
1BCC|              ; Do default boot from Twiggys specified drive.
1BCC|              ; Assumes regs:
1BCC|              ;     D0 = drive #
1BCC|              ; Following assumptions are made for power-up status:
1BCC|              ;     1) No interrupt from disk unless diskette inserted or button pushed
1BCC|              ;-----
1BCC|
1BCC| 47FA F5B8      TWGBOOT LEA      DSKVCT,A3          ; first set up bus error vector
1BD0| 21CB 0008      MOVE.L    A3,BUSVCTR
1BD4|
1BD4|              .IF USERINT = 0
1BD4|              .ELSE
1BD4| 11C0 0535      MOVE.B    D0,DRIVE          ;save drive id
1BD8| 6100 02CC      BSR      WAITALRT          ;display wait icon
1BDC|              .ENDC
1BDC|
1BDC| 207C 00FC C001      MOVE.L    #DISKMEM,A0          ; set ptr to controller memory
1BE2| 227C 0001 FFF4      MOVE.L    #TWGHDR,A1          ; set ptr to load header
1BE8| 247C 0002 0000      MOVE.L    #TWGHDR+12,A2        ; and ptr to load data
1BEE| 4281          CLR.L    D1                  ; set drive/side/sector/track ptr
1BF0| 0280 0000 00FF      ANDI.L    #$00FF,D0          ; mask off junk
1BF6|
1BF6| 4A40          TST      D0                  ;enable only drive selected
1BF8| 6608          BNE.S    @1
1BFA| 117C 0008 0002      MOVE.B    #$08,CMD(A0)        ;enable drive #1
1C00| 6006          BRA.S    @2
1C02| 117C 0080 0002      @1      MOVE.B    #$80,CMD(A0)    ;enable drive #2
1C08|
1C08| E098          @2      ROR.L    #8,D0            ; get actual drive desired
1C0A| 8280          OR.L     D0,D1              ; and save for shared mem format
1C0C|
1C0C| 10BC 0086      MOVE.B    #ENBLINT,(A0)          ;go do enable
1C10| 6100 01F2      BSR      CMDCHK              ;wait until cmd taken
1C14| 6500 0082      BCS      DSKTIMERR          ;skip if timeout error
1C18| 267C 00FC DD81      MOVE.L    #VIA1BASE,A3          ;else enable
1C1E| 08AB 0004 0004      BCLR     #FDIR,DDRB1(A3)      ; FDIR
1C24|
1C24|              ;      BTST     #FDIR,(A3)          ;FDIR present?
1C24|              ;      BEQ.S    DOREAD            ;skip if no to do read
1C24|

```



```

1C24| 6100 024C      CLRINT BSR      CLRFDIR      ;clear interrupts
1C28| 6500 006E      BCS      DSKTIMERR      ;exit if timeout error
1C2C|
1C2C|      .PAGE
1C2C|      ;-----
1C2C|      ; Read boot data - retry on sector 0 if needed.
1C2C|      ;-----
1C2C|
1C2C|      DOREAD
1C2C| 4240      CLR      D0      ; set speed value
1C2E| 6100 0140      BSR      TWGRD      ; go read sector 0      RM000
1C32| 640A      BCC.S      @1      ; skip if OK
1C34| 0C00 0027      CMP.B      #TIMOUT,D0      ; timeout error?
1C38| 6700 0060      BEQ      DSKCHK      ; exit if yes
1C3C| 600A      BRA.S      RDRETRY      ; else go do retry
1C3E|
1C3E| 3029 0004      @1      MOVE      FILEID(A1),D0      ; else get file ID
1C42| 0C40 AAAA      CMP      #BOOTPAT,D0      ; is it a boot file?
1C46| 6724      BEQ.S      RDSCTR1      ; skip if yes
1C48|
1C48|      ; Do retry by reading track 1 to try to get head properly aligned, then
1C48|      ; retry reading track 0
1C48| 123C 0001      RDRETRY MOVE.B      #1,D1      ; set for track 1
1C4C| 4240      CLR      D0      ; set speed value
1C4E| 6100 0120      BSR      TWGRD      ; go do read      RM000
1C52| 6546      BCS.S      DSKCHK      ; exit if second error
1C54| 4201      CLR.B      D1      ; else retry track 0
1C56| 4240      CLR      D0      ; set speed value
1C58| 6100 0116      BSR      TWGRD      ; go do read      RM000
1C5C| 653C      BCS.S      DSKCHK      ; exit if error
1C5E|
1C5E|      ; Now check again for a valid boot track
1C5E| 3029 0004      MOVE      FILEID(A1),D0      ; get file ID
1C62| 0C40 AAAA      CMP      #BOOTPAT,D0      ; is it a boot file?
1C66| 6704      BEQ.S      RDSCTR1      ; skip if yes
1C68| 7026      MOVEQ      #BADTHDR,D0      ; set error code
1C6A| 602E      BRA.S      DSKCHK      ; and exit
1C6C|
1C6C|      RDSCTR1
1C6C|      .IF NEWTWIG = 0
1C6C|      .ENDC
1C6C|
1C6C| 42B8 01B4      CLR.L      BOOTDATA      ;set for no error
1C70| 47FA 009C      LEA      DSKERR2,A3      ;set up vectors in case of bad diskette
1C74| 49FA 009E      LEA      DSKERR3,A4
1C78| 6100 0236      BSR      VCTRINIT
1C7C| 247C 0002 0000      MOVE.L      #TWGDATA,A2      ;set data ptr (software view)

```



```

1C82|
1C82|           ; Do keyboard/mouse reset before continuing boot process
1C82|
1C82|           STRTBOOT
1C82| 48E7 8080           MOVEM.L A0/D0,-(SP)           ;save regs
1C86| 6100 EE22           BSR      RSTKBD             ;send reset signal
1C8A| 6100 EE38           BSR      CLRRST            ;then clear it
1C8E| 4238 02B0           CLR.B    KBDQ             ;clear first byte of keyboard queue
1C92| 4CDF 0101           MOVEM.L (SP)+,A0/D0         ;restore regs
1C96| 4ED2                JMP      (A2)                ;and away we go ...
1C98|
1C98|           .PAGE
1C98|
1C98|           ;-----
1C98|           ; Error occurred - output error message and go to monitor
1C98|           ;-----
1C98|
1C98|           DSKTIMERR
1C98| 7027                MOVEQ    #TIMOUT,D0           ;set timeout error code
1C9A| 11C0 01B4           DSKCHK  MOVE.B  D0,BOOTDATA   ;save the error status
1C9E| 0C00 0027           CMPI.B  #TIMOUT,D0           ;timeout?
1CA2| 6720                BEQ.S    DSKERR             ;skip if yes
1CA4| 6100 01CC           BSR      CLRFDIR            ;ensure intrpt cleared
1CA8| 6512                BCS.S    DSKOUT             ;exit if error
1CAA|
1CAA|           .IF USERINT = 0
1CAA|           .ENDC
1CAA|
1CAA| 11E8 00BA 01B6       DSKBAD  MOVE.B  CHKCNT(A0),BOOTDATA+2 ; and data checksum count
1CB0|
1CB0|           .IF NEWTWIG = 1
1CB0| 11E8 00C4 01B5       MOVE.B  CHKCNT2(A0),BOOTDATA+1 ; and address checksum count
1CB6| 11E8 0058 01B7       MOVE.B  RTRYCNT(A0),BOOTDATA+3 ; and retry count
1CBC|           .ENDC
1CBC|
1CBC| 6100 0198           DSKOUT  BSR      EJCTDSK             ;then eject the disk
1CC0|
1CC0| 6100 0084           DSKDIS  BSR      DSABLDISK          ;disable both drives
1CC4|
1CC4| 6100 EA1C           DSKERR  BSR      SETBUSVCT          ; restore default bus error vector      RM000
1CC8|
1CC8|           .IF USERINT = 0
1CC8|           .ELSE
1CC8| 6100 0092           BSR      CHKDRIVE             ;go determine drive
1CCC| 1038 01B4           MOVE.B  BOOTDATA,D0          ;get error code
1CD0| 0C00 0007           CMP.B    #NODISK,D0          ;no disk error?
1CD4| 6614                BNE.S    @1                ;skip if not
1CD6|

```



```

1CD6|                                     ;2 statements deleted                                CHG009
1CD6|
1CD6| 45FA 1E7E                        LEA     INSERTD,A2      ;set icon for insert rqst      CHG009
1CDA| 6100 1850                        BSR     DSPALRTICON    ;display basic icon          CHG009
1CDE| 3A7C 287E                        MOVE    #ERRSTRT,A5    ;set display pt for id #    CHG009
1CE2| 6100 1780                        BSR     DSPNUM        ; and display it           CHG009
1CE6| 6000 0256                        BRA      BFAIL2      ;then go signal user
1CEA|
1CEA| 0C00 0017                @1      CMP.B   #RDWRERR,D0    ;read error?
1CEE| 670C                        BEQ.S    @2           ;skip if yes
1CF0| 0C00 0026                CMP.B   #BADTHDR,D0    ;bad file id?
1CF4| 6706                        BEQ.S    @2           ;
1CF6| 0C00 004B                CMP.B   #EBOOT,D0     ;boot error?
1CFA| 6606                        BNE.S    @3           ;skip if not
1CFC|
1CFC| 6100 1752                @2      BSR     DSPNUMICON    ;display diskette icon with id #
1D00| 6008                        BRA.S    TBOOTERR     ;and exit
1D02|
1D02| 45FA 1C47                @3      LEA     IOBRD,A2    ;error must be on I/O board
1D06| 6100 17D2                BSR     DSPERRICON    ;display icon
1D0A|                                .ENDC
1D0A|
1D0A|                                TBOOTERR
1D0A|                                .IF  USERINT = 0
1D0A|                                .ELSE
1D0A|                                BRA      BOOTFAIL    ;and go signal boot failure
1D0A| 6000 022E                                .ENDC
1D0E|
1D0E|
1D0E|                                ;-----
1D0E|                                ;  Handler for Twiggy boot errors
1D0E|                                ;-----
1D0E|                                DSKERR2 BSR$4    SAVEXCP      ;go save exception info
1D0E| 49FA 0004                #          LEA     @1,A4
1D12| 6008                #          BRA.S    SAVEXCP
1D14|                #@1
1D14|
1D14|                                DSKERR3 BSR$4    BTERR        ;regroup
1D14| 49FA 0004                #          LEA     @1,A4
1D18| 6010                #          BRA.S    BTERR
1D1A|                #@1
1D1A| 60A0                BRA.S    DSKOUT      ;and display error
1D1C|
1D1C|                                SAVEXCP
1D1C| 31DF 0280                MOVE    (SP)+,EXCFC    ;save function code
1D20| 21DF 0282                MOVE.L   (SP)+,EXCADR    ;and address
1D24| 31DF 0286                MOVE    (SP)+,EXCIR    ;and instruction reg
1D28|                                RTS$4

```




```

1D28| 4ED4          #          JMP          (A4)
1D2A|
1D2A| 31DF 0288      BTERR  MOVE      (SP)+,EXCSR      ;save status reg
1D2E| 21DF 028A      MOVE.L  (SP)+,EXCPC      ;and pc
1D32| 6100 E306      BSR      SAVEREGS      ;save regs
1D36| 3E7C 0480      MOVEA   #STKBASE,SP      ;reset stack pointer
1D3A| 6100 E970      BSR      SETVCTRS      ;reinit vectors
1D3E| 704B          MOVEQ    #EBOOT,D0      ;set boot error
1D40| 11C0 01B4      MOVE.B   D0,BOOTDATA    ;and save
1D44| RTS4
1D44| 4ED4          #          JMP          (A4)
1D46|
1D46| ;-----
1D46| ; Subroutine to disable interrupts from both Twiggly drives
1D46| ; Inputs:
1D46| ;     None
1D46| ; Outputs:
1D46| ;     Carry bit set if timeout error (in CMDCHK).
1D46| ; Side Effects:
1D46| ;     A0 trashed (other regs trashed in CMDCHK)
1D46| ;-----
1D46|
1D46| DSABLDISK
1D46| 207C 00FC C001    MOVE.L   #DISKMEM,A0      ;set ptr to shared memory
1D4C| 117C 0088 0002    MOVE.B   #$88,CMD(A0)    ;disable ints from both drives
1D52| 10BC 0087    MOVE.B   #DSABLINT,(A0)
1D56| 6100 00AC    BSR      CMDCHK      ;wait for command to be taken
1D5A| 4E75          RTS              ;then return
1D5C|
1D5C| ;-----
1D5C| ; Subroutine to determine drive # in error and get icon ptr
1D5C| ; Inputs:
1D5C| ;     Location DRIVE = drive id # (0 or $80)
1D5C| ; Outputs:
1D5C| ;     A2 = ptr to diskette icon
1D5C| ;     D1 = id #
1D5C| ; Side Effects:
1D5C| ;     None
1D5C| ;-----
1D5C|
1D5C| CHKDRIVE
1D5C| 45FA 1FC0    LEA        DISKETTE,A2      ;set ptr for diskette icon
1D60| 1238 0535    MOVE.B   DRIVE,D1      ;get drive id
1D64| 4A01          TST.B    D1              ; drive #1?
1D66| 6604          BNE.S   @1              ; skip if no
1D68|
1D68| .IF USERINT = 0

```



```

1D68|          .ELSE
1D68| 7201      MOVEQ    #1,D1          ; else set up id code
1D6A|          .ENDC
1D6A|
1D6A| 6002      BRA.S    @2
1D6C|          @1
1D6C|          .IF USERINT = 0
1D6C|          .ELSE
1D6C| 7202      MOVEQ    #2,D1          ; else set up id code
1D6E|          .ENDC
1D6E|          @2
1D6E| 4E75      RTS              ;exit
1D70|
1D70|          .IF    EXTERNAL = 1
1D70|          .ENDC
1D70|          .PAGE
1D70|          ;-----
1D70|          ; Read a Twiggy sector routine.  Uses hardware view of the world, with
1D70|          ; 12 bytes for header and 256 bytes (512 for new format) of data per sector.
1D70|          ; Also assumes registers:
1D70|          ; D0 = speed (for new Twiggy code)    A0 = disk shared memory address
1D70|          ; D1 = drive/side/sector/track        A1 = address to load header(12 bytes)
1D70|          ; D2 = timeout for read                A2 = address to load data(256 or 512 bytes)
1D70|          ; D3 = scratch                        A3 = VIA address for FDIR
1D70|          ; If error, returns with carry bit set and error code in D0.
1D70|          ;-----
1D70|
1D70| TWGRD
1D70| 243C 00C0 0000      MOVE.L  #FDIRTIME,D2      ; set default timeout value          RM000
1D76|
1D76| TWGREAD
1D76|          .IF TWIGGY = 1
1D76| 48E7 1078      MOVEM.L D3/A1-A4,-(SP)      ; save regs
1D7A|          DISABLE          ; disable interrupts
1D7A|          #          MOVE      SR,-(SP)
1D7C|          #          ORI      #$0700,SR
1D80|          BSR      CHKFDIR          ;ensure no ints pending
1D84|          MOVEP.L D1,DRV(A0)      ; set disk ptrs
1D88|          .IF    NEWTWIG = 1
1D88| 1140 000C      MOVE.B  D0,SPEED(A0)      ;set speed value
1D8C|          .ENDC
1D8C| 4228 0002      MOVE.B  #READS,CMD(A0)      ; set for read operation
1D90|          MOVE.B  #EXRW,(A0)      ; and go do it
1D94|          BSR      CHKFIN          ; wait
1D98| 6556          BCS.S  TWGOUT      ; exit if timeout
1D9A| 1028 0010      MOVE.B  STAT(A0),D0      ; get disk return code

```



```

1D9E| 117C 00CC 0002          MOVE.B  #$CC,CMD(A0)      ;clear RWTS interrupt bits
1DA4| 10BC 0085          MOVE.B  #CLRSTAT,(A0)
1DA8| 615A          BSR.S    CMDCHK          ;wait until cmd taken
1DAA| 6544          BCS.S    TWGOUT          ;exit if error
1DAC| 4A00          TST.B    D0              ;check status code
1DAE| 6642          BNE.S    TWGERR          ; and exit if error
1DB0|
1DB0|          ; Read successful - transfer header and then data to main memory
1DB0|
1DB0|          .IF NEWTWIG = 0
1DB0|          .ELSE
1DB0| 49E8 03E8          LEA      DSKBUFF(A0),A4      ;set address for Twiggy buffer
1DB4| 074C 0000      XFRHDR  MOVEP.L (A4),D3          ;load header bytes
1DB8| 22C3          MOVE.L  D3,(A1)+
1DBA| 074C 0008          MOVEP.L 8(A4),D3
1DBE| 22C3          MOVE.L  D3,(A1)+
1DC0| 074C 0010          MOVEP.L 16(A4),D3
1DC4| 22C3          MOVE.L  D3,(A1)+
1DC6|
1DC6| 49E8 0400          LEA      DSKDATA(A0),A4      ;set address for data
1DCA| 303C 001F          MOVE.W  #31,D0          ;need to load 512 bytes
1DCE| 074C 0000      XFRDATA MOVEP.L (A4),D3          ;load data bytes
1DD2| 24C3          MOVE.L  D3,(A2)+
1DD4| 074C 0008          MOVEP.L 8(A4),D3
1DD8| 24C3          MOVE.L  D3,(A2)+
1DDA| 074C 0010          MOVEP.L 16(A4),D3
1DDE| 24C3          MOVE.L  D3,(A2)+
1DE0| 074C 0018          MOVEP.L 24(A4),D3
1DE4| 24C3          MOVE.L  D3,(A2)+
1DE6| D8FC 0020          ADD.W  #32,A4
1DEA| 51C8 FFE2          DBF     D0,XFRDATA          ;loop until done
1DEE| 600A          BRA.S    TWGOK          ;and go to exit
1DF0|
1DF0|          .ENDC
1DF0|
1DF0|          ; Error exit - set carry bit as error indicator
1DF0|
1DF0| 7027          TWGOUT  MOVEQ   #TIMOUT,D0          ; set timeout error
1DF2|          TWGERR  ENABLE          ; restore interrupt mask
1DF2| 46DF          #      MOVE     (SP)+,SR
1DF4| 003C 0001          ORI.B   #$01,CCR
1DF8| 6004          BRA.S    TWGRXIT          ; and exit
1DFA|
1DFA|          TWGOK   ENABLE          ; restore interrupt mask
1DFA| 46DF          #      MOVE     (SP)+,SR
1DFC| 4280          CLR.L    D0              ; set OK return code
1DFE|

```

CHG025

```

1DFE| 4CDF 1E08      TWGRXIT MOVEM.L (SP)+,D3/A1-A4 ; restore regs
1E02| 4E75          RTS ; and return to caller
1E04|
1E04| ;-----
1E04| ; Subroutine to check for disk command taken. Also does check for DSKDIAG
1E04| ; in case Twiggy controller becomes busy servicing second disk drive before
1E04| ; command is seen. Loop takes about 12.4 us if DSKDIAG OK, else DSKDIAG
1E04| ; loop takes about 9.6 us.
1E04| ;
1E04| ; Destroys register A0
1E04| ;-----
1E04|
1E04| 48E7 1010      CMDCHK MOVEM.L D3/A3,-(SP) ;save regs
1E08| 263C 0012 0000 MOVE.L #CMDTIME,D3 ;set timeout for about 15 secs
1E0E| 207C 00FC C001 MOVE.L #DISKMEM,A0 ;set ptr to shared memory
1E14| 267C 00FC D901 MOVE.L #VIA2BASE,A3 ;also set up to monitor DSKDIAG
1E1A| 022B 00BF 0010 ANDI.B #$BF,DDRB2(A3)
1E20| 4A10          @1 TST.B (A0) ;cmd taken when byte 0'ed
1E22| 6714          BEQ.S @2
1E24| 0813 0006      @3 BTST #DSKDIAG,IRB2(A3) ;check if controller not ready
1E28| 6606          BNE.S @4 ;skip if OK
1E2A| 5383          SUBQ.L #1,D3 ;else loop until timeout or ready
1E2C| 66F6          BNE.S @3
1E2E| 6004          BRA.S @5 ;take error exit
1E30| 5383          @4 SUBQ.L #1,D3
1E32| 66EC          BNE.S @1 ;loop until yes or timeout
1E34| 003C 0001      @5 ORI.B #$01,CCR ;set error
1E38| 4CDF 0808      @2 MOVEM.L (SP)+,D3/A3 ;restore regs
1E3C| 4E75          RTS
1E3E|
1E3E| ;-----
1E3E| ; Subroutine to check for disk interrupt (FDIR asserted) - loop takes 8.8us
1E3E| ; Destroys register D3 and A3
1E3E| ;-----
1E3E|
1E3E| CHKFIN
1E3E| .IF NEWTWIG = 0
1E3E| .ENDC
1E3E| .IF NEWTWIG = 1
1E3E| 2602 MOVE.L D2,D3 ;set user-supplied timeout
1E40| .ENDC
1E40| 267C 00FC DD81 MOVE.L #VIA1BASE,A3 ;set ptr for interface
1E46| 0813 0004      @1 BTST #4,(A3) ;FDIR?
1E4A| 6608          BNE.S @2 ;exit if yes
1E4C| 5383          SUBQ.L #1,D3
1E4E| 66F6          BNE.S @1 ;else loop
1E50| 003C 0001      ORI.B #$01,CCR ;set error

```



```

1E54| 4E75          @2      RTS
1E56|
1E56|          .PAGE
1E56|          ;-----
1E56|          ; Subroutine to eject disk
1E56|          ; Assumes A0 = ptr to disk shared memory
1E56|          ;-----
1E56|
1E56|      EJCTDSK
1E56|          .IF      NEWTWIG = 1
1E56|      611A      BSR.S      CLRFDIR      ;ensure interrupts cleared
1E58|      6516      BCS.S      @1          ;exit if error
1E5A|      243C 0018 0000      MOVE.L      #EJCTTIME,D2      ;set eject timeout
1E60|          .ENDC
1E60|      117C 0002 0002      MOVE.B      #UNCLAMP,CMD(A0) ;set up cmd
1E66|      10BC 0081      MOVE.B      #EXRW,(A0)      ;go do it
1E6A|      61D2      BSR.S      CHKFIN      ;wait for FDIR
1E6C|      6502      BCS.S      @1          ;skip if error
1E6E|      6102      BSR.S      CLRFDIR      ;clear intrpt and return
1E70|      4E75          @1      RTS
1E72|
1E72|          ;-----
1E72|          ; Subroutine to clear interrupt. Waits for FDIR to go low before return.
1E72|          ; Assumes A0 = ptr to disk shared memory.
1E72|          ;-----
1E72|
1E72|      117C 00FF 0002      CLRFDIR MOVE.B      #$FF,CMD(A0)      ;clear FDIR
1E78|      10BC 0085      MOVE.B      #CLRSTAT,(A0)
1E7C|      6186      BSR.S      CMDCHK      ;wait until cmd taken
1E7E|      267C 00FC DD81      MOVE.L      #VIA1BASE,A3      ;then wait for FDIR to go low
1E84|      7619      MOVEQ      #25,D3      ;set timeout for about 200 us
1E86|      0813 0004      @1      BTST      #FDIR,(A3)      ;FDIR?
1E8A|      6708      BEQ.S      @2          ;skip if none
1E8C|      5343      SUBQ      #1,D3      ;else loop until gone or timeout
1E8E|      66F6      BNE.S      @1
1E90|      003C 0001      ORI.B      #01,CCR      ;set error indicator
1E94|      4E75          @2      RTS
1E96|
1E96|          ;-----
1E96|          ; Subroutine to ensure FDIR gone after clear status cmd
1E96|          ;-----
1E96|
1E96|      267C 00FC DD81      CHKFDIR MOVE.L      #VIA1BASE,A3      ;set ptr for FDIR status
1E9C|      0813 0004      BTST      #FDIR,(A3)      ;FDIR present?
1EA0|      6702      BEQ.S      @1          ;skip if not
1EA2|      61CE      BSR.S      CLRFDIR      ;else do clear
1EA4|      4E75          @1      RTS      ;and exit

```



```

1EA6|
1EA6|          .IF USERINT = 0
1EA6|          .ENDC                                ;{USERINT}
1EA6|
1EA6|          .IF USERINT = 1
1EA6|          ;-----
1EA6|          ; Subroutine to enable display of wait icon. Main entry point creates
1EA6|          ; alert box also, secondary entry point (DSPWTICON) invokes icon display
1EA6|          ; only.
1EA6|          ; Inputs:
1EA6|          ;     None
1EA6|          ; Outputs:
1EA6|          ;     None
1EA6|          ; Side Effects:
1EA6|          ;     A2,A3 trashed
1EA6|          ;-----
1EA6|
1EA6|          WAITALRT
1EA6|          DSPWTICON
1EA6| 45FA 1B95          LEA     WAITICON,A2          ;and display wait icon
1EAA| 6100 1680          BSR     DSPALRTICON
1EAE| 4E75             RTS
1EB0|          .ENDC                                ;{USERINT}
1EB0|
1EB0|          .ENDC                                ;{TWIGGY}
1EB0|
1EB0|          ;-----
1EB0|          ; Routine to reinit vectors before release of control to boot loader.
1EB0|          ; Sets all vectors other than reset and interrupts to jump to the
1EB0|          ; failing boot device handler.
1EB0|          ;
1EB0|          ; Inputs:
1EB0|          ;     A3 = address of boot error handler for bus/address errors
1EB0|          ;     A4 = address of boot error handler for other exceptions
1EB0|          ; Outputs:
1EB0|          ;     None
1EB0|          ; Side Effects:
1EB0|          ;     A0/D1 trashed
1EB0|          ;-----
1EB0|
1EB0|          VCTRINIT
1EB0| 307C 0008          MOVEA  #BUSVCTR,A0          ;get ptr to vector locations          RM000
1EB4| 20CB             MOVE.L A3,(A0)+          ;set up for bus error
1EB6| 20CB             MOVE.L A3,(A0)+          ;and address error
1EB8| 7214             MOVEQ  #20,D1             ;# of remaining low vectors to init
1EBA| 20CC             @1    MOVE.L A4,(A0)+          ;setup handler for errors up
1EBC| 5341             SUBQ   #1,D1             ; to spurious intrpt vector

```



```

1EBE| 66FA          BNE.S    @1
1EC0| 307C 0080      MOVEA    #TRPVCT0,A0      ;next do all trap vectors      RM000
1EC4| 7220          MOVEQ     #32,D1      ;set count
1EC6| 20CC          @2      MOVE.L    A4,(A0)+      ;and do init
1EC8| 5341          SUBQ      #1,D1
1ECA| 66FA          BNE.S    @2
1ECC| 4E75          RTS
1ECE|
1ECE|          .IF      EXTERNAL = 1
1ECE|          .ENDC
1ECE|          .PAGE
1ECE|          .IF PROFILE = 1      ;ASSEMBLE ONLY IF PROFILE CODE NEEDED
1ECE|          ;-----
1ECE|          ; Routine to boot from Profile hard disk attached to parallel port
1ECE|          ; Sets up input parameters and then calls READ routine. If no error
1ECE|          ; on return (carry not set), a jump to the loaded boot code is done.
1ECE|          ;-----
1ECE|
1ECE|          PROBOOT
1ECE|          .IF USERINT = 0
1ECE|          .ELSE
1ECE| 61D6          BSR      WAITALRT      ; display wait icon
1ED0|          .ENDC
1ED0|
1ED0| 227C 0001 FFEC      MOVE.L    #HDRBUFR,A1      ; set up ptr to save header
1ED6| 247C 0002 0000      MOVE.L    #HDRBUFR+20,A2      ; ptr for data
1EDC| 4281          CLR.L     D1      ; set sector #
1EDE| 243C 0120 0000      MOVE.L    #STRTIME,D2      ; set timeout count (3 mins)
1EE4| 760A          MOVEQ     #RCNT,D3      ; set retry count
1EE6| 7803          MOVEQ     #TCNT,D4      ; set threshold count
1EE8| 6100 0086          BSR      PROREAD      ; go get data
1EEC| 6522          BCS.S    HDSKERR      ; exit if error
1EEE|
1EEE|          ; Now verify header and if OK, jump to startup program
1EEE|
1EEE| 3029 0004          MOVE     FILEID(A1),D0      ; get file id
1EF2| 0C40 AAAA          CMP      #BOOTPAT,D0      ; is it a boot block?
1EF6| 6704          BEQ.S     PBOOT      ; continue if OK
1EF8| 7054          MOVEQ     #BADHDR,D0      ; else set error code
1EFA| 6014          BRA.S    HDSKERR      ; and exit
1EFC|
1EFC| 47FA 0060          PBOOT  LEA      HDERR2,A3      ;set up vectors in case of errors
1F04| 61AA          BSR      VCTRINIT4
1F06|
1F06| 247C 0002 0000      MOVE.L    #HDRBUFR+20,A2      ; set ptr for data
1F0C| 6000 FD74          BRA      STRTBOOT      ; and go start execution
1F10|

```



```

1F10|                ; Error detected - output message and abort boot
1F10|
1F10|                HDSKERR
1F10|                .IF USERINT = 0
1F10|                .ELSE
1F10|
1F10| 0C38 0003 02AF          CMP.B  #3,SYSTYPE          ;check system type          CHG009/CHG029
1F16| 6706                  BEQ.S  @2                  ;skip if internal disk          CHG009/CHG029
1F18| 45FA 1B85              LEA     PROICON,A2          ;else setup Profile icon
1F1C| 6004                  BRA.S  @3                  ;skip to do display          CHG009
1F1E| 45FA 1BBA              @2    LEA     UPPER,A2          ;set for integral hard disk    CHG009
1F22|
1F22| 0C00 0050              @3    CMP.B  #NODSK,D0          ;Profile not attached error?
1F26| 660E                  BNE.S  @1                  ;skip if not
1F28|
1F28|                ; If default boot and no Profile attached, go to boot menu
1F28|
1F28| 0807 001C              BTST     #ALTBOT,D7          ;is this a default boot?
1F2C| 6608                  BNE.S  @1                  ;skip if not
1F2E| 6100 11AA              BSR     CLRDESK          ;clear desktop
1F32| 6000 F8FA              BRA     LSTCHK          ;and do beep and display the boot menu
1F36|
1F36| 6100 15A2              @1    BSR     DSPERRICON          ;display with bad mark
1F3A|
1F3A|                .ENDC                                ;{PROFILE}
1F3A|
1F3A|                ; Sound error tones and display error code
1F3A|
1F3A|                BOOTFAIL
1F3A| 6100 F6E6              BSR     DSPCODE          ;display error code
1F3E|
1F3E|                BFAIL2
1F3E| 6100 F788              BSR     HIPTCH          ;startup failure causes hi,hi,hi tones
1F42| 6100 F784              BSR     HIPTCH
1F46| 6100 F780              BSR     HIPTCH
1F4A| 6100 11C8              BSR     CLRMENU          ;clear menu bar
1F4E| 0238 00FC 02A2        ANDI.B  #$FC,STATFLGS          ;allow CONTINUE option from boot error
1F54| 08F8 0000 02A2        BSET     #NORSTRT,STATFLGS          ; but not restart option
1F5A| 6000 0640              BRA     MONITOR          ;and go to monitor
1F5E|
1F5E|                HDERR2 BSR4    SAVEXCP          ;save exception info
1F5E| 49FA 0006              #      LEA     @1,A4
1F62| 6000 FDB8              #      BRA     SAVEXCP
1F66|                #@1
1F66|                HDERR3 BSR4    BTERR          ;regroup from error
1F66| 49FA 0006              #      LEA     @1,A4
1F6A| 6000 FDBE              #      BRA     BTERR

```



```

1F6E|      #c1
1F6E| 60A0      BRA.S   HDSKERR           ;and go display it
1F70|
1F70|      .IF      EXTERNAL = 1
1F70|      .ENDC
1F70|
1F70|      .PAGE
1F70|      ;-----
1F70|      ; First initialize and then ensure disk is attached by checking OCD line.
1F70|      ; Assumes ACR and IER registers of VIA set up by caller. For boot, these
1F70|      ; are cleared by power-up reset.
1F70|      ; Register usage:
1F70|      ;   D0 = scratch use           A0 = VIA address for parallel port interface
1F70|      ;   D1 = block to read        A1 = address to save header
1F70|      ;   D2 = timeout count        A2 = address to save data
1F70|      ;   D3 = retry count          A3 = scratch
1F70|      ;   D4 = threshold count      A4 = unused
1F70|      ; Returns:
1F70|      ;   D0 = error code (0 = OK)
1F70|      ;   D1 = error bytes (4)
1F70|      ;   D2 - D7 and A1 - A6 are preserved
1F70|      ;-----
1F70| 48E7 3F7E      PROREAD MOVEM.L D2-D7/A1-A6,-(SP)      ; save regs
1F74|      DISABLE                                ; ensure interrupts off
1F74| 40E7      #      MOVE      SR,-(SP)
1F76| 007C 0700      #      ORI      #$0700,SR
1F7A| 6174      BSR.S   PROINIT                    ; init for Profile access and check
1F7C|      ; if attached
1F7C| 6704      BEQ.S   CHKBSY                      ; go on if OK
1F7E| 7050      MOVEQ   #NODSK,D0                  ; else set error code and
1F80| 605C      BRA.S   PROERR                      ; skip if error - no disk attached
1F82|
1F82|      ; Now check if Profile ready - wait time presently set for about 100 seconds
1F82|      ; to allow enough time for normal Profile startup time of about 80 seconds
1F82|
1F82| 0810 0001      CHKBSY BTST   #BSY,IRB2(A0)          ; check if Profile ready (not busy)
1F86| 6608      BNE.S   TRYRD                      ; skip if yes
1F88| 5382      SUBQ.L   #1,D2                      ; else loop until timeout
1F8A| 66F6      BNE.S   CHKBSY
1F8C|
1F8C| 7051      MOVEQ   #DSKBSY,D0                  ; set disk busy error code
1F8E| 604E      BRA.S   PROERR                      ; and go to error exit
1F90|
1F90|      ; Now start read and check status to see if OK
1F90|
1F90| 6100 00B6      TRYRD  BSR      STRTRD           ; go begin read process

```



```

1F94| 6418          BCC.S  @1          ; skip if OK          CHG016
1F96| 6100 018A     BSR     WFNBSY     ; else check for ready CHG016
1F9A| 6100 00AC     BSR     STRTRD     ; and do retry       CHG016
1F9E| 640E          BCC.S  @1          ; continue if OK     CHG016
1FA0| 6100 01A4     BSR     DOCRES     ; else issue reset signal CHG016
1FA4| 6100 017C     BSR     WFNBSY     ; wait until ready    CHG016
1FA8| 6100 009E     BSR     STRTRD     ; and try one more time CHG016
1FAC| 6530          BCS.S  PROERR      ; finally exit if error
1FAE|
1FAE| 4A78 01B6     @1    TST.W  STAT3      ; check if reset error CHG016
1FB2| 6A06          BPL.S  @2          ; skip if not
1FB4| 6100 0092     BSR     STRTRD     ; else go try read again
1FB8| 6524          BCS.S  PROERR      ; and abort if error
1FBA|
1FBA| 4AB8 01B4     @2    TST.L  STATBFR    ; check complete status
1FBE| 6712          BEQ.S  RDDATA      ; skip if OK
1FC0| 2238 01B4     MOVE.L  STATBFR,D1    ; else get status
1FC4| 2001          MOVE.L  D1,D0        ; save for use
1FC6| 0280 C140 C000 ANDI.L  #STATMSK,D0 ; mask don't care bits
1FCC| 6704          BEQ.S  RDDATA      ; and continue if OK
1FCE| 7053          MOVEQ   #STATNZ,D0   ; else set error code
1FD0| 600C          BRA.S  PROERR      ; and go to error exit
1FD2|
1FD2|              ; All OK - go read block and transfer to memory; do as multiple moves for
1FD2|              ; max transfer rate.
1FD2|
1FD2| 7004          RDDATA MOVEQ   #<HDRSIZE/4>-1,D0 ; set count for header read
1FD4| 615C          BSR.S  READIT      ; go do it
1FD6|
1FD6| 707F          MOVEQ   #<BLKSIZE/4>-1,D0 ; set count for data read
1FD8| 224A          MOVEA.L A2,A1      ; set new read location
1FDA| 6156          BSR.S  READIT
1FDC|
1FDC| 6008          BRA.S  PROXIT      ; and go to exit
1FDE|
1FDE|              ; Error exit - set carry bit as indicator flag
1FDE|
1FDE| 46DF          #      MOVE     (SP)+,SR ;restore interrupt mask
1FE0| 003C 0001     ORI.B  #$01,CCR
1FE4| 6004          BRA.S  PROXIT2
1FE6|
1FE6|              ; Normal exit - restore regs and exit
1FE6|  PROXIT  ENABLE ;restore interrupt mask
1FE6| 46DF          #      MOVE     (SP)+,SR
1FE8| 4280          CLR.L  D0          ;set OK return code          CHG025
1FEA|
1FEA| 4CDF 7EFC     PROXIT2 MOVEM.L (SP)+,D2-D7/A1-A6

```



```

1FEE| 4E75                RTS
1FF0|
1FF0|                    ;-----
1FF0|                    ; Subroutine to init parallel port for Profile access.
1FF0|                    ; Inputs:
1FF0|                    ;     None
1FF0|                    ; Outputs:
1FF0|                    ;     D0 cleared for error use
1FF0|                    ;     A0 = VIA base address for parallel port
1FF0|                    ;     CCR zero bit set if cable connected
1FF0|                    ; Side Effects:
1FF0|                    ;     None
1FF0|                    ;-----
1FF0|
1FF0|                PROINIT
1FF0| 4280                CLR.L    D0                ; clear for result use
1FF2| 267C 00FC DD81    MOVE.L    #VIA1BASE,A3        ; get kybd VIA base address      CHG036
1FF8| 0013 00A0                ORI.B    #$A0,ORB1(A3)    ; initialize profile-reset and parity-reset  CHG036
1FFC| 002B 00A0 0004                ORI.B    #$A0,DDRB1(A3)    ; and set lines as outputs          CHG036
2002| 207C 00FC D901    MOVE.L    #VIA2BASE,A0        ; get paraport VIA base address
2008| 0228 007B 0060                ANDI.B    #$7B,PCR2(A0)    ; set ctrl CA2 pulse mode/positive edge
200E| 0028 006B 0060                ORI.B    #$6B,PCR2(A0)
2014| 4228 0018                MOVE.B    #$00,DDRA2(A0)        ; set port A bits to input
2018| 0010 0018                ORI.B    #$18,ORB2(A0)        ; then set direction=in, cmd=false,      CHG036
201C| 0210 00FB                ANDI.B    #$FB,ORB2(A0)        ; enable=true                          CHG036
2020| 0228 00FC 0010                ANDI.B    #$FC,DDRB2(A0)        ; set port B bits 0,1=in,
2026| 0028 001C 0010                ORI.B    #$1C,DDRB2(A0)        ; 2,3,4=out
202C| 0810 0000                BTST     #OCD,IRB2(A0)        ; check OCD line
2030| 4E75                RTS                ; and exit
2032|
2032|                    ;-----
2032|                    ; Subroutine to read bytes from Profile. Assumes:
2032|                    ;     D0 = byte count - 1
2032|                    ;     A0 = VIA address for parallel interface
2032|                    ;     A1 = memory load address
2032|                    ;-----
2032|
2032| 12E8 0008                READIT  MOVE.B    IRA2(A0),(A1)+        ; read the bytes
2036| 12E8 0008                MOVE.B    IRA2(A0),(A1)+
203A| 12E8 0008                MOVE.B    IRA2(A0),(A1)+
203E| 12E8 0008                MOVE.B    IRA2(A0),(A1)+
2042| 51C8 FFEE                DBF      D0,READIT
2046| 4E75                RTS
2048|
2048|                .PAGE
2048|                    ;-----
2048|                    ; Subroutine to begin the read process. First calls a routine that

```



```

2048|          ; an appropriate response from Profile is executed. Then a wait for
2048|          ; Assumes regs:
2048|          ;   D0 = scratch use          A0 = VIA address
2048|          ;   D1 = block to read       A1 = unused
2048|          ;   D2 = used for Profile cmd A2 = unused
2048|          ;   D3 = retry count          A3 = scratch use
2048|          ;   D4 = threshold count      A4 = unused
2048|          ; If error, carry bit set and error code in D0.
2048|          ;-----
2048|
2048| 367C 0304      STRTRD MOVEA    #CMDBUFR,A3          ; first set up command          RM000
204C| 2681          MOVE.L    D1,(A3)                  ; set read command (0) and block #
204E| 1743 0004      MOVE.B    D3,RETRY(A3)            ; set retry count
2052| 1744 0005      MOVE.B    D4,THRESH(A3)           ; set threshold for sparing
2056| 611E          BSR.S      STAT01                   ; get 01 byte and send read command
2058| 651A          BCS.S      STRTXIT                   ; exit if error
205A|
205A|          ; OK so far - go check if Profile ready to send data
205A|
205A| 7402          MOVEQ     #2,D2                      ; get 02 byte          RM000
205C| 615E          BSR.S      FINDD2
205E| 6514          BCS.S      STRTXIT                   ; exit if timeout error
2060|
2060|          ; Get status bytes
2060| 367C 01B4      GETSTAT MOVEA    #STATBFR,A3          ; set buffer ptr          RM000
2064| 16E8 0008      MOVE.B    IRA2(A0),(A3)+            ; read and save the status
2068| 16E8 0008      MOVE.B    IRA2(A0),(A3)+
206C| 16E8 0008      MOVE.B    IRA2(A0),(A3)+
2070| 16E8 0008      MOVE.B    IRA2(A0),(A3)+
2074|
2074| 4E75          STRTXIT RTS                          ; return to caller
2076|
2076|          .PAGE
2076|          ;-----
2076|          ; Subroutine to get in sync with Profile.
2076|          ; Input regs:
2076|          ;   A0 = VIA address
2076|          ;   A3 = ptr to command buffer
2076|          ; If error, returns with carry bit set and error code in D0
2076|          ;-----
2076|
2076| 48E7 2800      STAT01 MOVEM.L D2/D4,-(SP)           ; save regs
207A| 7401          MOVE.L    #1,D2                      ; try to find state 01
207C| 613E          BSR.S      FINDD2
207E| 6412          BCC.S      COPY6                     ; skip if OK
2080| 0C00 0055      CMP.B     #TMOUT,D0                 ; else check if timeout error
2084| 672C          BEQ.S      STATERR                    ; and exit if yes

```



```

2086|
2086| 6100 00AA          BSR      WFNBSY3          ; ensure Profile ready          RM000
208A| 4A00          TST.B    D0                  ; check for timeout error
208C| 6624          BNE.S    STATERR              ; and exit if yes
208E|
208E| 612C          @2      BSR.S    FINDD2          ; try to find state 01 again
2090| 6524          BCS.S    STATXIT              ; exit if error again
2092|
2092| 0210 00F7          COPY6  ANDI.B  #$F7,ORB2 (A0)      ; set dir=out
2096| 117C 00FF 0018          MOVE.B  #$FF,DDRA2 (A0)      ; set port A bits to output
209C| 303C 0005          MOVE.W  #PCMSZ,D0              ; set command size
20A0| 115B 0008          COPY6LP MOVE.B  (A3)+,ORA2 (A0)    ; send the command bytes
20A4| 51C8 FFFA          DBF      D0,COPY6LP
20A8| 0010 0008          ORI.B   #$08,ORB2 (A0)          ; reset dir=in
20AC| 4228 0018          MOVE.B  #$00,DDRA2 (A0)          ; and set port A bits to input
20B0| 6004          BRA.S    STATXIT              ; then exit
20B2|
20B2| 003C 0001          STATERR ORI.B  #$01,CCR          ; set error indicator
20B6|
20B6| 4CDF 0014          STATXIT MOVEM.L (SP)+,D2/D4        ; restore regs
20BA| 4E75          RTS
20BC|
20BC|          .PAGE
20BC|
20BC|          ;-----
20BC|          ; Subroutine to handshake with Profile and wait for command completion.
20BC|          ; Polls busy bit until it goes low (not busy).
20BC|          ; Assumes regs:
20BC|          ;   A0 = VIA address
20BC|          ;   D2 = Expected response to previously issued command
20BC|          ;   If error, carry bit set and error code in D0.
20BC|          ;-----
20BC|
20BC| 48E7 7800          FINDD2  MOVEM.L D1-D4,-(SP)          ; save regs
20C0| 0210 00EF          ANDI.B  #$EF,ORB2 (A0)          ; set cmd=true
20C4| 4228 0018          MOVE.B  #$00,DDRA2 (A0)          ; set port A bits to input
20C8| 4280          CLR.L    D0                  ; used for return code
20CA| 6130          BSR.S    WFBSY              ; wait for busy
20CC| 6618          BNE.S    FINDERR              ; exit if error
20CE|
20CE| 1228 0078          GETRSP  MOVE.B  PORTA2 (A0),D1      ; get response in D1 w/o handshake
20D2| 4203          CLR.B    D3                  ; used for reply to Profile
20D4| B202          CMP.B    D2,D1              ; did pippin return state requested ?
20D6| 6704          BEQ.S    RSPOK              ; skip if yes
20D8| 7052          MOVEQ    #BADRSP,D0          ; else set error code
20DA| 6002          BRA.S    SNDR1              ; and go send reply
20DC|
20DC| 7655          RSPOK    MOVEQ    #$55,D3          ; set up OK reply          RM000

```



```

20DE|
20DE| 612E          SNDRI  BSR.S  SENDRSP          ; send response
20E0|
20E0| 4A00          TST.B  D0          ; check return code
20E2| 6602          BNE.S  FINDERR        ; skip if error
20E4|
20E4| 613C          BSR.S  WFNBSY        ; now go wait for not busy
20E6|
20E6| 4228 0018      FINDERR MOVE.B  #$00,DDRA2(A0)    ; reset port A bits to input
20EA| 0010 0018      ORI.B  #$18,ORB2(A0)    ; and dir = in, cmd=false
20EE|
20EE| 4A00          TST.B  D0          ; check return code
20F0| 6704          BEQ.S  FNDXIT        ; skip if OK
20F2| 003C 0001      ORI.B  #$01,CCR        ; else set error indicator
20F6|
20F6| 4CDF 001E      FNDXIT  MOVEM.L (SP)+,D1-D4    ; restore regs (but don't affect CCR bits)
20FA| 4E75          RTS
20FC|
20FC|
20FC| ;-----
20FC| ; Subroutine to wait for Profile busy signal. Polls busy bit until it
20FC| ; goes high (busy).
20FC| ; Assumes regs:
20FC| ;   A0 = VIA address
20FC| ;   D4 = timeout value if WFBSY1 entry point used
20FC| ;   If error, error code in D0.
20FC| ;-----
20FC|
20FC| 383C FFFF      WFBSY  MOVE  #RSPTIME,D4          ; set response timeout = 100 msec
2100|
2100| 0810 0001      WFBSY1 BTST  #BSY,IRB2(A0)        ; wait for busy
2104| 6706          BEQ.S  @9          ; skip if OK
2106| 5344          SUBQ  #1,D4          ; else loop until timeout
2108| 66F6          BNE.S  WFBSY1
210A| 7055          MOVEQ  #TMOUT,D0          ; set timeout error
210C| 4E75          @9    RTS
210E|
210E| ;-----
210E| ; Subroutine to send response command to Profile.
210E| ; Assumes regs:
210E| ;   A0 = VIA address
210E| ;-----
210E|
210E| 0210 00E7      SENDRSP ANDI.B  #$E7,ORB2(A0)    ; set dir=out, cmd=true
2112| 117C 00FF 0018      MOVE.B  #$FF,DDRA2(A0)    ; set port A bits to output
2118| 1143 0078      MOVE.B  D3,PORTA2(A0)    ; send reply(00 or 55) w/o handshake
211C| 0010 0010      ORI.B  #$10,ORB2(A0)    ; set cmd=false

```

```

2120| 4E75                      RTS
2122|
2122| ;-----
2122| ; Subroutine to wait for Profile not busy signal.  Polls busy bit until it
2122| ; goes low (not busy).
2122| ; Assumes regs:
2122| ;   A0 = VIA address
2122| ;   D4 = timeout value if WFNBSY1 entry point used
2122| ;   If error, D0 has error code.
2122| ;-----
2122|
2122| 283C 0018 0000      WFNBSY  MOVE.L  #RDTIME,D4          ; set timeout for about 16 secs      CHG037
2128| 600E                BRA.S    WFNBSY1                    ;                          CHG019
212A|
212A| 283C 0120 0000      WFNBSY2 MOVE.L  #STRTIME,D4         ; set initial Profile self-test time  CHG019
2130| 6006                BRA.S    WFNBSY1                    ;                          CHG019
2132|
2132| 283C 0000 0500      WFNBSY3 MOVE.L  #BSYTIME,D4         ; set timeout for about 10 ms        RM000
2138|
2138| 0810 0001           WFNBSY1 BTST    #BSY,IRB2(A0)        ; wait for not busy
213C| 6606                BNE.S    @9                          ; exit if OK
213E| 5384                SUBQ.L   #1,D4                      ; else loop until timeout
2140| 66F6                BNE.S    WFNBSY1
2142| 7055                MOVEQ    #TMOUT,D0                  ; set timeout error
2144| 4E75                @9      RTS
2146|
2146| ;-----
2146| ; Subroutine to send reset to Profile controller to enable handshake retry  CHG016
2146| ;-----
2146|
2146| 267C 00FC DD81      DOGRES  MOVE.L  #VIA1BASE,A3         ;use keyboard 6522      CHG026
214C| 0213 007F          ANDI.B   #$7F,ORB1(A3)              ;set reset signal      CHG016/CHG026/CHG036
2150| 6100 E97A          BSR      DELAY_1                     ;delay for controller to get signal  CHG016
2154| 0013 0080          ORI.B    #$80,ORB1(A3)              ;remove reset signal    CHG016/CHG026/CHG036
2158| 6100 E972          BSR      DELAY_1                     ;delay for controller to respond    CHG038
215C| 4E75              RTS                                    ;and exit
215E|
215E|                      .ENDC                               ;{PROFILE}
215E|                      .IF      EXTERNAL = 1
215E|                      .ENDC
215E|                      .PAGE
215E| ;-----
215E| ; Routine to boot from I/O slot.
215E| ; Verifies that slot has bootable card installed and then reads in ROM
215E| ; data.  If status routine exists it is executed, else jump to boot
215E| ; routine done if checksum OK.
215E| ;

```



```

215E|      ; Inputs:
215E|      ;      D0 = boot device id ($4-$C)
215E|      ;      A0 = scratch use
215E|      ;      A1 = slot address
215E|      ; Outputs: (relayed to loaded boot program)
215E|      ;      D0 = boot device id
215E|      ;      A1 = slot address
215E|      ;-----
215E|
215E| 1800      IOSBOOT MOVE.B  D0,D4      ;save boot device id
2160|          ; also acts as flag for status check
2160| 2A78 0008      MOVE.L  BUSVCTR,A5    ;save bus error vector
2164| 2C4F          MOVE.L  SP,A6        ;save current stack pointer
2166|
2166| 47FA 002C      LEA     NOCRD,A3      ;setup new bus error vector
216A| 21CB 0008      MOVE.L  A3,BUSVCTR
216E| 0309 0000      MOVEP   (A1),D1      ;read card id
2172|
2172|          .IF  USERINT = 0
2172|          .ENDC
2172|
2172| 4A41          TST      D1            ;check if card installed
2174| 6A28          BPL.S    INVID        ;exit if not bootable
2176| 0C41 FFFF      CMP     #$FFFF,D1    ;check if special case
217A| 6722          BEQ.S    INVID
217C|
217C|          .IF  USERINT = 1
217C| 6100 FD28      BSR     WAITALRT     ;display wait icon
2180|          .ENDC
2180|
2180| 2449          MOVE.L  A1,A2          ;get slot address
2182| 6148          BSR.S    RDIOSLT      ;go check the board
2184| 651C          BCS.S    BADBRD       ;exit if error
2186|
2186|          STATOK
2186| 1038 01B3      MOVE.B  BOOTDVCE,D0  ;setup boot device id
218A| 247C 0002 0002 MOVE.L  #BTENTRY,A2  ;and starting program address
2190| 6000 FAF0      BRA     STRTBOOT     ;and go do boot ...
2194|
2194|      ; Error routines for I/O slot booting. Error code saved and error message output.
2194| 705A          NOCRD   MOVEQ   #NOC,D0 ;set error code
2196| 21CD 0008      MOVE.L  A5,BUSVCTR    ;restore bus error vector
219A| 2E4E          MOVE.L  A6,SP          ;and stack ptr
219C| 6008          BRA.S    SENDMSG       ;then go display msg
219E|
219E| 705B          INVID   MOVEQ   #INV,D0 ;set error code
21A0| 6004          BRA.S    SENDMSG       ;go do display

```




```

21A2|
21A2| 1838 01B3      BADBRD  MOVE.B  BOOTDVCE,D4      ;restore boot device id for checking
21A6|
21A6| SENDMSG
21A6| 11C0 01B4      MOVE.B  D0,BOOTDATA      ;save error code
21AA|
21AA|      .IF  USERINT = 0
21AA|      .ELSE
21AA|      ;  determine which slot # being used
21AA|
21AA| 45FA 1868      LEA      XCARD,A2      ;set general I/O slot card ptr
21AE| 0C04 0004      CMP.B  #IO1PORT2,D4      ;in slot 1 range?
21B2| 6E04      BGT.S  @1
21B4| 7201      MOVEQ   #1,D1      ;yes - set slot #1
21B6| 600C      BRA.S  @3
21B8|
21B8| 0C04 0007      @1  CMP.B  #IO2PORT2,D4      ;slot 2 range?
21BC| 6E04      BGT.S  @2
21BE| 7202      MOVEQ   #2,D1      ;set slot 2 id
21C0| 6002      BRA.S  @3
21C2|
21C2| 7203      @2  MOVEQ   #3,D1      ;else must be slot 3
21C4|
21C4| 6100 128A      @3  BSR      DSPNUMICON      ;display error icon and slot #
21C8| 6000 FD70      BRA      BOOTFAIL      ;and signal boot failure
21CC|      .ENDC
21CC|
21CC|      .PAGE
21CC|      .IF  NEWTWIG = 1
21CC|
21CC|      ;-----
21CC|      ;  Routine to read ROM on an I/O slot.
21CC|      ;  Inputs:
21CC|      ;      A2 = I/O slot base address
21CC|      ;      D4 = nonzero if status check also to be done, else 0 for no check
21CC|      ;      D1 = card id if status check needed
21CC|      ;  Outputs:
21CC|      ;      Carry bit set if error, error code saved in D0, error code from
21CC|      ;      status check saved in BOOTDATA+1
21CC|      ;  Side Effects:
21CC|      ;      D0, A2 trashed
21CC|      ;-----
21CC| 48E7 70C0      RDIOSLT  MOVEM.L  D1-D3/A0-A1,-(SP) ;save regs
21D0| 207C 0001 FFFC      MOVE.L  #ADR128K-4,A0      ;set load pt (also load id/word count)
21D6| 224A      MOVE.L  A2,A1      ;save slot address for later use
21D8| 4280      CLR.L  D0      ;clear for use
21DA| 010A 0004      MOVEP   4(A2),D0      ;read word count
21DE| 5440      ADDQ    #2,D0      ;incr for id/count fields

```



```

21E0| 0C40 0FFF          CMPI    #$0FFF,D0          ;check for max count
21E4| 6244              BHI.S    INVSUM             ;exit if error
21E6| 4282              CLR.L    D2                 ;clear for use
21E8| 4283              CLR.L    D3
21EA|
21EA| 050A 0000          LOADPGM MOVEP   (A2),D2          ;read word
21EE| 3082              MOVE     D2,(A0)             ;save in memory
21F0| 3418              MOVE     (A0)+,D2           ;reread it from memory
21F2| D642              ADD      D2,D3              ;add to checksum
21F4| E35B              ROL      #1,D3              ;rotate for better effectiveness
21F6| 588A              ADDQ.L   #4,A2              ;bump address ptr
21F8| 5340              SUBQ     #1,D0
21FA| 66EE              BNE.S    LOADPGM            ;loop until done
21FC|
21FC| 050A 0000          MOVEP   (A2),D2          ;read expected checksum (2 bytes)
2200| D642              ADD      D2,D3              ;add to calculated checksum
2202| 4A43              TST      D3                 ;check for 0 result (also clears carry bit)
2204| 6624              BNE.S    INVSUM             ;skip if error
s2206|
2206|                  ; Do status check if desired and available
2206|
2206| 4A44              TST      D4                 ;do status check?
2208| 672A              BEQ.S    RDIOXIT            ;skip if not
220A| 0801 000E          BTST     #STBIT,D1          ;status routine available?
220E| 6724              BEQ.S    RDIOXIT            ;skip if not
2210|
2210| 48E7 0F3E          MOVEM.L  D4-D7/A2-A6,-(SP) ;save regs not already saved
2214| 4EB9 0002 0000      JSR      STENTRY           ;go execute status routine
221A| 4CDF 7CF0          MOVEM.L  (SP)+,D4-D7/A2-A6 ;restore regs
221E|
221E| 4A40              TST      D0                 ;check status
2220| 6712              BEQ.S    RDIOXIT            ;skip if no error
2222| 11C0 01B5          MOVE.B   D0,BOOTDATA+1       ;save card error code
2226| 705D              MOVEQ    #BADST,D0           ;set general error code
2228| 6002              BRA.S    SAVERR
222A|
222A| 705C              INVSUM   MOVEQ    #BADSM,D0          ;set invalid checksum
222C|
222C| 11C0 01B4          SAVERR   MOVE.B   D0,BOOTDATA       ;save error code
2230| 003C 0001          ORI.B    #$01,CCR           ;set error indicator
2234|
2234| 4CDF 030E          RDIOXIT  MOVEM.L  (SP)+,D1-D3/A0-A1 ;restore regs
2238| 4E75              RTS
223A|
223A|                  .ENDC
223A|
223A|                  .IF      BURNIN = 1

```



```

223A|          .PAGE
223A|          ;-----
223A|          ; Special code to enable burnin cycling by the ROM. Does initialization
223A|          ; on first pass, and then causes cycling for the specified cycle count,
223A|          ; which defaults to approximately 60 minutes. At
223A|          ; that point a branch to a system shutdown routine is performed.
223A|          ;-----
223A|          ; Do first pass initialization
223A| 223A| 0C39 0001 00FC C191  CHKPASS CMP.B  #$01,INITFLG  ;first pass done?
2242| 6756                      BEQ.S   CHKTIM      ;skip if yes
2244| 207C 00FC C191      CHKPAS2 MOVEA.L #INITFLG,A0  ;set ptr for other data areas
224A| 227C 00FC C1FF              MOVEA.L #ENDPM,A1    ;and ending ptr
2250|
2250| 4210                      CLRPM   CLR.B   (A0)      ;do clear
2252| 5488                      ADDQ.L  #2,A0          ;bump ptr
2254| B3C8                      CMPA.L  A0,A1          ;loop until done
2256| 66F8                      BNE.S   CLRPM
2258|
2258| 7001                      MOVEQ   #1,D0
225A| 13C0 00FC C191      MOVE.B   D0,INITFLG  ;set init flag
2260| 13FC 003C 00FC C1C3  MOVE.B   #60,CYCLVAL  ;set cycling for 60 minutes
2268| 42B8 01BA                      CLR.L   CLKDATA  ;and init clock data area
226C| 4278 01BE                      CLR     CLKDATA+4
2270|
2270|          ; Set clock to initial value so run can be ended at cycle count.  Sends
2270|          ; value of day=01, all other values=0 (e.g., time = 00:00:00).
2270|
2270| 702C                      MOVEQ   #$2C,D0      ;set up clock set cmd
2272| 6100 E6E2                      BSR    COPSCMD    ;and send to COPS
2276| 651E                      BCS.S   @9          ;exit if error                      RM000
2278| 4281                      MOVE.L  #SET1,D1      ;set up value for clock
227A| 7408                      MOVEQ   #8,D2        ;set "char" count
227C| 6100 01E8                      BSR    TODSET     ;and go do it
2280| 6514                      BCS.S   @9          ;                      RM000
2282| 223C 1000 0000      MOVE.L  #SET2,D1      ;set up next value for clock
2288| 7408                      MOVEQ   #8,D2        ;set "char" count
228A| 6100 01DA                      BSR    TODSET     ;and go do it
228E| 6506                      BCS.S   @9          ;                      RM000
2290| 7025                      MOVEQ   #$25,D0      ;finally set up clock enable cmd
2292| 6100 E6C2                      BSR    COPSCMD    ;and send it
2296| 6500 01B0      @9      BCS    SETERR1      ;exit if error
229A|
229A|          ; Check to see if cycle count to be changed and if time data needs to be saved
229A|
229A|          CHKTIM

```



```

229A|          .IF USERINT = 1
229A| 6100 0EC8      BSR      MAKEPCALRT      ;setup powercycle alert box
229E|          .ENDC
229E|
229E| 0838 0002 02A2      BTST      #MSBUTN,STATFLGS ;mouse button detected?
22A4| 6718          BEQ.S      @3              ;skip if no                      RM000
22A6| 1039 00FC C1C3      MOVE.B      CYCLVAL,D0      ;read current setting          RM000
22AC| 0C00 003C          CMP.B      #60,D0              ;long cycle set?                RM000
22B0| 6604          BNE.S      @1
22B2| 7003          MOVEQ      #3,D0              ;set for 3 minute cycle          RM000
22B4| 6002          BRA.S      @2
22B6| 703C          @1      MOVEQ      #60,D0              ;set for 60 minute cycle        RM000
22B8| 13C0 00FC C1C3      @2      MOVE.B      D0,CYCLVAL      ;save in PM                    RM000
22BE|
22BE| 0C39 0001 00FC C199      @3      CMP.B      #$01,TIMFLG      ;time data saved?            RM000
22C6| 6722          BEQ.S      TWGCHK              ;skip if yes
22C8|
22C8| 2038 01BC          MOVE.L      HOUR,D0              ;get minutes
22CC| E998          ROL.L      #4,D0
22CE| 4840          SWAP      D0
22D0| 13C0 00FC C19B      MOVE.B      D0,MINSAV      ;save minutes
22D6| 4239 00FC C1C5      CLR.B      MINCNT              ;and clear minute count
22DC| 4239 00FC C1C1      CLR.B      CYCLCNT              ; and cycle count
22E2| 13FC 0001 00FC C199      MOVE.B      #$01,TIMFLG      ;and set flag
22EA|
22EA|          ; Check if time for Twiggy test (do every two minutes)
22EA|
22EA| 0C39 0002 00FC C1C5      TWGCHK      CMP.B      #2,MINCNT      ;check minute counter
22F2| 6600 0082          BNE      WRTMSG
22F6|
22F6| 4239 00FC C1C5          CLR.B      MINCNT              ;clear counter
22FC| 47FA 1AEB          LEA      TWGMSG,A3              ;get msg ptr
2300| 6100 13F4          BSR      DSPMSGR              ;and display it
2304| 7C0C          MOVEQ      #PCCOL,D6              ;reset left margin
2306|
2306| 47FA EE7E          LEA      DSKVCT,A3              ;set up bus error vector
230A| 21CB 0008          MOVE.L      A3,BUSVCTR
230E| 207C 00FC C001          MOVE.L      #DISKMEM,A0      ;set ptr to shared memory
2314| 267C 00FC DD81          MOVE.L      #VIA1BASE,A3
231A| 4A38 02AF          TST.B      SYSTYPE              ;check system type          CHG009
231E| 6704          BEQ.S      @1              ;skip if Lisa 1.0          CHG009
2320| 7850          MOVEQ      #80,D4              ;else set track count for SONY drive CHG009
2322| 6012          BRA.S      @2              ;and go test single drive    CHG009
2324|
2324| 4281          @1      CLR.L      D1              ;else set for drive 1, track 0 to start
2326| 782D          MOVEQ      #45,D4              ;set count (# of tracks)
2328|

```



```

2328|                ; Now do the drive test, one drive at a time
2328|
2328| 6100 FB48                BSR      CLRFDIR      ;first clear interrupts
232C| 6516                BCS.S   TSTERR        ;exit if error
232E| 6100 014E            BSR      TWGTST        ;go do test
2332| 6510                BCS.S   TSTERR
2334| 782D                MOVEQ    #45,D4          ;reset track count          CHG009
2336|
2336| 4281                @2      CLR.L   D1          ;set for drive 2          CHG009
2338| 123C 0008            MOVE.B   #$08,D1
233C| E899                ROR.L    #4,D1
233E| 6100 013E            BSR      TWGTST        ;and do test again
2342| 6420                BCC.S   DISINT        ;and continue if OK
2344|
2344| 47FA 1ACA            TSTERR  LEA      TWGFAIL,A3    ;display error msg
2348| 6100 13AC            BSR      DSPMSGR
234C| 7C0C                MOVEQ    #PCCOL,D6          ;reset left margin
234E| 0C00 0027            CMP.B    #TIMOUT,D0        ;timeout error?
2352| 6700 010A            BEQ      CMDERR        ;exit testing if yes
2356| 5239 00FC C19F        ADDQ.B   #1,DSKCNTL        ;else bump low error count
235C| 6406                BCC.S   DISINT        ;skip if no overflow
235E| 5239 00FC C19D        ADDQ.B   #1,DSKCNTH        ;else bump high counter also
2364|
2364|                ; Disable interrupt so disks can be ejected
2364|
2364| 6100 01AA            DISINT  BSR      TWGDSP        ;display Twiggys error count
2368| 117C 0088 0002        MOVE.B   #$88,CMD(A0)      ;set ptr for both drives
236E| 10BC 0087            MOVE.B   #DSABLINT,(A0)    ;send disable cmd
2372| 6100 FA90            BSR      CMDCHK        ;wait until done
2376|
2376|                ; Output initial message
2376|
2376| 47FA 1A56            WRTMSG  LEA      BRNMSG,A3    ;get msg ptr
237A| 6100 1384            BSR      DSPMSG        ;and display it
237E| 1039 00FC C1C3        MOVE.B   CYCLVAL,D0        ;get cycling value
2384| 6100 F2AA            BSR      DSPDEC        ;display as decimal
2388| 7C0C                MOVEQ    #PCCOL,D6          ;set col for window limits
238A|                ; Increment loop count and display it on screen
238A| 5239 00FC C197        CNTINC  ADDQ.B   #1,LCNTLO        ;inc low byte
2390| 6406                BCC.S   DSPTIM        ;skip if no carry
2392| 5239 00FC C195        ADDQ.B   #1,LCNTHI        ;else inc high byte also
2398|
2398| 6100 0130            DSPTIM  BSR      DSPCLK        ;go display time
239C|
239C|                ; Now check time to see if update needed
239C|
239C| 2038 01BC            MOVE.L   HOUR,D0          ;get minute value

```



```

23A0| E998          ROL.L   #4,D0
23A2| 4840          SWAP    D0
23A4| B039 00FC C19B  CMP.B   MINSAB,D0      ;has value changed?
23AA| 6712          BEQ.S   NOCHG      ;skip if not
23AC| 5239 00FC C1C5  ADDQ.B   #1,MINCNT      ;else bump minute count
23B2| 5239 00FC C1C1  ADDQ.B   #1,CYCLCNT      ;and cycle count
23B8| 13C0 00FC C19B  MOVE.B   D0,MINSAB      ;save new minute value
23BE|
23BE|              ; Delay so screen can be read
23BE|
23BE| 6100 E714      NOCHG   BSR     DELAY5      ;delay for 5 secs
23C2|              ; Check to see if run should be ended
23C2|
23C2| 1039 00FC C1C1  MOVE.B   CYCLCNT,D0      ;get cycle count
23C8| 1239 00FC C1C3  MOVE.B   CYCLVAL,D1      ;get desired cycle value
23CE| B001          CMP.B   D1,D0      ;cycle if same or greater
23D0| 6C16          BGE.S   SHUTDOWN
23D2|
23D2|              ; If not, cause double bus fault to restart diagnostics
23D2|              ; First make parameter memory valid
23D2|
23D2| 103C 000F      MOVE.B   #PC,D0      ;set power-cycle boot code
23D6| 6100 F476      BSR     SAV2PM      ;and go set param mem
23DA| 6100 FC14      BSR     PROINIT      ;check for attached hard disk   CHG019
23DE| 6604          BNE.S   @1          ;skip if none                     CHG019
23E0| 6100 FD48      BSR     WFNBSY2      ;else wait until disk ready   CHG019
23E4| 6000 026C      @1    BRA     DORESET      ;then go cause a system reset
23E8|
23E8|              ; Do soft power-off for specified cycle period
23E8| SHUTDOWN
23E8| 4239 00FC C199  CLR.B   TIMFLG      ;reset time save indicator
23EE| 2038 01BC      MOVE.L   CLKDATA+2,D0      ;and save clock data
23F2| 227C 00FC C1A1  MOVE.L   #CLKSAVE,A1
23F8| 01C9 0000      MOVEP.L  D0,(A1)
23FC|
23FC|              ; Disable Twiggy controller to avoid any RAM problems
23FC|
23FC| DSCONT
23FC| 207C 00FC C001  MOVE.L   #DISKMEM,A0      ;set ptr to shared memory
2402| 10BC 0089      MOVE.B   #DIE,(A0)      ;and send "die" cmd
2406| 6100 F9FC      BSR     CMDCHK      ;wait until done
240A| 6552          BCS.S   CMDERR      ;exit if error
240C|
240C| 702D          MOVEQ    #$2D,D0      ;enable alarm setting
240E| 6100 E546      BSR     COPSCMD
2412| 6538          BCS.S   SETERR2
2414|

```



```

2414| 4281          CLR.L   D1
2416| 1239 00FC C1C3  MOVE.B   CYCLVAL,D1      ;get desired shutdown time
241C| 703C          MOVEQ    #60,D0      ;multiply by 60 for seconds
241E| C2C0          MULU     D0,D1
2420| 700C          MOVEQ    #12,D0      ;rotate to send as alarm value
2422| E1B9          ROL.L    D0,D1
2424|
2424| 227C 00FC C1B1  MOVE.L   #ALRMSAV,A1
242A| 03C9 0000      MOVEP.L  D1,(A1)      ;save alarm value
242E| 7405          MOVEQ    #5,D2      ;5 digits for alarm value
2430| 6134          BSR.S    TODSET
2432| 6518          BCS.S    SETERR2
2434|
2434|          ; Make parameter memory valid
2434| 103C 000F      MOVE.B   #PC,D0      ;set power-cycle boot code
2438| 6100 F414      BSR      SAV2PM      ;and go set param mem
243C|          ; And finally send power-off cmd
243C| 7023          MOVEQ    #$23,D0      ;set up enable/power off cmd
243E| 6100 E516      BSR      COPSCMD     ;send it
2442| 6508          BCS.S    SETERR2
2444| 4E71          SELF    NOP
2446| 60FC          BRA.S    SELF      ;goodbye ...
2448|
2448| 703D          SETERR1 MOVEQ    #SERR1,D0      ;set error code
244A| 6002          BRA.S    DSPERR      ;and go display
244C|
244C| 703E          SETERR2 MOVEQ    #SERR2,D0      ;set error code
244E|
244E|          DSPERR
244E|          .IF USERINT = 0
244E|          .ELSE
244E| 45FA 14FB      LEA      IOBRD,A2      ;set icon ptr
2452| 6100 1086      BSR      DSPERRICON    ;display it
2456|          .ENDC
2456|
2456| 6100 F1CA      BSR      DSPCODE
245A| 6000 0140      BRA      MONITOR      ;and exit to monitor
245E|
245E|          ; Error routine if disk cmd not taken
245E|
245E| 08C7 0011      CMDERR  BSET    #DISK,D7      ;set error bit
2462| 6000 EF36      BRA      TSTCHK      ;and exit
2466|
2466|          .PAGE
2466|          ;-----
2466|          ; Subroutine to send clock data. Assumes registers:
2466|          ; D0 = scratch use

```



```

2466|          ;      D1 = clock data
2466|          ;      D2 = digit count
2466|          ;-----
2466|
2466| E999      TODSET  ROL.L   #4,D1          ;get digit
2468| 1001      MOVE.B   D1,D0          ;set up for COPS as 1X
246A| 0200 000F      ANDI.B   #$0F,D0      ; where X = digit for clock
246E| 0000 0010      ORI.B    #$10,D0
2472| 6100 E4E2      BSR      COPSCMD      ;and send it
2476| 6504      BCS.S     SETXIT      ;exit if error
2478| 5342      SUBQ     #1,D2          ;decr count
247A| 66EA      BNE.S     TODSET      ;and loop until done
247C| 4E75      SETXIT  RTS
247E|
247E|          .PAGE
247E|          ;-----
247E|          ; Subroutine to do Twiggy testing
247E|          ; Expects
247E|          ;      D0 = scratch use          A0 = shared memory address
247E|          ;      D1 = drive parameters      A1 = unused
247E|          ;      D2 = FDIR timeout value      A2 = unused
247E|          ;      D3 = unused          A3 = VIA address for FDIR access
247E|          ;      D4 = loop count for reads
247E|          ;-----
247E|
247E| 117C 0088 0002      TWGTST  MOVE.B   #$88,CMD(A0)      ;enable interrupts from both drives
2484| 10BC 0086      MOVE.B   #ENBLINT,(A0)      ;do it
2488| 6100 F97A      BSR      CMDCHK      ;wait until done
248C| 6536      BCS.S     TERR      ;exit if error
248E| 08AB 0004 0004      BCLR   #FDIR,DDRB1(A3)      ;enable FDIR bit
2494| 243C 00C0 0000      MOVE.L   #FDIRTIME,D2      ;set timeout value for FDIR
249A|
249A| 03C8 0004      TWGLOOP  MOVEP.L  D1,DRV(A0)      ; set disk ptrs
249E| 4228 0002      MOVE.B   #READS,CMD(A0)      ; set for read operation
24A2| 10BC 0081      MOVE.B   #EXRW,(A0)      ; and go do it
24A6| 6100 F996      BSR      CHKFIN      ; wait
24AA| 6516      BCS.S     TOOLONG      ; exit if timeout
24AC| 1028 0010      MOVE.B   STAT(A0),D0      ; get disk return code
24B0| 6100 F9C0      BSR      CLRFDIR      ; clear interrupt indicator
24B4| 650C      BCS.S     TOOLONG
24B6| 4A00      TST.B     D0          ;any error?
24B8| 660A      BNE.S     TERR      ; and exit if error
24BA| 5241      ADDQ     #1,D1          ;incr track ptr
24BC| 5344      SUBQ     #1,D4          ;decrement count
24BE| 66DA      BNE.S     TWGLOOP      ;loop until done
24C0| 4E75      RTS
24C2|

```




```

24C2| 7027          TOOLONG MOVEQ    #TIMOUT,D0      ;set error code
24C4| 003C 0001     TERR      ORI.B    #$01,CCR      ;set indicator
24C8|
24C8| 4E75          RTS                          ;and exit
24CA|
24CA|              ;-----
24CA|              ; Subroutine to display clock reading as D HH MM SS
24CA|              ;-----
24CA|
24CA| 47FA 1914     DSPCLK LEA      TIMMSG,A3          ;get msg ptr
24CE| 6100 1230     BSR      DSPMSG          ;and display it
24D2| 5246         ADDQ      #1,D6              ;add extra space
24D4| 6100 EDCA     BSR      READCLK           ;go read clock
24D8| 2038 01BC     MOVE.L    CLKDATA+2,D0         ;get time (minus Ey/dd digits)
24DC| 227C 00FC C1A1 MOVE.L    #CLKSAVE,A1         ;and save it
24E2| 01C9 0000     MOVEP.L D0,(A1)
24E6|
24E6| E998          ROL.L     #4,D0              ;get day value
24E8| 7201          MOVEQ     #1,D1              ;set # of digits to display
24EA| 6100 F18C     BSR      OUTCH            ;and display it
24EE| 5246         ADDQ      #1,D6              ;bump col ptr
24F0|
24F0| E198          ROL.L     #8,D0              ;get hour
24F2| 7202          MOVEQ     #2,D1              ;and display
24F4| 6100 F182     BSR      OUTCH
24F8| 5246         ADDQ      #1,D6
24FA|
24FA| E198          ROL.L     #8,D0              ;display minute
24FC| 7202          MOVEQ     #2,D1
24FE| 6100 F178     BSR      OUTCH
2502| 5246         ADDQ      #1,D6
2504|
2504| E198          ROL.L     #8,D0              ;display seconds
2506| 7202          MOVEQ     #2,D1
2508| 6100 F164     BSR      OUTCHR
250C|
250C|              .IF USERINT = 1
250C| 7C0C          MOVEQ     #PCCOL,D6          ;set col for window
250E|              .ENDC
250E|
250E| 4E75          RTS
2510|
2510|
2510|              .PAGE
2510|              ;-----
2510|              ; Subroutine to display Twiggy error count
2510|              ;-----

```



```

2510|
2510| 48E7 C010          TWGDSP  MOVEM.L D0-D1/A3,-(SP)  ;save regs
2514| 47FA 190D          LEA      TWGRSLT,A3           ;output msg
2518| 6100 11E6          BSR      DSPMSG
251C| 267C 00FC C19D     MOVE.L  #DSKCNTNTH,A3         ;set ptr to error count
2522| 010B 0000          MOVEP   (A3),D0              ;get count
2526| 7204              MOVEQ    #4,D1                ;# of digits to display
2528| 6100 F144          BSR      OUTCHR
252C|
252C|                  .IF  USERINT = 1
252C| 7C0C              MOVEQ    #PCCOL,D6              ;set col for window
252E|                  .ENDC
252E|
252E| 4CDF 0803          MOVEM.L  (SP)+,D0-D1/A3        ;restore and exit
2532| 4E75              RTS
2534|
2534|                  .ENDC
2534|
2534|                  .PAGE
2534|                  .IF  USERINT = 0
2534|                  .ENDC
2534|
2534|                  .INCLUDE RM248.M.TEXT
2534|
2534|                  .PAGE
2534|
2534|-----
2534| ; Monitor code - first displays requested icons, error codes or messages,
2534| ; and then outputs menu of options to user and awaits input
2534|-----
2534|
2534| INITMON                                ;entry point for displays
2534|                  .IF  ROM4K = 0
2534| 6100 DB04          BSR      SAVEREGS                ;save registers
2538| 4287              CLR.L    D7                      ;reset reg for error indicators
253A| 4238 02A2          CLR.B    STATFLGS                ; and status flags
253E| 08F8 0001 02A2     BSET     #NOCONT,STATFLGS        ;set external entry indicator
2544|
2544| INIT1
2544| 48E7 8030          MOVEM.L  D0/A2-A3,-(SP)          ;save incoming arguments
2548| 6100 F7FC          BSR      DSABLDISK                ;disable ints from both drives
254C| 6100 E55C          BSR      RSTKBD                  ;reset keyboard
2550| 6100 E572          BSR      CLRRST                  ;and clear reset
2554| 6100 0A76          BSR      CursorInit              ;init cursor and mouse
2558| 4CDF 0C01          MOVEM.L  (SP)+,D0/A2-A3          ;restore arguments
255C|
255C| INIT2                                ;internal entry point

```



```

255C|          .IF  DEBUG = 0
255C| 3E7C 0480  MOVEA  #STKBASE,SP      ;reset stack pointer      RM000
2560|          .ENDC
2560|
2560|          .IF  USERINT = 1
2560| 48E7 8030  MOVEM.L D0/A2-A3,-(SP)  ;save incoming arguments
2564|          .ENDC
2564|          .ENDC                      ;{ROM4K}
2564|
2564| 6100 E320  BSR      SETVLATCH      ;set video latch
2568|
2568|          .IF  USERINT = 0
2568|          .ELSE
2568| 6100 0B6C  BSR      DRAWDESK      ;display the desktop
256C| 6100 0BFA  BSR      MAKEALERT     ;draw alert box (in case no icon display)
2570|
2570| 4CDF 0401  INIT3   MOVEM.L (SP)+,D0/A2  ;restore arguments
2574|
2574| 220A      MOVE.L  A2,D1            ;icon display?
2576| 6704      BEQ.S   @0              ;skip if no
2578| 6100 0FB2  BSR      DSPALRTICON    ;go do icon display
257C|
257C| 4A40      @0      TST    D0          ;error code display?
257E| 6712      BEQ.S   @2              ;skip if no
2580| 220A      MOVE.L  A2,D1            ;icon displayed?
2582| 660A      BNE.S   @1              ;skip if yes
2584| 7A7E      MOVEQ   #MSGROW,D5       ;else display error code on same line
2586| 7C12      MOVEQ   #CODECOL,D6     ; as error msg
2588| 6100 F0A6  BSR      DSPDEC         ;display as decimal #
258C| 6004      BRA.S   @2
258E|
258E| 6100 F092  @1      BSR      DSPCODE   ;output error code under icon
2592|
2592| 265F      @2      MOVE.L  (SP)+,A3   ;restore msg ptr
2594|          .ENDC
2594|
2594| 200B      MOVE.L  A3,D0            ;message display?
2596| 6704      BEQ.S   MONITOR
2598|
2598|          .IF  USERINT = 0
2598|          .ELSE
2598| 6100 1140  BSR      DSPALRTMSG     ;go display message in alert box
259C|          .ENDC
259C|
259C|          MONITOR                    ;entry point for no screen setup
259C|          .IF  ROM4K = 0
259C| 007C 0700  ORI      #$0700,SR     ;disable all interrupts

```



```

25A0| 6100 E10A          BSR      SETVCTRS          ;set vectors for ROM space          CHG028
25A4|                  .ELSE
25A4|                  .ENDC
25A4|
25A4|                  .IF ROM4K = 0
25A4|          ;-----
25A4|          ; Now output first level menu, prompt line and cursor. Do preliminary
25A4|          ; check to see if CONTINUE option can be displayed. This is the Customer
25A4|          ; mode level of the monitor code.
25A4|          ;-----
25A4|
25A4|          LEVEL1
25A4|          .IF USERINT = 1
25A4| 4278 053A          CLR      RECTCNT          ;clear active rectangle count
25A8| 0238 000F 02A2    ANDI.B  #$0F,STATFLGS    ;init flags
25AE| 08F8 0006 02A2    BSET     #BTN,STATFLGS    ;set operating with buttons flag
25B4|
25B4| 0838 0001 02A2    BTST     #NOCONT,STATFLGS ;display continue?
25BA| 6630          BNE.S    OTHRBTNS          ;skip if no
25BC| 2038 0180    MOVE.L   STATUS,D0          ;get test status
25C0| 0280 001E 3FFA    ANDI.L   #CONTMSK,D0    ;mask don't cares
25C6| 661E          BNE.S    @1                ;skip if error that disallows continuing
25C8| 4A78 0188    TST      BOOTMEM          ;check boot memory area for R/W errors
25CC| 6618          BNE.S    @1                ;skip if any errors
25CE|
25CE| 327C 2956    MOVE     #BTN2STRT,A1        ;display CONTINUE button
25D2| 103C 00F1    MOVE.B   #KEY2,D0          ;with alternate keycode
25D6| 47FA 18A0    LEA      CONTMSG,A3        ;and description
25DA| 347C 2CE8    MOVEA.L  #BTN2MSG,A2        ;and location          RM000
25DE| 4281          CLR.L    D1                ;don't append '...' string
25E0| 6100 0C98    BSR      MAKEBTN
25E4| 6006          BRA.S    OTHRBTNS          ;and go make other buttons
25E6| 08F8 0001 02A2    @1      BSET     #NOCONT,STATFLGS ;set indicator for no CONTINUE option
25EC|
25EC|          OTHRBTNS
25EC|          .ENDC
25EC|
25EC|          DOMENU
25EC|          .IF USERINT = 0
25EC|          .ELSE
25EC| 0838 0000 02A2    BTST     #NORSTRT,STATFLGS ;display RESTART button?
25F2| 6616          BNE.S    @1                ;skip if not
25F4| 327C 1876    MOVE     #BTN1STRT,A1        ;else do display
25F8| 103C 00F4    MOVE.B   #KEY1,D0
25FC| 47FA 185A    LEA      RTRYMSG,A3
2600| 347C 1C08    MOVEA.L  #BTN1MSG,A2        ;
2604| 4281          CLR.L    D1                ;don't append '...' string

```



```

2606| 6100 0C72          BSR      MAKEBUTN
260A|
260A| 327C 3A36          @1      MOVE    #BTN3STRT,A1      ;display STARTUP button
260E| 303C 00F2          MOVE    #KEY3,D0
2612| 47FA 1884          LEA      STRTMSG,A3
2616| 347C 3DC8          MOVEA   #BTN3MSG,A2      ;
261A| 72FF              MOVEQ    #-1,D1      ;append '...' string
261C| 6100 0C5C          BSR      MAKEBUTN
2620|
2620|          ;          MOVE.L   #KBDBFR,KBDQPTR ;init queue ptr
2620|
2620| 6100 09EC          BSR      CursorDisplay ;display mouse cursor
2624| 08F8 0005 02A2     BSET     #CHKCMD,STATFLGS ;require user keyboard input to be prefaced
262A|          ; by the CMD key
262A| 6100 061A          GETL1    BSR      GETINPUT      ;and go wait for input
262E| 6500 00B6          BCS      GETERR      ;exit if error
2632| 6100 09B6          BSR      CursorHide      ;remove cursor from screen
2636|
2636|          .ENDC
2636|
2636|          ; Check if input valid.  If invalid, beep speaker.
2636|
2636| 0C00 00F2          CMP.B     #KEY3,D0      ;alternate boot?
263A| 6608          BNE.S     @2
263C|
263C|          .IF USERINT = 0
263C|          .ELSE
263C| 6100 0A9C          BSR      CLRDESK      ;close the alert box
2640| 6000 F2B6          BRA      BOOTMENU      ;and go display boot menu
2644|          .ENDC
2644|
2644| 0838 0000 02A2     @2      BTST     #NORSTRT,STATFLGS ;RESTART button displayed?
264A| 6612          BNE.S     CONTCHK      ;skip if not
264C| 0C00 00F4          CMP.B     #KEY1,D0      ;retry?
2650| 660C          BNE.S     CONTCHK      ;skip if not
2652|
2652|          DORESET
2652| 4287          CLR.L     D7      ;clear error reg
2654| 4A39 00FC E010     TST.B     SETUPON      ;turn on setup bit
265A| 6000 DB2E          BRA      BEGIN3      ;and restart diags
265E|
265E|          CONTCHK
265E|          .IF USERINT = 1
265E| 0838 0001 02A2     BTST     #NOCONT,STATFLGS ;continue option displayed?
2664| 666C          BNE.S     @4      ;skip if not
2666| 0C00 00F1          CMP.B     #KEY2,D0      ;continue option selected?

```



```

266A| 6666                BNE.S   @4
266C|
266C|                ; continue from point of failure
266C|
266C| 6100 0A6C            BSR      CLRDESK        ;clear desktop          CHG008
2670| 0287 7000 0000      ANDI.L   #ALTMSK,D7    ;erase error indicators
2676| 2038 0180            MOVE.L   STATUS,D0     ;get power-up status
267A| 0800 0000            BTST     #MMU,D0       ;MMU error?
267E| 6600 E100            BNE      VIA2TST       ;yes - continue from VIA tests
2682|
2682| 0280 008F FFFF      ANDI.L   #BOOTMSK,D0    ;check if error that continues to boot attempt
2688| 6700 F05C            BEQ       BOOTCHK       ;skip if yes
268C| 2F00                MOVE.L   D0,-(SP)        ;else save status
268E| 6100 0AF0            BSR      MAKETEST       ;make test window
2692|
2692|                ; do init for continue to other tests
2692| 103C 0070            MOVE.B   #$70,D0         ;turn off mouse
2696| 6100 E2BE            BSR      COPSCMD
269A| 201F                MOVE.L   (SP)+,D0       ;restore status
269C|
269C| 0800 0002            BTST     #VID,D0         ;serial # error?
26A0| 670C                BEQ.S     @1             ;skip if not
26A2| 327C 1DF6            MOVEA   #CPUSTRT,A1     ;display CPU icon
26A6| 6100 0ECC            BSR      INVICON
26AA| 6000 E6B0            BRA      PARTST         ;continue with parity test
26AE|
26AE| EE88                @1      LSR.L   #7,D0     ;skip other CPU errors
26B0| 4A00                TST.B    D0             ;clock error?
26B2| 6600 EC3A            BNE      CONFIG         ;yes - continue with config check
26B6|
26B6| E488                LSR.L   #2,D0
26B8| 4A00                TST.B    D0             ;RS232 error?
26BA| 670C                BEQ.S     @2             ;skip if not
26BC| 327C 1E12            MOVEA   #IOSTRT,A1     ;else display I/O board icon
26C0| 6100 0EB2            BSR      INVICON
26C4| 6000 EA46            BRA      DSKTST         ;cont with disk test
26C8|
26C8| 4A39 00FC E01E      @2      TST.B    PARON     ;must be memory error - reenable parity
26CE| 6000 E930            BRA      IOTST         ; and continue with I/O board testing
26D2|
26D2|                .ENDC
26D2|
26D2|                @4
26D2|                .IF USERINT = 0
26D2|                .ENDC
26D2|
26D2| 0C00 00F6            CMP.B     #SKEY,D0         ; service mode desired?
26D6| 6700 0064            BEQ       LEVEL2        ; skip if yes

```



```

26DA|
26DA|           ; Indicate invalid by beeping speaker
26DA|
26DA| GETL1XIT
26DA| 6100 0030      BSR      SQUAWK          ; sorry charlie
26DE|              .IF USERINT = 0
26DE|              .ELSE
26DE| LEV1LOOP
26DE| 6100 092E      BSR      CursorDisplay    ;redisplay cursor
26E2| 6000 FF46      BRA.S    GETL1          ;go get more input
26E6|
26E6|           ; Error exit - go output error and return to level 1
26E6|
26E6| GETERR
26E6| 45FA 1263      LEA      IOBRD,A2          ;get I/O board icon
26EA| 97CB          SUBA.L   A3,A3            ;no error message
26EC| 6000 FE6E      BRA      INIT2
26F0|              .ENDC
26F0|
26F0|              .ENDC                      ;{ROM4K}
26F0|              .PAGE
26F0|
26F0| ;-----
26F0| ; Subroutine to clear video page of memory or write arbitrary long word
26F0| ; pattern to entire screen (WRTSCRN entry point).
26F0| ;-----
26F0|
26F0| 4280           CLRSCRN CLR.L   D0          ; write 0's for white screen
26F2| WRTSCRN        ; entry pt for write to screen (assumes D0 set)
26F2|              MOVE.L   SCRNBASE,A0        ; get screen base address
26F6| 323C 1FFD      MOVE     #HEX8K-3,D1      ; set longs count
26FA| 20C0          @1      MOVE.L   D0,(A0)+    ; clear for video page
26FC| 51C9 FFFC      DBF      D1,@1
2700| 4E75           RTS
2702|
2702|              .IF ROM4K = 0
2702|              .IF USERINT = 0
2702|              .ENDC
2702|              .PAGE
2702|              .IF USERINT = 0
2702|              .ELSE
2702| ;-----
2702| ; Subroutine to read keycode from COPS - returns down transitions in D0
2702| ;-----
2702|
2702| ReadKey
2702| 6100 0634      BSR      WT4INPUT
2706| 4A00           TST.B    D0                ;ignore "up" transitions and mouse data

```



```

2708| 6AF8          BPL.S   ReadKey
270A| 4E75          RTS              ;exit with data
270C|
270C|          .ENDC
270C|
270C|          ;-----
270C|          ; Subroutine to beep speaker for invalid input
270C|          ;-----
270C|
270C| 7020          SQUAWK  MOVEQ  #$20,D0          ; set frequency
270E| 323C 00FA          MOVE   #250,D1          ; 1/8 sec duration
2712| 7404          MOVEQ  #4,D2          ; low volume
2714| 6100 E3E0          BSR    TONE          ; and go do it
2718| 4E75          RTS
271A|
271A|          ;-----
271A|          ; Subroutine to convert keycodes to Ascii
271A|          ; Inputs: D0 = keycode (word)
271A|          ; Outputs: D0 = Ascii (byte) or =2 if input invalid
271A|          ;-----
271A|
271A|          KeyToAscii
271A| 48E7 4080          MOVEM.L D1/A0,-(SP)      ;save regs
271E| 41FA 119C          LEA    AsciiTable,A0     ;keycode to ascii table
2722| 3200          MOVE    D0,D1          ;keycode to convert
2724| 0241 007F          ANDI   #$007F,D1        ;ensure valid
2728| 0441 0020          SUBI   #32,D1          ;decrement for table      RM000
272C| 6A04          BPL.S   @1              ;skip if valid              RM000
272E| 7002          MOVEQ  #2,D0          ;else set for invalid char    RM000
2730| 6004          BRA.S   @2              ;                             RM000
2732| 1030 1000          @1    MOVE.B  0(A0,D1.W),D0 ;get ascii
2736| 4CDF 0102          @2    MOVEM.L (SP)+,D1/A0 ;restore
273A| 4E75          RTS              ;exit
273C|
273C|          .PAGE
273C|          ;-----
273C|          ; Monitor level 2 (Service mode) - enables access to memory and disk
273C|          ;-----
273C|
273C|          LEVEL2
273C|          .IF  USERINT = 0
273C|          .ELSE
273C| 6100 099C          BSR    CLRDESK          ;display the desktop
2740|
2740|          .IF  BMENU = 0
2740|          .ENDC

```




```

2740|
2740|           ; make window for output, and display menu line and pull down menu
2740|
2740| 6100 00D2           BSR      MAKESVCW      ;output service window
2744| 6100 005A       DSPMENU BSR      WRTMENU
2748|
2748|           ; do final initialization and await input
2748|
2748| 6100 08C4           BSR      CursorDisplay ;display cursor
274C| -----
274C|           ; Program NMI key
274C|           ;
274C|           ;      MOVEQ    #$5A,D0          ;set / key for NMI
274C|           ;      BSR      COPSCMD
274C|           ;      MOVEQ    #$61,D0
274C|           ;      BSR      COPSCMD
274C| -----
274C|
274C| GETLEV2
274C| 6100 04F8           BSR      GETINPUT      ;and go await input
2750| 6594              BCS      GETERR        ;exit if error
2752| 6100 0896           BSR      CursorHide    ;else remove cursor from screen and go
2756|                                     ; analyze input
2756|
2756|           .ENDC
2756|
2756|           .IF USERINT = 0
2756|           .ENDC
2756|
2756|           ; Check for valid input
2756|
2756| 0C00 00F4           CMP.B    #KEY1,D0      ; display memory?
275A| 6700 00DA           BEQ      DSPMEM
275E| 0C00 00F1           CMP.B    #KEY2,D0      ; set memory?
2762| 6700 013E           BEQ      SETMEM
2766|
2766| 0C00 00F2           CMP.B    #KEY3,D0      ; call routine
276A| 6700 019C           BEQ      CALLRTN
276E|
276E|           .IF ROM16K = 1
276E| 0C00 00F3           CMP.B    #KEY4,D0      ; loop?
2772| 6700 01C0           BEQ      LOOPTST
2776|           .ENDC
2776|
2776| 0C00 00E4           CMP.B    #KEY5,D0      ; video adjust?
277A| 6700 0278           BEQ      VIDAJUST
277E|

```



```

277E|          .IF BURNIN = 1
277E| 0C00 00E1      CMP.B  #KEY6,D0          ; power cycle?
2782| 6700 02E8      BEQ    PowerCycle
2786|          .ENDC
2786|
2786| 0C00 00E2      CMP.B  #KEY7,D0          ; quit?
278A| 6610          BNE.S  @1
278C| 08B8 0000 02A2  BCLR   #NORSTRT,STATFLGS ;clear no reset status flag
2792| 4280          CLR.L  D0                  ;set parms for levell - no error code
2794| 95CA          SUBA.L  A2,A2              ;no icon display
2796| 97CB          SUBA.L  A3,A3              ;no message display
2798| 6000 FDC2      BRA     INIT2            ;and go back to levell
279C|
279C| 6000 02D6      @1    BRA     INVALID          ; else invalid input
27A0|
27A0|          .PAGE
27A0|          .IF USERINT = 1
27A0|          ;-----
27A0|          ; Routine to display the preliminary pull-down menu
27A0|          ;-----
27A0|
27A0| WRTMENU
27A0|          .IF BMENU = 0
27A0|          .ELSE
27A0| 4278 053A      CLR     RectCnt          ;clear active rectangle count
27A4| 0238 000F 02A2  ANDI.B  #$0F,STATFLGS    ;init flags
27AA| 7012          MOVEQ   #MENUWIDTH,D0      ;set menu parms
27AC| 7207          MOVEQ   #MITEMS,D1         ;set # of items in menu
27AE| C2FC 000B      MULU    #MENULEN,D1       ;length depends on # of items
27B2| 47FA 170D      LEA     MENUHDG,A3        ;set ptr for menu heading
27B6| 6118          BSR.S   DSPMENUBOX        ;go display blank menu box w/ heading
27B8|
27B8| 7807          MOVEQ   #MITEMS,D4         ;set # of items in menu
27BA| 327C 05A2      MOVE    #MENUSTRT,A1      ;set menu starting point
27BE| 347C 0658      MOVE    #MENU1MSG,A2     ;menu items display address
27C2| 47FA 1705      LEA     DISPMMSG,A3      ;set ptr to menu entries
27C6| 49FA 1771      LEA     MENUID,A4        ;ptr to id's for menu entries
27CA| 6100 0B9E      BSR     MAKEMENU         ;go fill in the menu
27CE|
27CE|          .ENDC          ;{MENU}
27CE|
27CE| 4E75          RTS
27D0|
27D0|          .IF BMENU = 1
27D0|          ;-----
27D0|          ; Subroutine to display blank menu box with heading
27D0|          ; Inputs:

```



```

27D0|      ;      D0 = menu width
27D0|      ;      D1 = menu length
27D0|      ;      A3 = menu heading
27D0|      ; Outputs:
27D0|      ;      None
27D0|      ; Side Effects:
27D0|      ;      D2/A1,A3 trashed
27D0|      ;-----
27D0|
27D0|      DSPMENUBOX
27D0| 48E7 C000      MOVEM.L D0-D1,-(SP)      ;save regs
27D4| 08F8 0007 02A2      BSET      #MENU,STATFLGS      ;set working with menu flag
27DA| 6100 0938      BSR      CLRMENU      ;first clear the menu bar
27DE| 5441      ADDQ      #2,D1      ;bump length for bottom border
27E0| 327C 05A2      MOVE      #MENUSTRT,A1      ;set menu starting point
27E4| 6100 0A00      BSR      MAKEBOX      ;display the box
27E8| 327C 0111      MOVEA     #MENULOC,A1      ;set up menu heading display point
27EC| 6100 0C18      BSR      GETROWCOL      ;convert to screen ptrs
27F0|
27F0| 4281      CLR.L      D1      ;don't display '...' string
27F2| 6100 0E40      BSR      DSPSTRING      ;display the title
27F6| 97CA      SUBA.L     A2,A3      ;get length of menu title
27F8| 240B      MOVE.L     A3,D2      ;move to working reg
27FA| 5442      ADDQ      #2,D2      ;add extra bytes to cover entire title
27FC| 0202 00FE      ANDI.B     #$FE,D2      ;ensure it's even
2800| 3002      MOVE      D2,D0      ;save as width of menu heading "box"
2802| 720E      MOVEQ      #14,D1      ;set height for "box"
2804| 74FF      MOVEQ      #-1,D2      ;set fill pattern
2806| 92FC 005B      SUB.W     #91,A1      ;decrement start pt by 1 row + 1 byte
280A| 6100 0922      BSR      INVERSE      ;go hilite it
280E| 4CDF 0003      MOVEM.L (SP)+,D0-D1      ;restore regs
2812| 4E75      RTS
2814|
2814|      .ENDC      ;{MENU}
2814|
2814|      ;-----
2814|      ; Subroutine to create Service mode window
2814|      ;-----
2814|
2814|      MAKESVCW
2814| 327C 0EDA      MOVEA     #SVCSTRT,A1      ;left corner point      RM000
2818| 7042      MOVEQ      #SVCWIDTH,D0      ;width of window
281A| 223C 0000 0140      MOVE.L     #SVCHIGH,D1      ;height
2820| 47FA D830      LEA      SVCMSG,A3      ;title
2824| 6100 09A0      BSR      MAKEWINDOW      ;go do it
2828| 31FC 003E 0300      MOVE      #FIRSTROW,CRTROW      ;init screen ptrs
282E| 31FC 0018 0302      MOVE      #FIRSTCOL,CRTCOL

```



```

2834| 4E75                RTS
2836|
2836|                .ENDC                ;{USERINT}
2836|
2836|                ;-----
2836|                ; Do display memory operation
2836|                ;-----
2836|
2836| 6100 036C      DSPMEM BSR      GETA                ;go get address
283A| 6500 0238      BCS      INVALID
283E| 4A43          TST      D3                ;if no input go back to menu line
2840|
2840|                .IF USERINT = 0
2840|                .ELSE
2840| 6700 0240      BEQ      LEV2LOOP
2844|                .ENDC
2844|
2844| 0882 0000      BCLR      #0,D2                ;ensure even address
2848| 2442          MOVE.L   D2,A2                ;and save
284A|
284A|                ; Check for all input on one line
284A| 6100 034A      BSR      GETCH                ;read queue to see if more input
284E| 6506          BCS.S    @1                ;skip if not
2850| 0C00 0020      CMPI.B   #' ',D0                ;must be a space separator
2854| 6708          BEQ.S    RDCNT                ;skip if yes
2856|
2856| 47FA 1774      @1      LEA      CNTMSG,A3                ;display count prompt
285A| 6100 0236      BSR      PROMPT
285E|
285E|                ; Decode count input and do display
285E|
285E| 7204          RDCNT    MOVEQ   #4,D1                ;go get count (max of $FFFF)
2860| 6100 0350      BSR      GETPARM
2864| 6500 020E      BCS      INVALID
2868| 6100 02E6      BSR      PUTLF                ;set display ptrs and space 1 line
286C| 4A43          TST      D3                ;set default count if no input
286E| 6702          BEQ.S    @4
2870| 6002          BRA.S    @5
2872|
2872| 7410          @4      MOVEQ   #16,D2                ;set default count
2874|
2874|                ; Do display of memory
2874|
2874| 200A          @5      MOVE.L   A2,D0                ;get display address
2876| 7208          MOVEQ   #8,D1                ;and display it
2878| 6100 EDFE      BSR      OUTCH
287C|

```



```

287C|          .IF USERINT = 0
287C|          .ELSE
287C| 5846      ADDQ    #4,D6          ;bump col for data display
287E|          .ENDC
287E| 7808      MOVEQ   #8,D4          ;set loop count
2880| 301A      @6      MOVE   (A2)+,D0      ;read data word
2882| 7204      MOVEQ   #4,D1          ;display it
2884| 6100 EDF2      BSR     OUTCH
2888| 5246      ADDQ    #1,D6          ;add space
288A| 5344      SUBQ    #1,D4          ;loop for one line
288C| 66F2      BNE.S   @6
288E|
288E| 5182      SUBQ.L   #8,D2          ;decr data count          RM000
2890| 5182      SUBQ.L   #8,D2          ;                          RM000
2892| 6F06      BLE.S   @7          ;exit if done
2894| 6100 02BA      BSR     PUTLF      ;go to next line
2898| 60DA      BRA.S   @5          ;and continue until done
289A|
289A| 6100 02B4      @7      BSR     PUTLF      ;add blank line
289E|
289E|          .IF USERINT = 0
289E|          .ELSE
289E| 6000 01E2      BRA     LEV2LOOP      ;continue level2 loop
28A2|          .ENDC
28A2|
28A2|          .PAGE
28A2|          ;-----
28A2|          ; Do set memory operation - enables setting of bytes, words or longs
28A2|          ; up to 24 bytes max. Decodes data length to determine type of operation.
28A2|          ;-----
28A2|
28A2| 6100 0300      SETMEM BSR     GETA          ;go get address
28A6| 6500 01CC      BCS     INVALID      ;abort if invalid
28AA| 4A43      TST     D3          ;any input?
28AC|
28AC|          .IF USERINT = 0
28AC|          .ELSE
28AC| 6700 01D4      BEQ     LEV2LOOP      ;abort if none
28B0|          .ENDC
28B0|
28B0| 2442      MOVE.L   D2,A2          ;save target address
28B2|          ; Check for all input on one line
28B2|
28B2| 6100 02E2      BSR     GETCH      ;read queue to see if more input
28B6| 6506      BCS.S   @1          ;skip if not
28B8| 0C00 0020      CMPI.B  #' ',D0      ;must be a space separator
28BC| 6708      BEQ.S   RDDTA      ;skip if yes

```

```

28BE|
28BE| 47FA 1705      @1      LEA      DATMSG,A3      ;else output data prompt
28C2| 6100 01CE      BSR      PROMPT
28C6|
28C6|      ; Decode parameter input and do operation
28C6| 7208      RDDTA  MOVEQ   #8,D1      ;get max of 8 chars
28C8| 6100 02E8      BSR      GETPARM
28CC| 6500 01A6      BCS      INVALID
28D0| 4A43      TST      D3      ;any input?
28D2|
28D2|      .IF  USERINT = 0
28D2|      .ELSE
28D2| 6700 01AE      BEQ      LEV2LOOP
28D6|      .ENDC
28D6|
28D6|      ; write data to memory
28D6|
28D6| 0C03 0002      CMP.B   #2,D3      ;first test for byte operation
28DA| 6E04      BGT.S   @1
28DC| 14C2      MOVE.B   D2,(A2)+      ;write byte
28DE| 6014      BRA.S   @3
28E0| 200A      @1      MOVE.L   A2,D0      ;ensure even address for word or long op
28E2| 0880 0000      BCLR    #0,D0
28E6| 2440      MOVE.L   D0,A2
28E8| 0C03 0004      CMP.B   #4,D3      ;test for word op
28EC| 6E04      BGT.S   @2
28EE| 34C2      MOVE     D2,(A2)+      ;write word
28F0| 6002      BRA.S   @3
28F2|
28F2| 24C2      @2      MOVE.L   D2,(A2)+      ;write long
28F4|
28F4| 6100 02A0      @3      BSR      GETCH      ;read input queue
28F8| 650A      BCS.S   @4      ;skip if none
28FA| 0C00 0020      CMPI.B   #' ',D0      ;must be a space separator
28FE| 6600 0174      BNE      INVALID      ;exit if error
2902| 60C2      BRA.S   RDDTA      ;else continue operation
2904|
2904|      @4
2904|      .IF  USERINT = 0
2904|      .ELSE
2904| 6000 017C      BRA      LEV2LOOP      ;continue level2 loop
2908|      .ENDC
2908|
2908|      .PAGE
2908|
2908|      ;-----
2908|      ; Do 'call' function - ensures address is on word boundary.
2908|      ;-----

```



```

2908|
2908| 6100 029A      CALLRTN BSR      GETA          ;go get address
290C| 6500 0166      BCS      INVALID
2910| 4A43          TST      D3          ;abort if no input
2912|
2912|          .IF USERINT = 0
2912|          .ELSE
2912| 6700 016E      BEQ      LEV2LOOP      ;continue level2 loop
2916|          .ENDC
2916|
2916| 0882 0000      BCLR      #0,D2          ;else ensure on word boundary
291A| 21C2 01F8      MOVE.L   D2,A6SAV      ;save for jump
291E|
291E|          ; load registers from save area before jumping
291E|
291E| 4DF8 01C0      LEA      DATARG$A6      ;get ptr
2922| 4CDE 3FFF      MOVEM.L (A6)+,D0-D7/A0-A5 ;load regs
2926| 2C78 01F8      MOVE.L   A6SAV,A6      ;restore address
292A| 4E96          JSR      (A6)          ;and do call
292C| 6100 D70C      BSR      SAVEREGS      ;save registers on exit
2930|
2930|          .IF USERINT = 0
2930|          .ELSE
2930| 6000 0150      BRA      LEV2LOOP      ;continue level2 loop
2934|          .ENDC
2934|
2934|          .IF      ROM16K = 1
2934|          .PAGE
2934|          ;-----
2934|          ; Do loop on diagnostic
2934|          ;-----
2934|
2934|          LOOPTST
2934|          .IF USERINT = 1
2934| 6100 FEDE      BSR      MAKESVCW      ;redraw service window
2938| 6100 0216      BSR      PUTLF        ;add blank line and setup ptrs
293C| 47FA 1602      LEA      TSTMENU,A3    ;set ptr to test choices
2940| 7818          MOVEQ    #FIRSTCOL,D4   ;set left margin
2942| 6100 0DBC      BSR      DSPMSG        ;and go do display choices
2946|          .ENDC
2946|
2946| 47FA 168C      LEA      TSTMSG,A3      ;display test routine prompt
294A| 6100 0146      BSR      PROMPT
294E| 6500 0124      BCS      INVALID      ;skip if bad input
2952| 4A43          TST      D3          ;any input?
2954|
2954|          .IF USERINT = 0

```

```

2954|          .ELSE
2954| 6608          BNE.S    @0          ;skip if yes
2956| 6100 FEBC          BSR      MAKESVCW      ;else redraw service window
295A| 6000 0126          BRA      LEV2LOOP      ;and return to level 2
295E|          .ENDC
295E|
295E| 5343          @0      SUBQ    #1,D3          ;ensure only one char input
2960| 6600 0112          BNE      INVALID        ;skip if more than one
2964|
2964| 7201          MOVEQ   #1,D1          ;go get one character of input
2966| 6100 024A          BSR      GETPARM
296A| 0C02 000C          CMP.B   #MAXTEST,D2      ;check if within max range
296E| 6200 0104          BHI      INVALID        ;exit if not
2972| 5342          SUBQ    #1,D2          ;decr for index
2974| 6B00 00FE          BMI      INVALID        ;skip if negative (i.e., 0 was input)
2978|
2978| ;-----
2978| ;   Program NMI key to exit loop
2978| ;
2978| ;       LEA      LOOPEND,A3          ;set NMI vector
2978| ;       MOVE.L   A3,NMIVCT
2978| ;       MOVEQ    #$5A,D0          ;set / key for NMI
2978| ;       BSR      COPSCMD
2978| ;       MOVEQ    #$61,D0
2978| ;       BSR      COPSCMD
2978| ;-----
2978|
2978|          .IF  USERINT = 1
2978|
2978|          .IF  FULLSCC = 0
2978|          .ENDC
2978|
2978| 2F02          MOVE.L   D2,-(SP)          ;save test #
297A| 6100 075E          BSR      CLRDESK        ;clear desktop except for menu bar
297E| 6100 0800          BSR      MAKETEST        ;draw test window
2982| 241F          MOVE.L   (SP)+,D2          ;restore test #
2984| 0C02 0004          CMP.B   #4,D2          ;CPU test?
2988| 6C06          BGE.S    @1          ;skip if not
298A| 327C 1DF6          MOVEA   #CPUSTRT,A1      ;else set ptr for CPU board icon      RM000
298E| 601C          BRA.S    @4          ;and go hilite it
2990|
2990| 0C02 000A          @1      CMP.B   #10,D2          ;I/O board test?
2994| 6C06          BGE.S    @2
2996| 327C 1E12          MOVEA   #IOSTRT,A1      ;set ptr for I/O board icon      RM000
299A| 6010          BRA.S    @4
299C|
299A| 6C06          BGE.S    @31,D2          ;memory test?

```



```

29A2| 327C 1E04          MOVEA    #MEMSTRT,A1      ;set ptr for memory board icon          RM000
29A6| 6004                BRA.S     @4
29A8| 327C 1E20          @3    MOVEA    #XCRDSTRT,A1  ;else must be I/O slot card          RM000
29AC|
29AC| 6100 0BC6          @4    BSR      INVICON      ;display in test window
29B0| .ENDC
29B0|
29B0| D442                ADD      D2,D2          ;double test # for table index
29B2| 08C7 001F          BSET     #LOOP,D7      ;set loop flag
29B6| 41FA 0024          LEA      LOOP_TBL,A0      ;and jump to requested routine
29BA| D0F0 2000          ADD      0(A0,D2.W),A0
29BE| 4ED0                JMP      (A0)
29C0|
29C0| ;-----
29C0| ; Loop exit via NMI routine
29C0| ;
29C0| ;LOOPEND BTST     #1,STATREG      ;parity error?
29C0| ;      BEQ      NMI              ;skip if yes to report error
29C0| ;      MOVE.L   #STKBASE,SP      ;else restore stack
29C0| ;      BRA      LEVEL2          ;and redisplay service mode
29C0| ;-----
29C0|
29C0| ; special entry points for routines that require initial setup
29C0|
29C0| MMUTSTE1
29C0| 4A39 00FC E010      TST.B     SETUPON      ;turn on SETUP bit for MMU tests
29C6| 6000 D7E8          BRA      MMUTST      ;go do main test
29CA|
29CA| .IF FULLSCC = 0
29CA| .ENDC
29CA|
29CA| 08F9 0006 00FC C18D MEMTST3 BSET     #6,MEMCODE      ;set for extended memory test
29D2| 7002                MOVEQ    #PROFILE,D0      ;and for normal boot default (Profile)
29D4| 6100 EE78          BSR      SAV2PM      ;save in parameter memory
29D8| 6000 E434          BRA      MEMLOOP      ;and go do memory testing
29DC|
29DC| ; jump table for looping on start-up diagnostics
29DC|
29DC| D7B8                LOOP_TBL .WORD    ROMTST-LOOP_TBL      ;1 = ROM checksum test
29DE| FFE4                .WORD    MMUTSTE1-LOOP_TBL      ;2 = MMU test
29E0| E1C6                .WORD    VIDCHK-LOOP_TBL      ;3 = Video test
29E2| E380                .WORD    PARTST-LOOP_TBL      ;4 = Parity logic test
29E4| DDA4                .WORD    VIA2TST-LOOP_TBL      ;5 = Parallel port VIA test
29E6| DED4                .WORD    VIA1CHK-LOOP_TBL      ;6 = Keyboard port VIA test
29E8| DF0E                .WORD    COPSENBL-LOOP_TBL      ;7 = I/O board COPS test
29EA|
29EA| .IF FULLSCC = 1

```



```

29EA| E62C          .WORD    SCCTEST-LOOPCTL      ;8 = SCC test
29EC|              .ELSE
29EC|              .ENDC
29EC|
29EC| E730          .WORD    DSKTST-LOOPCTL      ;9 = disk controller test
29EE| E8B0          .WORD    CLKTST-LOOPCTL      ;A = clock test
29F0| FFEE          .WORD    MEMTST3-LOOPCTL     ;B = memory test
29F2| E91A          .WORD    CONFIG2-LOOPCTL     ;C = configuration check
29F4|
29F4|              .ENDC
29F4|              .PAGE
29F4|
29F4|              ;-----
29F4|              ; Display video adjust pattern
29F4|              ;-----
29F4|
29F4| 70FF          VIDAJS MOVEQ    #-1,D0          ;first erase the screen
29F6| 6100 FCFA          BSR      WRTSCRN
29FA|
29FA|              ; Next draw the horizontal lines
29FA|
29FA| 4280          CLR.L    D0          ;set scan line
29FC| 6128          BSR.S    DRWHORZ        ;go draw white line
29FE| 701B          MOVEQ    #27,D0        ;set next scan line
2A00| 721C          MOVEQ    #28,D1        ;set increment value also
2A02| 740C          MOVEQ    #12,D2        ;set line count
2A04| 6120          @1      BSR.S    DRWHORZ        ;draw some more
2A06| D041          ADD      D1,D0        ;incr to next line position
2A08| 51CA FFFA          DBF      D2,@1        ;loop until done
2A0C|
2A0C|              ; Now draw the vertical lines
2A0C|
2A0C| 4280          CLR.L    D0          ;set pixel #
2A0E| 6130          BSR.S    DRWVERT        ;draw a vertical line
2A10| 702C          MOVEQ    #44,D0        ;set next pixel position
2A12| 722D          MOVEQ    #45,D1        ;set incr value also
2A14| 740F          MOVEQ    #15,D2        ;set line count
2A16| 6128          @2      BSR.S    DRWVERT        ;draw some more
2A18| D041          ADD      D1,D0        ;incr to next pixel position
2A1A| 51CA FFFA          DBF      D2,@2        ;loop until done
2A1E|
2A1E|              ; Wait for any keystroke to terminate display
2A1E|
2A1E| 6100 FCE2          BSR      READKEY
2A22| 6000 FD18          BRA      LEVEL2        ;return to menu display
2A26|
2A26|              .PAGE
2A26|              ;-----

```

```

2A26|           ; Subroutine to draw horizontal lines. Requires inputs:
2A26|           ;   D0 = scan line (0 to 363 decimal)
2A26|           ;   $110 = base address of screen
2A26|           ;-----
2A26|
2A26| 48E7 C080      DRWHORZ MOVEM.L D0/D1/A0,-(SP) ;save regs
2A2A| 725A          MOVEQ   #90,D1                ;line length in bytes
2A2C| C0C1          MULU    D1,D0                  ;compute address offset
2A2E| 2078 0110      MOVE.L  SCRNBASE,A0          ;get base screen address
2A32| D1C0          ADDA.L  D0,A0                  ;add offset
2A34| 4218          @1    CLR.B  (A0)+              ;draw the line
2A36| 5341          SUBQ    #1,D1
2A38| 66FA          BNE.S   @1
2A3A| 4CDF 0103      MOVEM.L (SP)+,D0/D1/A0 ;restore
2A3E| 4E75          RTS
2A40|
2A40|           ;-----
2A40|           ; Subroutine to draw vertical lines. Requires inputs:
2A40|           ;   D0 = pixel position (0 to 719 decimal)
2A40|           ;   $110 = base address of screen
2A40|           ;-----
2A40|
2A40| 48E7 F080      DRWVERT MOVEM.L D0-D3/A0,-(SP) ;save regs
2A44| 7208          MOVEQ   #8,D1                  ;pixels per byte
2A46| 80C1          DIVU    D1,D0                  ;compute address offset
2A48| 4282          CLR.L   D2
2A4A| 3400          MOVE    D0,D2                  ;save offset
2A4C| 2078 0110      MOVE.L  SCRNBASE,A0          ;get base screen address
2A50| D1C2          ADDA.L  D2,A0                  ;add offset
2A52|
2A52| 4840          SWAP     D0                      ;get remainder
2A54| 9240          SUB      D0,D1                  ;compute bit position to set
2A56| 5341          SUBQ     #1,D1
2A58|
2A58| 745A          MOVEQ    #90,D2                  ;distance to next pixel
2A5A| 363C 016C      MOVE     #364,D3              ;line length
2A5E| 0390          @1    BCLR   D1,(A0)            ;draw the line
2A60| D1C2          ADDA.L  D2,A0                  ;compute next pixel to set
2A62| 5343          SUBQ    #1,D3
2A64| 66F8          BNE.S   @1                      ;loop until done
2A66| 4CDF 010F      MOVEM.L (SP)+,D0-D3/A0 ;restore
2A6A| 4E75          RTS
2A6C|
2A6C|           .IF BURNIN = 1
2A6C|           ;-----
2A6C|           ; Power cycle entry point - branches to cycling routine
2A6C|           ;-----

```

```

2A6C|
2A6C|
2A6C| 6100 066C      PowerCycle      BSR      CLRDESK      ;clear desktop
2A70| 6000 F7D2      BRA      CHKPAS2      ;go start power cycle
2A74|
2A74|
2A74|      .ENDC
2A74|
2A74|      .PAGE
2A74|
2A74| ;-----
2A74| ; Invalid input detected - beep speaker and notify user
2A74| ;-----
2A74| 6100 FC96      INVALID BSR      SQUAWK      ;that's a no no
2A78|      .IF USERINT = 0
2A78|      .ELSE
2A78| 47FA 1561      LEA      WHATMSG,A3      ;output question mark
2A7C| 6100 018C      BSR      DBOXDSPLY      ;display in dialog box
2A80| 6004      BRA.S      INVXIT
2A82|
2A82|
2A82|      LEV2LOOP
2A82| 6100 01A4      BSR      CLRDBOX      ;go remove dialog box
2A86|      INVXIT
2A86| 6100 FD18      BSR      WRTMENU      ;redisplay pull-down menu
2A8A| 6100 0582      BSR      CursorDisplay ;redisplay cursor
2A8E| 6000 FCBC      BRA      GETLEV2      ;and go get more input
2A92|      .ENDC
2A92|
2A92|      .IF ROM16K = 1
2A92|      .IF FULLSCC = 0
2A92|      .ENDC      ;{FULLSCC}
2A92|      .ENDC      ;{ROM16K}
2A92|
2A92|      .PAGE
2A92|      .IF USERINT = 0
2A92|      .ELSE
2A92| ;-----
2A92| ; Subroutine to output prompt line and gather input
2A92| ;-----
2A92|
2A92| 48E7 0620      PROMPT  MOVEM.L D5-D6/A2,-(SP) ;save screen ptrs and target address
2A96| 6100 071A      BSR      MAKEDBOX      ;make dialog box
2A9A| 3A3C 0018      MOVE     #DBOXROW,D5      ;set msg ptrs
2A9E| 3C3C 0018      MOVE     #DBOXCOL,D6
2AA2| 6100 0974      BSR      GETLENGTH      ;get message length
2AA6| 5442      ADDQ     #2,D2      ;incr for spacing
2AA8| 31C2 052E      MOVE     D2,MSGLEN      ;save as message length
2AAC| 6100 0C52      BSR      DSPMSG      ;write msg
2AB0| 6100 0008      BSR      RDINPUT      ;go handle input

```

```

2AB4| 4CDF 0460          MOVEM.L (SP)+,D5-D6/A2 ;restore and exit
2AB8| 4E75              RTS
2ABA|
2ABA| ;-----
2ABA| ; Subroutine to read keyboard input and save in buffer.  Accepts max of
2ABA| ; 48 characters.
2ABA| ;-----
2ABA|
2ABA| RDINPUT
2ABA| 207C 0000 02C0      MOVE.L #KBDBFR,A0 ;set buffer ptrs
2AC0| 2248              MOVE.L A0,A1 ;same for head and tail
2AC2| 4283              CLR.L D3 ;clear for result use
2AC4|
2AC4| 6100 FC3C          READIN BSR READKEY ;get char
2AC8| 6100 FC50          BSR KeyToASCII ;convert to ASCII
2ACC|
2ACC| 4A00              TST.B D0 ;ignore CMD, Option, Shift, Alpha lock
2ACE| 67F4              BEQ.S READIN
2AD0| 0C00 0008          CMP.B #BS,D0 ;backspace key?
2AD4| 6612              BNE.S @2
2AD6| 4A43              TST D3 ;any input
2AD8| 67EA              BEQ.S READIN ;no - ignore
2ADA| 5343              SUBQ #1,D3 ;decrement count
2ADC| 4A21              TST.B -(A1) ;delete char from queue
2ADE| 6100 008C          BSR PUTBS ;do backspace on screen
2AE2| 6100 0096          BSR CLRIT ;clear char on screen
2AE6| 60DC              BRA.S READIN ;and continue
2AE8|
2AE8| 0C00 000D          @2 CMP.B #RET,D0 ;return key?
2AEC| 6718              BEQ.S @3 ; yes - exit
2AEE|
2AEE| 0C03 0030          CMP.B #48,D3 ;at max?
2AF2| 6D06              BLT.S @4 ;skip if no
2AF4| 6100 FC16          BSR SQUAWK ;else notify user
2AF8| 60CA              BRA.S READIN ;and ignore input
2AFA|
2AFA| 5243              @4 ADDQ #1,D3 ;incr char count
2AFC| 6100 0088          BSR ENQKBD ;queue it
2B00| 6100 0C38          BSR DSPVAL ;and output it
2B04| 60BE              BRA.S READIN ;and continue read
2B06|
2B06| 4E75              @3 RTS
2B08|
2B08| ;-----
2B08| ; SCROLL - move contents of Service Window up one whole line.  Assumed
2B08| ; that we are at bottom line when called. D6 (column) and D5 (row) are
2B08| ; set to start of last line.
2B08| ;-----

```



```

2B08|
2B08| 48E7 E080          SCROLL  MOVEM.L D0-D2/A0,-(SP)          ;save data regs and bfr ptr
2B0C| 7A48              MOVEQ   #FIRSTROW+ROWLINES,D5        ;set beginning character row +1
2B0E| 7C18              MOVEQ   #FIRSTCOL,D6                  ; and beginning column
2B10| 6100 0C08          BSR     SETCRSR                      ;get address of screen
2B14| 204E              MOVE.L   A6,A0                        ;set as to ptr
2B16| D0FC 0384          ADDA     #<RBYTES*10>,A0              ;set from ptr down one character row      RM000
2B1A| 343C 001B          MOVE     #NROWS,D2                   ;number of rows to move
2B1E|
2B1E| 323C 000A          @1      MOVE     #ROWLINES,D1         ;number of pixel lines per character row
2B22| 303C 0042          @2      MOVE     #ROWLEN,D0           ;length of a pixel line in window
2B26| 3CD8              @3      MOVE     (A0)+,(A6)+           ;scroll it
2B28| 5540              SUBQ     #2,D0                        ;do entire pixel line
2B2A| 6EFA              BGT.S    @3
2B2C|
2B2C| 5245              ADDQ     #1,D5                        ;bump to next row
2B2E| 7C18              @4      MOVEQ   #FIRSTCOL,D6          ;set first col
2B30| 6100 0BE8          BSR     SETCRSR                      ;compute address
2B34| 204E              MOVE.L   A6,A0                        ;set as to ptr
2B36| D0FC 0384          ADDA     #<RBYTES*10>,A0              ;set from ptr down one character row      RM000
2B3A| 5341              SUBQ     #1,D1                        ;do all pixel lines
2B3C| 66E4              BNE.S    @2
2B3E|
2B3E| 5342              SUBQ     #1,D2                        ;finished a character row
2B40| 66DC              BNE.S    @1                          ; loop until done
2B42|
2B42| 3A3C 014C          MOVE     #LASTROW,D5                 ;peg at bottom
2B46| 3C3C 0018          MOVE     #FIRSTCOL,D6
2B4A| 4CDF 0107          MOVEM.L (SP)+,D0-D2/A0                ;restore data regs and bfr ptr
2B4E| 4E75              RTS
2B50|
2B50|                  .PAGE
2B50|
2B50|                  ; PUTLF - advance to next row; this may cause a scroll if at bottom
2B50|
2B50| 3A38 0300          PUTLF    MOVE     CRTROW,D5              ;get last state
2B54| 7C18              MOVEQ   #FIRSTCOL,D6                  ;update column to left edge of window
2B56| 0645 000A          ADD      #ROWLINES,D5                 ;bump row by number of pixel lines per row
2B5A| 0C45 014C          CMPI     #LASTROW,D5
2B5E| 6F02              BLE.S    @9                          ;skip if its ok
2B60| 61A6              BSR.S    SCROLL                      ; else, do a scroll operation
2B62| 31C5 0300          @9      MOVE     D5,CRTROW              ;save updates
2B66| 31C6 0302          MOVE     D6,CRTCOL
2B6A| 4E75              RTS
2B6C|
2B6C|                  ; PUTBS - move cursor left one position.
2B6C|                  ; Assumes location MSGLEN = left most column for window.

```

```

2B6C|
2B6C| 5346          PUTBS   SUBQ    #1,D6
2B6E| BC78 052E          CMP     MSGLEN,D6      ;stop at left edge
2B72| 6C04          BGE.S   @9
2B74| 3C38 052E          MOVE    MSGLEN,D6
2B78| 4E75          @9    RTS
2B7A|
2B7A|          ; Routine to erase data on screen
2B7A| 303C 0020      CLRIT   MOVE    #' ',D0      ; output a space
2B7E| 6100 0BBA          BSR     DSPVAL
2B82| 5346          SUBQ    #1,D6      ; reposition col ptr
2B84| 4E75          RTS              ; and that's all there is...
2B86|
2B86|          .ENDC
2B86|          .PAGE
2B86|
2B86|          ;-----
2B86|          ; Subroutine to save keyboard input in buffer - ignores data if buffer full.
2B86|          ;-----
2B86|
2B86| 2F0A          ENQKBD  MOVE.L   A2,-(SP)      ;save working reg
2B88| 347C 0300          MOVEA   #KBDEND,A2      ;get ptr to end of buffer          RM000
2B8C| B5C9          CMPA.L   A1,A2      ; at end of buffer?
2B8E| 6702          BEQ.S    @9          ; exit if yes
2B90| 12C0          MOVE.B   D0,(A1)+      ; else do save
2B92| 245F          @9    MOVE.L   (SP)+,A2      ;restore
2B94| 4E75          RTS
2B96|
2B96|          ;-----
2B96|          ; This code gets the next byte from the keyboard queue and delivers it to
2B96|          ; caller in D0.
2B96|          ;-----
2B96|
2B96| B3C8          GETCH   CMPA.L   A0,A1      ;check if any data
2B98| 6704          BEQ.S    @1          ;exit if none
2B9A| 1018          MOVE.B   (A0)+,D0      ;get data
2B9C| 6004          BRA.S    @2
2B9E| 003C 0001          @1    ORI.B   #$01,CCR      ;set empty indicator
2BA2| 4E75          @2    RTS
2BA4|
2BA4|          .PAGE
2BA4|
2BA4|          ;-----
2BA4|          ; Subroutine to get address parameter
2BA4|          ;-----
2BA4|
2BA4| 47FA 1415          GETA   LEA     ADDRMSG,A3      ;output prompt and get input
2BA8| 6100 FEE8          BSR     PROMPT
2BAC| 7208          MOVEQ   #8,D1      ;decode address (max of 8 digits)

```



```

2BAE| 6102          BSR.S  GETPARM
2BB0| 4E75          RTS
2BB2|
2BB2|          .PAGE
2BB2|          ;-----
2BB2|          ; Subroutine to get input parameters.  Reads Ascii values from keyboard
2BB2|          ; buffer and calls conversion routine to return hex values.  Stops after
2BB2|          ; reading requested input or 'space' separator encountered.
2BB2|          ; Inputs:  D1 = max chars to read
2BB2|          ; Outputs: D2 = value read
2BB2|          ;          D3 = # of chars read
2BB2|          ; Carry bit set if invalid chars.
2BB2|          ;-----
2BB2|
2BB2| 4283          GETPARM CLR.L  D3          ;use for counter
2BB4| 4282          CLR.L  D2          ;use for result
2BB6|
2BB6| 61DE          READQ  BSR      GETCH          ;check input queue
2BB8| 652E          BCS.S  GETEXIT          ;exit if no chars
2BBA| 0C00 0020    CMPI.B  #' ',D0          ;space separator?
2BBE| 6604          BNE.S  @3          ;if not, go response
2BC0| 4A20          TST.B  -(A0)          ;replace on queue
2BC2| 6024          BRA.S  GETEXIT          ;and exit
2BC4|
2BC4| 0C00 0030    @3      CMPI.B  #'0',D0          ;check if valid hex char
2BC8| 6D24          BLT.S  INVPARM
2BCA| 0C00 0039    CMPI.B  #'9',D0
2BCE| 630C          BLS.S  OKCH          ;OK if 0-9
2BD0| 0C00 0041    CMPI.B  #'A',D0          ; or A-F
2BD4| 6D18          BLT.S  INVPARM
2BD6| 0C00 0046    CMPI.B  #'F',D0
2BDA| 6E12          BGT.S  INVPARM
2BDC|
2BDC| 6116          OKCH   BSR.S  CONVERT          ;convert to hex digit
2BDE| E98A          LSL.L  #4,D2          ;save char
2BE0| 8400          OR.B   D0,D2
2BE2| 5243          ADDQ   #1,D3          ;bump counter
2BE4| B203          CMP.B  D3,D1          ;at max?
2BE6| 66CE          BNE.S  READQ          ;continue if no
2BE8|
2BE8| 023C 00FE      GETEXIT ANDI.B  #$FE,CCR          ;clear error indicator
2BEC| 6004          BRA.S  GETXIT2          ;and exit
2BEE|
2BEE| 003C 0001      INVPARM ORI.B  #$01,CCR          ;set error indicator
2BF2|
2BF2| 4E75          GETXIT2 RTS
2BF4|

```




```

2BF4| ;-----
2BF4| ; Subroutine to convert Ascii character to hex. Expects input in D0
2BF4| ; and returns converted value in D0.
2BF4| ;-----
2BF4|
2BF4| 0C00 0040      CONVERT CMP.B   #$40,D0      ;check if number or letter
2BF8| 6E06          BGT.S    @1          ;skip if letter
2BFA| 0400 0030      SUBI.B   #$30,D0      ;simple operation for number
2BFE| 6008          BRA.S    @9
2C00|
2C00| 0400 0041      @1      SUBI.B   #$41,D0      ;a little different for letters
2C04| 0600 000A      ADDI.B   #$0A,D0
2C08|
2C08| 4E75          @9      RTS
2C0A|
2C0A| .PAGE
2C0A| .IF USERINT = 1
2C0A|
2C0A| ;-----
2C0A| ; Subroutine to write to dialog box
2C0A| ;-----
2C0A|
2C0A| DBOXDSPLY
2C0A| 48E7 8600      MOVEM.L D0/D5-D6,-(SP) ;save D0 and current cursor ptrs
2C0E| 6100 05A2      BSR      MAKEDBOX ;clear dialog box and redraw
2C12| 3A3C 0018      MOVE     #DBOXROW,D5 ;set box coordinates
2C16| 3C3C 0018      MOVE     #DBOXCOL,D6
2C1A| 6100 0AFE      BSR      SETCRSR ;get address in A6
2C1E| 6100 0AE0      BSR      DSPMSG ;write msg
2C22| 4CDF 0061      MOVEM.L (SP)+,D0/D5-D6 ;restore
2C26| 4E75          RTS ;and exit
2C28|
2C28| ;-----
2C28| ; Subroutine to remove dialog box from screen
2C28| ;-----
2C28|
2C28| 705A          CLRDBOX MOVEQ    #ROWBYTES,D0 ;set pixel line length
2C2A| 3C7C 05FA          MOVEA    #DESKLINE,A6 ;set starting point as bottom of menu line
2C2E| 4281          CLR.L    D1
2C30| 2A4E          MOVE.L    A6,A5 ;set limit as bottom of dialog box
2C32| 223C 0000 0708      MOVE.L    #<DBOXHIGH*ROWBYTES>,D1 ; by adding box height to start pt
2C38| 0681 0000 0168      ADD.L    #DBOXTOP,D1
2C3E| DBC1          ADDA.L    D1,A5
2C40| 6100 04AC      BSR      GRAY ;redraw gray pattern
2C44| 4E75          RTS
2C46|
2C46| ;-----
2C46| ; GETINPUT routine - waits for inputs from mouse or keyboard and returns
2C46| ; with keycode in D0 if keyboard input, or rectangle ID if active rect is

```

```

2C46|                ; selected with the mouse.  If CMD flag is set, keyboard input returned
2C46|                ; only when prefaced by the CMD key.
2C46|                ;-----
2C46|
2C46|                ; State 1 - General wait
2C46|
2C46|                GETINPUT
2C46| 0838 0003 02A2          BTST    #CMDFLG,STATFLGS ;command key still down?
2C4C| 666C                  BNE.S   GET2            ;skip if yes
2C4E| 6100 00E8          GET1    BSR     WT4INPUT      ;else go wait for COPS input
2C52|                CHKIT
2C52| 0C00 0006          CMP.B   #MOUSUP,D0          ;mouse button up?
2C56| 6608                  BNE.S   @1
2C58| 08B8 0004 02A2          BCLR   #MOUSE,STATFLGS ;clear mouse flag
2C5E| 60EE                  BRA.S   GET1            ;and go wait for more input
2C60|
2C60| 0C00 0086          @1      CMP.B   #MOUSDWN,D0        ;mouse button down?
2C64| 6608                  BNE.S   @2
2C66| 08F8 0004 02A2          BSET    #MOUSE,STATFLGS ;set reminder flag
2C6C| 6014                  BRA.S   @3
2C6E|
2C6E| 4A00                @2      TST.B   D0              ;mouse data?
2C70| 661A                  BNE.S   @5
2C72| 6100 0376          BSR     CursorHide          ;else clear old cursor
2C76| 6100 0396          BSR     CursorDisplay        ;and redisplay cursor in new position
2C7A|
2C7A| 0838 0004 02A2          BTST    #MOUSE,STATFLGS ;is mouse button down?
2C80| 67CC                  BEQ.S   GET1            ;skip if not
2C82|
2C82| 6100 01C2          @3      BSR     CHKPOSN          ;else go check mouse position
2C86| 67C6                  BEQ.S   GET1            ;continue if not over a rect
2C88| 6000 0082          BRA      GET3            ;else go to state 3
2C8C|
2C8C| 0C00 00FF          @5      CMP.B   #CmdDwn,D0          ;command key down?
2C90| 6608                  BNE.S   @6
2C92| 08F8 0003 02A2          BSET    #CMDFLG,STATFLGS ;set flag if yes
2C98| 6020                  BRA.S   GET2            ;and go to state 2
2C9A|
2C9A| 0C00 007F          @6      CMP.B   #CmdUp,D0           ;command key up?
2C9E| 6608                  BNE.S   @4
2CA0| 08B8 0003 02A2          BCLR   #CMDFLG,STATFLGS ;clear flag if yes
2CA6| 60A6                  BRA.S   GET1            ;and continue in loop
2CA8| 0838 0005 02A2          @4      BTST    #CHKCMD,STATFLGS ;CMD key prefix required?
2CAE| 669E                  BNE.S   GET1            ;loop if yes
2CB0| 4A00                @7      TST.B   D0              ;else check if downstroke
2CB2| 6A92                  BPL.S   GETINPUT        ;skip upstrokes
2CB4| 6100 01EC          BSR     CHKINPUT        ;go check if rectangle selected

```



```

2CB8| 4E75                      RTS                      ;and return with keycode
2CBA|
2CBA|                          ; State 2 - Command (apple) button down
2CBA|
2CBA|      GET2
2CBA| 6100 007C      WAIT2  BSR      WT4INPUT      ;else go wait for COPS input
2CBE|
2CBE| 0C00 007F      CHKIT2  CMP.B      #CMDUP,D0      ;command key up?
2CC2| 6608                      BNE.S      @2      ;skip if not
2CC4| 08B8 0003 02A2      BCLR      #CMDFLG,STATFLGS ;else clear flag
2CCA| 6082                      BRA.S      GET1      ;and return to state 1
2CCC|
2CCC| 0C00 0006      @2      CMP.B      #MOUSUP,D0      ;mouse button up?
2CD0| 6608                      BNE.S      @3      ;skip if not
2CD2| 08B8 0004 02A2      BCLR      #MOUSE,STATFLGS ;clear mouse flag
2CD8| 60E0                      BRA.S      GET2      ;and go wait for more input
2CDA|
2CDA| 0C00 0086      @3      CMP.B      #MOUSDWN,D0     ;mouse button down?
2CDE| 6608                      BNE.S      @4      ;skip if not
2CE0| 08F8 0004 02A2      BSET      #MOUSE,STATFLGS ;set reminder flag
2CE6| 6014                      BRA.S      @5
2CE8|
2CE8| 4A00      @4      TST.B      D0      ;mouse data?
2CEA| 6618                      BNE.S      @6      ;skip if not
2CEC| 6100 02FC      BSR      CursorHide      ;else clear old cursor
2CF0| 6100 031C      BSR      CursorDisplay     ;and redisplay cursor in new position
2CF4|
2CF4| 0838 0004 02A2      BTST      #MOUSE,STATFLGS ;is mouse button down?
2CFA| 67BE                      BEQ.S      GET2      ;skip if not
2CFC|
2CFC| 6100 0148      @5      BSR      CHKPOSN      ;else go check mouse position
2D00| 67B8                      BEQ.S      GET2      ;continue if not over a rect
2D02| 6008                      BRA.S      GET3      ;else go to state 3
2D04|
2D04| 6AB4      @6      BPL.S      WAIT2      ;ignore upstrokes
2D06| 6100 019A      BSR      CHKINPUT      ;go check if rectangle selected
2D0A| 4E75                      RTS      ;and return with keycode
2D0C|
2D0C|                          ; State 3 - Mouse button down and over an active rectangle
2D0C|
2D0C| 3200      GET3      MOVE      D0,D1      ;save rectangle ID
2D0E| 6100 0028      WAIT3  BSR      WT4INPUT      ;go wait for input
2D12|
2D12| 0C00 0006                      CMP.B      #MOUSUP,D0      ;mouse button up?
2D16| 660A                      BNE.S      @1      ;skip if not
2D18| 08B8 0004 02A2      BCLR      #MOUSE,STATFLGS ;clear indicator
2D1E| 3001                      MOVE      D1,D0      ;restore rectangle ID

```

```

2D20| 4E75                RTS                ;and go analyze input
2D22|
2D22| 4A00                @1      TST.B   D0                ;mouse data?
2D24| 66E8                BNE.S   WAIT3                ;continue wait if not - ignore keyboard input
2D26|
2D26| 6100 02C2            @2      BSR     CursorHide        ;move cursor to new position
2D2A| 6100 02E2            BSR     CursorDisplay
2D2E| 6100 0116            BSR     CHKPOSN        ;check if over a rect
2D32| 66D8                BNE.S   GET3                ;stay in this state if yes
2D34| 6000 FF10            BRA     GETINPUT        ;else return to state 1
2D38|
2D38|                      ;-----
2D38|                      ;  WT4INPUT
2D38|                      ;
2D38|                      ;      Routine to wait for input from COPS.  Returns with keycode in D0
2D38|                      ;      or sets D0 = 0 if mouse data received.
2D38|                      ;
2D38|                      ;-----
2D38|
2D38| WT4INPUT
2D38|
2D38|                      ;  State 0 - general wait
2D38|
2D38| 6100 0084            COPS0    BSR     ReadCOPS        ;get input from COPS
2D3C| 4A00                TST.B   D0                ;mouse data?
2D3E| 6708                BEQ.S   COPS1                ;go to state 1 if yes
2D40| 0C00 0080            CMP.B   #RSTCODE,D0        ;reset code?
2D44| 6716                BEQ.S   COPS2                ;skip to state 2 if yes
2D46| 4E75                RTS                ;else return with the keycode
2D48|
2D48|                      ;  State 1 - waiting for mouse data
2D48| 6174                COPS1    BSR.S   ReadCOPS        ;get COPS input
2D4A| 11C0 048A            MOVE.B   D0,MousDx        ;save mouse delta-x
2D4E| 616E                BSR.S   ReadCOPS        ;read and
2D50| 11C0 048B            MOVE.B   D0,MousDy        ;save mouse delta-y
2D54| 6100 01D4            BSR     MouseMovement        ;record the mouse movement
2D58| 4240                CLR     D0                ;set mouse flag
2D5A| 4E75                RTS                ;and exit
2D5C|
2D5C|                      ;  State 2 - waiting for reset code
2D5C|
2D5C| 6160                COPS2    BSR.S   ReadCOPS        ;get COPS input
2D5E| 0C00 00DF            CMP.B   #$DF,D0        ;reset code <= $DF?
2D62| 6318                BLS.S   @1                ;branch if yes
2D64| 0C00 00EF            CMP.B   #$EF,D0        ;reset code <= $EF?
2D68| 6318                BLS.S   @2                ;skip if yes
2D6A| 0C00 00FB            CMP.B   #$FB,D0        ;reset code <= $FB

```

```

2D6E| 6500 0018          BLO.S  @3          ;branch if < $FB
2D72| 6716          BEQ.S  @4          ;branch if = $FB
2D74| 0C00 00FD          CMP.B  #$FD,D0      ;reset code <= $FD?
2D78| 630E          BLS.S  @3          ;branch if <= $FD
2D7A| 6010          BRA.S  @5          ;branch if > $FD
2D7C|
2D7C|          ; $00 - $DF  Keyboard ID number - save and return to state 0
2D7C| 11C0 01B2      @1      MOVE.B  D0,KeyID      ;save new ID
2D80| 60B6          BRA.S  COPS0      ;return to general wait
2D82|
2D82|          ; $E0 - $EF  Clock data - save first nibble, and go to state 4
2D82|
2D82| 11C0 0480      @2      MOVE.B  D0,ClockBytes    ;save it
2D86| 6016          BRA.S  COPS4      ;and go get rest of data
2D88|
2D88|          ; $F0 - $FA  Reserved --- ignored
2D88|          ; $FC - $FD  Clock timer interrupt, keyboard unplugged --- ignored
2D88|
2D88| 60AE          @3      BRA.S  COPS0      ;go back to general wait
2D8A|
2D8A|          ; $FB          Soft on/off button - go to power-off routine
2D8A|
2D8A| 604C          @4      BRA.S  PowerOff      ;go shutdown the system
2D8C|
2D8C|          ; $FE - $FF  COPS failure codes - go to error routine
2D8C| 0C00 00FE      @5      CMP.B  #$FE,D0      ;I/O board COPS?
2D90| 6604          BNE.S  @6          ;
2D92| 7034          MOVEQ  #EIOCOP,D0      ;else set I/O COPS error code
2D94| 6002          BRA.S  @7          ;
2D96| 7035          @6      MOVEQ  #EKBCOP,D0      ;set keyboard COPS error code
2D98| 003C 0001      @7      ORI.B  #$01,CCR      ;set return error indicator
2D9C| 4E75          RTS
2D9E|
2D9E|          ; State 4 - waiting for clock data; use timeout routine to guard against error
2D9E|
2D9E| 48E7 60E0      COPS4    MOVEM.L D1-D2/A0-A2,-(SP) ;save regs
2DA2| 43F8 0481      LEA      ClockBytes+1,A1    ;set ptrs to save area
2DA6| 45F8 0486      LEA      ClockBytes+6,A2
2DAA| 7205          MOVEQ  #5,D1          ;5 more bytes expected
2DAC| 6100 DCD0      @1      BSR      GETDATA      ;get COPS data
2DB0| 6504          BCS.S  @2          ;skip if any errors
2DB2| 5341          SUBQ   #1,D1          ;loop until done
2DB4| 66F6          BNE.S  @1          ;
2DB6|
2DB6| 4CDF 0706      @2      MOVEM.L (SP)+,D1-D2/A0-A2 ;restore regs and
2DBA| 6000 FF7C      BRA.S  COPS0      ;go back to general wait
2DBE|

```



```

2DBE|                                     .ENDC                                     ;{USERINT}
2DBE|
2DBE|                                     ;-----
2DBE|                                     ; ReadCOPS
2DBE|                                     ;
2DBE|                                     ; Routine to get data from COPS. Returns with data in D0.
2DBE|                                     ;
2DBE|                                     ;-----
2DBE|
2DBE| ReadCOPS
2DBE| 2F08 MOVE.L A0,-(SP)
2DC0| 207C 00FC DD81 MOVE.L #VIA1BASE,A0 ;get interface ptr
2DC6| 1028 001A @1 MOVE.B IFR1(A0),D0 ;poll for data
2DCA| 0800 0001 BTST #1,D0
2DCE| 67F6 BEQ.S @1 ;loop until data received
2DD0| 1028 0002 MOVE.B ORA1(A0),D0 ;read
2DD4| 205F MOVE.L (SP)+,A0 ;and return
2DD6| 4E75 RTS
2DD8|
2DD8| .IF USERINT = 1
2DD8|                                     ;-----
2DD8|                                     ; PowerOff - routine to shutdown the system
2DD8|                                     ;-----
2DD8|
2DD8| PowerOff
2DD8| 0807 0011 BTST #DISK,D7 ;disk controller error? CHG023
2DDC| 6622 BNE.S @9 ;skip if yes CHG023
2DDE| 207C 00FC C001 MOVE.L #DISKMEM,A0 ;set ptr for shared memory
2DE4| 6144 BSR.S ENBLDRVS ;enable both drives
2DE6| 4228 0004 MOVE.B #DRV1,DRV(A0) ;eject diskette in drive 1
2DEA| 6100 F06A BSR EJCTDSK
2DEE| 117C 0080 0004 MOVE.B #DRV2,DRV(A0) ;and drive 2
2DF4| 6100 F060 BSR EJCTDSK
2DF8|
2DF8| 10BC 0089 MOVE.B #DIE,(A0) ;get Twiggy controller out of memory
2DFC| 6100 F006 BSR CMDCHK ;wait until command taken
2E00| @9 BSR4 CONOFF ;turn off contrast CHG003
2E00| 49FA 0006 # LEA @1,A4
2E04| 6000 D9FA # BRA CONOFF
2E08| #@1
2E08| 203C 0003 D090 MOVE.L #ONESEC,D0 ;wait for it to happen CHG003
2E0E| 6100 DCD2 BSR DELAY ; CHG003
2E12| 7021 MOVEQ #$21,D0 ;power off, timer off, clock on RM000
2E14| 6100 DB40 BSR COPSCMD ;go do it
2E18| 6100 DCC2 BSR KBDDELAY ;wait about 1.7 secs for power-off
2E1C|
2E1C| 6100 F8EE BSR SQUAWK ;error if still on

```



```

2E20| 45FA 0B29          LEA      IOBRD,A2
2E24| 7034              MOVEQ    #EIOCOP,D0      ;set error code
2E26| 6000 E798          BRA      TSTXIT        ;display error and go back to levell monitor
2E2A|
2E2A|                  ;-----
2E2A|                  ; Subroutine to enable drives
2E2A|                  ;-----
2E2A|
2E2A| ENBLDRVS
2E2A| 117C 0088 0002      MOVE.B    #$88,CMD(A0)      ;enable both drives
2E30| 10BC 0086          MOVE.B    #ENBLINT,(A0)
2E34| 6100 EFCE          BSR      CMDCHK
2E38| 267C 00FC DD81      MOVE.L    #VIA1BASE,A3      ;and enable FDIR
2E3E| 08AB 0004 0004      BCLR     #FDIR,DDRB1(A3)
2E44| 4E75              RTS
2E46|
2E46|                  ;-----
2E46|                  ; CHKPOSN
2E46|                  ;
2E46|                  ; Routine to check mouse position versus active rectangle table.
2E46|                  ; If over a rectangle, inverts it and returns with its ID.
2E46|                  ; If not over a rectangle, ensures all rectangles not inverted, and
2E46|                  ; returns with D0 = 0.
2E46|                  ;
2E46|                  ; Active rectangle table has following format:
2E46|                  ;
2E46|                  ; Word 1 : number of entries in table
2E46|                  ; Word 2 : ID for first entry, MSB = 1 if inverted on screen
2E46|                  ; Next 4 words contain X,Y pixel coordinates for upper left
2E46|                  ; and bottom right corners
2E46|                  ; Each successive entry follows same format, with 5 words per entry
2E46|                  ;
2E46|                  ; Register usage:
2E46|                  ; A0 = ptr to rectangle table
2E46|                  ; D0 = # of entries in table
2E46|                  ; D1 = ID for current entry
2E46|                  ; D2 = X-coordinate for upper left
2E46|                  ; D3 = Y-coordinate for upper left
2E46|                  ; D4 = X-coordinate for bottom right
2E46|                  ; D5 = Y-coordinate for bottom right
2E46|                  ; D6 = X-coordinate for current cursor location
2E46|                  ; D7 = Y-coordinate for current cursor location
2E46|                  ;
2E46|                  ; On exit, D0 = ID code of rectangle cursor is over or
2E46|                  ; = 0 if not over any rectangle
2E46|                  ;-----
2E46|

```



```

2E46|
2E46| 48E7 7F80      CHKPOSN MOVEM.L D1-D7/A0,-(SP)
2E4A| 3C38 0496      MOVE      CrsrX,D6          ;get current cursor location
2E4E| 3E38 0498      MOVE      CrsrY,D7
2E52| 41F8 053A      LEA       RectTable,A0      ;set ptr to table
2E56| 3018          MOVE      (A0)+,D0        ;get count
2E58| 6742          BEQ.S     CHKPXIT          ;exit if 0
2E5A| 4C98 003E      GETNTRY MOVEM      (A0)+,D1-D5      ;else get entry (5 words)
2E5E|
2E5E|              ; check if cursor over rectangle
2E5E|
2E5E|              CMP       D2,D6              ;CrsrX < upper left X?
2E60| BC42          BLT.S     @1              ;branch if cursor to left of rectangle
2E62| BC44          CMP       D4,D6              ;CrsrX > bottom right X?
2E64| 6E0A          BGT.S     @1              ;branch if cursor to right of rectangle
2E66| BE43          CMP       D3,D7              ;CrsrY < upper left Y?
2E68| 6D06          BLT.S     @1              ;branch if cursor above rectangle
2E6A| BE45          CMP       D5,D7              ;CrsrY > bottom right Y?
2E6C| 6E02          BGT.S     @1              ;branch if cursor below rectangle
2E6E| 600E          BRA.S     @3              ;cursor over this entry - go invert it
2E70|
2E70|              ; not over this entry - check if inverted, then continue through table
2E70| 4A41          @1      TST      D1              ;entry inverted?
2E74| 6100 004C      BSR      INVERT          ;else go reinvert
2E78| 5340          @2      SUBQ     #1,D0          ;decrement entry count
2E7A| 66DE          BNE.S     GETNTRY          ;check next entry if not done
2E7C| 601E          BRA.S     CHKPXIT          ;else exit with D0 = 0
2E7E|
2E7E|              ; over the rectangle - if not already, invert it and data saved
2E7E|
2E7E|              @3      TST      D1              ;already inverted?
2E80| 6B18          BMI.S     @6              ;exit if yes
2E82| 6100 003E      BSR      INVERT          ;go invert rectangle
2E86|
2E86|              ; check if any other entries previously inverted
2E86|
2E86|              @4      SUBQ     #1,D0          ;done?
2E88| 6710          BEQ.S     @6              ;skip if yes
2E8A| 4A58          TST      (A0)+          ;else check next entry
2E8C| 6B04          BMI.S     @5              ;skip if inverted
2E8E| 5048          ADDQ     #8,A0          ;else bump to next entry
2E90| 60F4          BRA.S     @4              ;and continue loop
2E92|
2E92| 4C98 003C      @5      MOVEM      (A0)+,D2-D5      ;get coordinates
2E96| 6100 002A      BSR      INVERT          ;and go reinvert and then exit
2E9A|              ;since at most one rect inverted
2E9A|

```



```

2E9A| 3001          @6      MOVE    D1,D0          ;set return code
2E9C|
2E9C| 4CDF 01FE      CHKPXIT MOVEM.L (SP)+,D1-D7/A0
2EA0| 4E75          RTS                      ;and exit
2EA2|
2EA2| ;-----
2EA2| ;  CHKINPUT
2EA2| ;
2EA2| ;  Routine to check keyboard input versus active rectangle table.
2EA2| ;  If rectangle selected, inverts it and returns.
2EA2| ;
2EA2| ;  Active rectangle table has following format:
2EA2| ;
2EA2| ;      Word 1 : number of entries in table
2EA2| ;      Word 2 : ID for first entry, MSB = 1 if inverted on screen
2EA2| ;      Next 4 words contain X,Y pixel coordinates for upper left
2EA2| ;              and bottom right corners
2EA2| ;      Each successive entry follows same format, with 5 words per entry
2EA2| ;
2EA2| ;      Register usage:
2EA2| ;          A0 = ptr to rectangle table
2EA2| ;          D0 = keyboard input
2EA2| ;          D1 = ID for current entry
2EA2| ;          D2 = X-coordinate for upper left
2EA2| ;          D3 = Y-coordinate for upper left
2EA2| ;          D4 = X-coordinate for bottom right
2EA2| ;          D5 = Y-coordinate for bottom right
2EA2| ;          D6 = # of entries in table
2EA2| ;-----
2EA2|
2EA2| CHKINPUT
2EA2| 48E7 7E80      MOVEM.L D1-D6/A0,-(SP)
2EA6| 41F8 053A      LEA      RectTable,A0      ;set ptr to table
2EAA| 3C18          MOVE     (A0)+,D6          ;get count
2EAC| 4C98 003E      RENTRY  MOVEM (A0)+,D1-D5    ;get entry (5 words)
2EB0| B200          CMP.B   D0,D1          ;match with keyboard input?
2EB2| 6706          BEQ.S   @1              ;skip if yes
2EB4| 5346          SUBQ    #1,D6          ;else loop thru all entries
2EB6| 66F4          BNE.S   RENTRY
2EB8| 6002          BRA.S   @2              ;skip to exit
2EBA|
2EBA| 6106          @1      BSR.S   INVERT      ;go invert rectangle
2EBC|
2EBC| 4CDF 017E      @2      MOVEM.L (SP)+,D1-D6/A0 ;restore regs
2EC0| 4E75          RTS                      ;and return
2EC2| ;-----

```

```

2EC2|          ; INVERT
2EC2|          ;
2EC2|          ; Routine to invert buttons, menu or icon. Computes upper left address,
2EC2|          ; width and height of rectangle, then calls INVERSE and other routines
2EC2|          ; as needed.
2EC2|          ;
2EC2|          ; Register inputs: D2 = upper left X-coordinate
2EC2|          ;                      D3 = upper left Y-coordinate
2EC2|          ;                      D4 = bottom right X-coordinate
2EC2|          ;                      D5 = bottom right Y-coordinate
2EC2|          ;                      A0 = ptr to start of next entry in rectangle table
2EC2|          ;
2EC2|          ;-----
2EC2|
2EC2| 48E7 F840      INVERT  MOVEM.L D0-D4/A1,-(SP)
2EC6|
2EC6| 6100 0122      BSR      CursorHide      ;remove cursor
2ECA|
2ECA| 0868 0007 FFF6  BCHG      #7,-10(A0)      ;flip the invert bit indicator for entry
2ED0|
2ED0| 3004          MOVE      D4,D0          ;compute width
2ED2| 9042          SUB       D2,D0
2ED4| 80FC 0008      DIVU      #8,D0          ;convert to bytes
2ED8| 3205          MOVE      D5,D1          ;compute height in pixel rows
2EDA| 9243          SUB       D3,D1
2EDC| 5241          ADDQ      #1,D1          ;bump by 1 to do bottom line also
2EDE|
2EDE| 93C9          SUBA.L     A1,A1          ;use for upper left address
2EE0| E64A          LSR       #3,D2          ;divide pixel column by 8 for bytes
2EE2| D2C2          ADD       D2,A1          ;add to address
2EE4| 785A          MOVEQ     #MaxX/8,D4      ;bytes per row on screen
2EE6| C6C4          MULU      D4,D3          ; * pixel row
2EE8| D2C3          ADD       D3,A1          ;address of upper left corner
2EEA|
2EEA| 0838 0007 02A2  BTST      #MENU,STATFLGS ;doing menu item?
2EF0| 6706          BEQ.S     @0              ;skip if not
2EF2| D2FC 005A      ADDA      #ROWBYTES,A1    ;else add 1 scan line to avoid menu      RM000
2EF6|
2EF6| 5341          SUBQ       #1,D1          ;and decr length to avoid inverting bottom line
2EF8|
2EF8|
2EF8| 74FF          @0      MOVEQ     #-1,D2      ;set fill pattern
2EFA| 48E7 E040      MOVEM.L     D0-D2/A1,-(SP) ;save args
2EFE| 6100 022E      BSR       INVERSE      ;invert it
2F02|
2F02| 4CDF 0207      MOVEM.L     (SP)+,D0-D2/A1 ;restore args
2F06|

```

```

2F06| 0838 0006 02A2          BTST    #BTN,STATFLGS    ;doing buttons?
2F0C| 6706                    BEQ.S    @1              ;skip if not
2F0E| 6100 03E8              BSR      DRAWBTN        ;redraw button
2F12| 600C                    BRA.S    @9              ;and exit
2F14|
2F14| 0838 0007 02A2          @1      BTST    #MENU,STATFLGS ;doing menu?
2F1A| 6704                    BEQ.S    @9              ;skip if not
2F1C| 6100 0424              BSR      DRAWSIDES      ;just redraw sides
2F20|
2F20| 6100 00EC              @9      BSR      CursorDisplay ;redisplay cursor
2F24|
2F24| 4CDF 021F              MOVEM.L (SP)+,D0-D4/A1 ;restore regs
2F28| 4E75                    RTS              ;and return
2F2A|
2F2A|                        .PAGE
2F2A|                        ;-----
2F2A|                        ;
2F2A|                        ;   Hardware Interface for the Mouse
2F2A|                        ;
2F2A|                        ;   Written by Rick Meyers
2F2A|                        ;   (c) Apple Computer Incorporated, 1983
2F2A|                        ;
2F2A|                        ;   The routines below provide an assembly language interface to the mouse.
2F2A|                        ;   Input parameters are passed in registers, output parameters are returned
2F2A|                        ;   in registers. Unless otherwise noted, all registers are preserved.
2F2A|                        ;
2F2A|                        ;   The Mouse
2F2A|                        ;
2F2A|                        ;   The mouse is a pointing device used to indicate screen locations. Mouse
2F2A|                        ;   coordinates are located between pixels on the screen. Therefore, the
2F2A|                        ;   X-coordinate can range from 0 to 720, and the Y-coordinate from 0 to 364.
2F2A|                        ;   The initial mouse location is 0,0.
2F2A|                        ;
2F2A|                        ;   Mouse Scaling
2F2A|                        ;
2F2A|                        ;   The relationship between physical mouse movements and logical mouse
2F2A|                        ;   movements is not necessary a fixed linear mapping. Three alternatives
2F2A|                        ;   are available: 1) unscaled, 2) scaled for fine movement and 3) scaled
2F2A|                        ;   for coarse movement.
2F2A|                        ;
2F2A|                        ;   When mouse movement is unscaled, a horizontal mouse movement of x units
2F2A|                        ;   yields a change in the mouse X-coordinate of x pixels. Similarly, a
2F2A|                        ;   vertical movement of y units yields a change in the mouse Y-coordinate
2F2A|                        ;   of y pixels. These rules apply independent of the speed of the mouse
2F2A|                        ;   movement.
2F2A|                        ;
2F2A|                        ;   When mouse movement is scaled, horizontal movements are magnified by 3/2

```

```

2F2A|           ; relative to vertical movements. This is intended to compensate for the
2F2A|           ; 2/3 aspect ratio of pixels on the screen. When scaling is in effect, a
2F2A|           ; distinction is made between fine (small) movements and coarse (large)
2F2A|           ; movements. Fine movements are slightly reduced, while coarse movements
2F2A|           ; are magnified. For scaled fine movements, a horizontal mouse movement of
2F2A|           ; x units yields a change in the X-coordinate of x pixels, but a vertical
2F2A|           ; movement of y units yields a change of (2/3)*y pixels. For scaled coarse
2F2A|           ; movements, a horizontal movement a x units yields a change of (3/2)*x
2F2A|           ; pixels, while a vertical movements of y units yields a change of y pixels.
2F2A|           ;
2F2A|           ; The distinction between fine movements and coarse movements is determined
2F2A|           ; by the sum of the x and y movements each time the mouse location is
2F2A|           ; updated. If this sum is at or below the 'threshold', the movement is
2F2A|           ; considered to be a fine movement. Values of the threshold range from 0
2F2A|           ; (which yields all coarse movements) to 256 (which yields all fine
2F2A|           ; movements). Given the default mouse updating frequency, a threshold of
2F2A|           ; about 8 (threshold's initial setting) gives a comfortable transition between
2F2A|           ; fine and coarse movements.
2F2A|           ;-----
2F2A|
2F2A|           ;-----
2F2A|           ;
2F2A|           ; Mouse Movement
2F2A|           ;
2F2A|           ; This routine is called by the GETINPUT routine when the COPS has
2F2A|           ; reported mouse movement. All registers are preserved.
2F2A|           ;
2F2A|           ;
2F2A|           ; Register Assignments:
2F2A|           ;
2F2A|           ;     D0 -- Mouse X-Coordinate (integer)
2F2A|           ;     D1 -- Mouse Y-Corrdinate (integer)
2F2A|           ;     D2 -- Mouse Dx (integer)
2F2A|           ;     D3 -- Mouse Dy (integer)
2F2A|           ;
2F2A| 48E7 7C00      MouseMovement  MOVEM.L D1-D5,-(SP)          ; save registers
2F2E| 3038 0486      MOVE.W  MousX,D0          ; mouse X-coordinate
2F32| 3238 0488      MOVE.W  MousY,D1          ; mouse Y-coordinate
2F36| 1438 048A      MOVE.B  MousDx,D2          ; mouse Dx (byte)
2F3A| 4882          EXT.W  D2                  ; mouse Dx (integer)
2F3C| 1638 048B      MOVE.B  MousDy,D3          ; mouse Dy (byte)
2F40| 4883          EXT.W  D3                  ; mouse Dy (integer)
2F42|
2F42| 3802          Scale      MOVE.W  D2,D4          ; mouse Dx
2F44| 6C02          BGE.S   @1                  ; branch if >= 0
2F46| 4444          NEG.W   D4                  ; ABS(mouse Dx)

```



```

2F48|
2F48| 3A03                @1          MOVE.W  D3,D5          ; mouse Dy
2F4A| 6C02                BGE.S   @2          ; branch if >= 0
2F4C| 4445                NEG.W   D5          ; ABS(mouse Dy)
2F4E|
2F4E| D845                @2          ADD.W   D5,D4          ; ABS(Dx) + ABS(Dy)
2F50| 9878 048E          SUB.W   MousThresh,D4 ; - MouseThreshold
2F54| 6E16                BGT.S   Coarse        ; branch if coarse movement
2F56|
2F56| D042                Fine        ADD.W   D2,D0          ; new X-coordinate (scale 1)
2F58| 3403                MOVE.W  D3,D2          ; save Dy
2F5A| D643                ADD.W   D3,D3          ; Dy*2
2F5C| D643                ADD.W   D3,D3          ; Dy*4
2F5E| D642                ADD.W   D2,D3          ; Dy*5
2F60| 5443                ADDQ   #2,D3          ; (Dy*5)+2
2F62| 6D02                BLT.S   @3          ; branch if negative
2F64| 5643                ADDQ   #3,D3          ; (Dy*5)+5
2F66| E643                @3          ASR.W   #3,D3          ; Dy*(5/8) with rounding
2F68| D243                ADD.W   D3,D1          ; new Y-coordinate (scale 5/8)
2F6A| 6010                BRA.S   Bounds        ; continue
2F6C|
2F6C| D243                Coarse      ADD.W   D3,D1          ; new Y-coordinate (scale 1)
2F6E| 3602                MOVE.W  D2,D3          ; save Dx
2F70| D443                ADD.W   D3,D2          ; Dx*2
2F72| D443                ADD.W   D3,D2          ; Dx*3
2F74| 6D02                BLT.S   @4          ; branch if negative
2F76| 5242                ADDQ.W #1,D2          ; (Dx*3)+1
2F78| E242                @4          ASR.W   #1,D2          ; Dx*(3/2) with rounding
2F7A| D042                ADD.W   D2,D0          ; new X-coordinate (scale 3/2)
2F7C|
2F7C| 4A40                Bounds      TST.W   D0          ; new X-coordinate >= 0
2F7E| 6C02                BGE.S   @5          ; branch if >= 0
2F80| 4240                MOVE.W  #0,D0         ; minimum X of 0
2F82|
2F82| 0C40 02D0            @5          CMP.W   #MaxX,D0         ; new X-coordinate <= 720
2F86| 6F04                BLE.S   @6          ; branch if <= 720
2F88| 303C 02D0            MOVE.W  #MaxX,D0         ; maximum X of 720
2F8C|
2F8C| 4A41                @6          TST.W   D1          ; new Y-coordinate >= 0
2F8E| 6C02                BGE.S   @7          ; branch if >= 0
2F90| 4241                MOVE.W  #0,D1         ; minimum Y of 0
2F92|
2F92| 0C41 016C            @7          CMP.W   #MaxY,D1         ; new Y-coordinate <= 364
2F96| 6F04                BLE.S   @8          ; branch if <= 364
2F98| 323C 016C            MOVE.W  #MaxY,D1         ; maximum Y of 364
2F9C|
2F9C| 31C0 0486            @8          MOVE.W  D0,MousX        ; update Mouse X-coordinate

```

```

2FA0| 31C1 0488          MOVE.W D1,MousY          ; update Mouse Y-coordinate
2FA4| 31C0 0496          MOVE.W D0,CrsrX          ; also update cursor coordinates
2FA8| 31C1 0498          MOVE.W D1,CrsrY
2FAC| 4CDF 003E          MOVEM.L (SP)+,D1-D5        ; restore registers
2FB0| 4E75              RTS                      ; return
2FB2|
2FB2| ;-----
2FB2| ;
2FB2| ; Routine to initialize mouse handling
2FB2| ;
2FB2| ;-----
2FB2|
2FB2| MousInit
2FB2| 31FC 0168 0486        MOVE.W #360,MousX          ; set initial mouse location
2FB8| 31FC 00B6 0488        MOVE.W #182,MousY          ; to center of screen
2FBE| 31FC 0008 048E        MOVE.W #8,MousThresh       ; set scaling threshold
2FC4| 707C              MOVEQ #$7C,D0              ; and enable mouse data
2FC6| 6100 D98E          BSR      COPSCMD
2FCA| 4E75              RTS
2FCC|
2FCC| ;-----
2FCC| ;
2FCC| ; Hardware Interface for the Cursor
2FCC| ;
2FCC| ; Written by Rick Meyers
2FCC| ; (c) Apple Computer Incorporated, 1983
2FCC| ;
2FCC| ;
2FCC| ; The routines below provide an assembly language interface to the cursor.
2FCC| ; Input parameters are passed in registers, output parameters are returned
2FCC| ; in registers. Unless otherwise noted, all registers are preserved.
2FCC| ;
2FCC| ; The Cursor
2FCC| ;
2FCC| ; The cursor is a small image that is displayed on the screen. It's shape
2FCC| ; is specified by two bitmaps, called 'data' and 'mask'. These bitmaps are
2FCC| ; 16 bits wide and from 0 to 32 bits high. The rule used to combine the
2FCC| ; bits already on the screen with the data and mask is
2FCC| ;
2FCC| ; screen <- (screen and (not mask)) xor data.
2FCC| ;
2FCC| ; The effect is that white areas of the screen are replaced with the cursor
2FCC| ; data. Black areas of the screen are replaced with (not mask) xor data.
2FCC| ; If the data and mask bitmaps are identical, the effect is to 'or' the data
2FCC| ; onto the screen.
2FCC| ;
2FCC| ; The cursor has both a location and a hotspot. The location is a position

```



```

2FCC|           ; on the screen, with X-coordinates of 0 to 720 and Y-coordinates of 0 to 364 .
2FCC|           ; The hotspot is a position within the cursor bitmaps, with X- and Y-coordi-
2FCC|           ; nates ranging from 0 to 16. The cursor is displayed on the screen with it's
2FCC|           ; hotspot at location. If the cursor's location is near an edge of the screen,
2FCC|           ; the cursor image may be partially or completely off the screen.
2FCC|           ;
2FCC|           ;-----
2FCC|           ;
2FCC|           ; Routine:   CursorInit
2FCC|           ; Arguments: None
2FCC|           ; Function:  Sets up the initial defaults used by the ROM cursor. Initial
2FCC|           ;               position is set for center of the screen.
2FCC|           ;
2FCC|           ;-----
2FCC|
2FCC| 4278 0490      CursorInit      MOVE.W #0,CrsrHotX      ; cursor hotspot X-coordinate
2FD0| 4278 0492      MOVE.W #0,CrsrHotY      ; cursor hotspot Y-coordinate
2FD4| 31FC 0010 0494 MOVE.W #16,CrsrHeight    ; cursor hieght, 0-32
2FDA| 31FC 0168 0496 MOVE.W #360,CrsrX      ; initial cursor X-coordinate
2FE0| 31FC 00B6 0498 MOVE.W #182,CrsrY      ; initial cursor Y-coordinate
2FE6| 61CA          BSR.S MousInit      ; init mouse for cursor control
2FE8| 4E75          RTS                      ; return
2FEA|
2FEA|           ;-----
2FEA|           ;
2FEA|           ; Cursor Hide and Display
2FEA|           ;
2FEA|           ; Care must be taken when updating the screen image which is 'under' the
2FEA|           ; cursor. The simplest approach is to remove the cursor from the screen
2FEA|           ; (hide), do the screen modification, then redisplay the cursor (display).
2FEA|           ; Each hide operation must be followed by a corresponding display
2FEA|           ; operation. The operations are paired and can be nested. The first of a
2FEA|           ; series of hides removes the cursor from the screen; it's corresponding
2FEA|           ; display redisplayes the cursor. Intervening operations have no apparent
2FEA|           ; effect.
2FEA|           ;
2FEA|           ;-----
2FEA|           ;
2FEA|           ; Routine:   CursorHide
2FEA|           ; Arguments: None
2FEA|           ; Function:  Remove the cursor from the screen. Note that every call to
2FEA|           ;               CursorHide must be followed by exactly one call to CursorDisplay.
2FEA|           ;-----
2FEA|
2FEA| 48E7 C0C0      CursorHide      MOVEM.L D0-D1/A0-A1,-(SP) ; save registers
2FEE| 41F8 04A2      LEA          SavedData,A0      ; saved data address

```

```

2FF2| 2278 0528          MOVE.L   SavedAddr,A1          ; saved data screen address
2FF6| 3038 0526          MOVE.W   SavedRows,D0         ; rows of saved data
2FFA| 323C 005A          MOVE.W   #MaxX/8,D1          ; bytes per row on screen
2FFE| 6004              BRA.S     @2                  ; test of rows=0
3000|
3000| 2298              @1          MOVE.L   (A0)+,(A1)          ; from saved to screen
3002| D2C1              ADD.W     D1,A1              ; screen address of next row
3004| 51C8 FFFA          @2          DBF      D0,@1              ; loop 'SavedRows' times
3008|
3008| 4CDF 0303          MOVEM.L  (SP)+,D0-D1/A0-A1      ; restore registers
300C| 4E75              RTS                          ; return
300E|
300E| ;-----
300E| ;
300E| ; Routine:   CursorDisplay
300E| ; Arguments: none
300E| ; Function:  Redisplay the cursor. Note that every call to CursorDisplay
300E| ;            must be preceded by exactly one call to CursorHide.
300E| ;
300E| ; Register Assignments:
300E| ;
300E| ;      D0 -- saved data X-coordinate, cursor data
300E| ;      D1 -- saved data Y-coordinate, cursor mask
300E| ;      D2 -- left shift count
300E| ;      D3 -- 32-bit mask
300E| ;      D4 -- rows of saved data
300E| ;      D5 -- bytes per row on screen
300E| ;
300E| ;      A0 -- saved data address
300E| ;      A1 -- saved data screen address
300E| ;      A2 -- cursor data address
300E| ;      A3 -- cursor mask address
300E| ;-----
300E|
300E| 48E7 FCF0          CursorDisplay  MOVEM.L D0-D5/A0-A3,-(SP) ; save registers
3012|
3012| 41F8 04A2          LEA       SavedData,A0          ; saved data address
3016| 2278 0110          MOVE.L   ScrnBase,A1          ; screen memory address
301A| 45FA 0900          LEA       CrsrData,A2          ; cursor data bitmap address
301E| 47FA 08FC          LEA       CrsrMask,A3          ; cursor mask bitmap address
3022| 3038 0490          MOVE.W   CrsrHotX,D0          ; cursor hotspot X-coordinate
3026| 3238 0492          MOVE.W   CrsrHotY,D1          ; cursor hotspot Y-coordinate
302A| 3838 0494          MOVE.W   CrsrHeight,D4         ; cursor height
302E|
302E| ; Compute and bounds check the X-coordinate of the data under the cursor.
302E| 9078 0496          @11         SUB.W   CrsrX,D0          ; cursor X-coordinate
3032| 4440              NEG.W     D0                  ; - cursor hotspot X-coordinate

```




```

3034| 3400          MOVE.W  D0,D2          ; upper left X-coordinate
3036| 0242 000F      AND.W   #$000F,D2     ; bit offset within word
303A| 4442          NEG.W   D2             ; negated and converted to
303C| 0642 0010      ADD.W   #16,D2        ; left shift count
3040| 4283          CLR.L   D3             ; 32-bit mask
3042| 4643          NOT     D3             ; D3 = $0000FFFF          RM000
3044| E5AB          LSL.L   D2,D3          ; shifted into position          RM000
3046|
3046| 0240 FFF0      AND.W   #$FFF0,D0     ; upper left X-coord rounded down
304A| 6C0A          BGE.S   @0             ; branch if >= 0
304C|
304C| 4240          MOVE.W   #0,D0          ; minimum upper left X-coord of 0
304E| E18B          LSL.L   #8,D3          ; adjust 32-bit mask
3050| E18B          LSL.L   #8,D3          ; adjust 32-bit mask
3052| 0642 0010      ADD.W   #16,D2        ; adjust left shift count
3056|
3056| 0C40 02B0      @0      CMP.W   #MaxX-32,D0 ; upper left X-coord <= 720-32
305A| 6F14          BLE.S   @2             ; branch if <= 720-32
305C|
305C| 0C40 02D0      CMP.W   #MaxX,D0        ; cursor off right edge ?
3060| 6602          BNE.S   @1             ; branch if not off right edge
3062| 7600          MOVEQ   #0,D3          ; mask off all bits
3064|
3064| 303C 02B0      @1      MOVE.W   #MaxX-32,D0 ; maximum X-coord of 720-32
3068| E08B          LSR.L   #8,D3          ; adjust 32-bit mask
306A| E08B          LSR.L   #8,D3          ; adjust 32-bit mask
306C| 0642 0010      ADD.W   #16,D2        ; adjust left shift count
3070|
3070|                ; Compute and bounds check the Y-coordinate of the data under the cursor.
3070|
3070| 9278 0498      @2      SUB.W   CrsrY,D1    ; cursor Y-coordinate
3074| 4441          NEG.W   D1             ; - cursor hotspot Y-coordinate
3076| 6C0C          BGE.S   @3             ; branch if upper left Y >= 0
3078|
3078| D841          ADD.W   D1,D4             ; decrease rows of saved data
307A| D241          ADD.W   D1,D1           ; double for byte count
307C| 94C1          SUB.W   D1,A2           ; increase cursor data address
307E| 96C1          SUB.W   D1,A3           ; increase cursor mask address
3080| 4241          MOVE.W   #0,D1          ; minimum upper left Y of 0
3082| 6010          BRA.S   @4             ; continue
3084|
3084| 3A3C 016C      @3      MOVE.W   #MaxY,D5    ; maximum Y-coordinate
3088| 9A44          SUB.W   D4,D5           ; - cursor height
308A| B245          CMP.W   D5,D1           ; cursor bottom <= 364-CrsrHeight ?
308C| 6F0C          BLE.S   @5             ; branch if <= 364-CrsrHeight
308E|
308E| 383C 016C      MOVE.W   #MaxY,D4        ; last row on screen

```

```

3092| 9841                SUB.W  D1,D4                ; adjust rows of saved data
3094|
3094| 4A44                @4                TST.W  D4                ; rows of saved data >= 0 ?
3096| 6C02                BGE.S  @5                ; branch if >= 0
3098| 4244                MOVE.W  #0,D4                ; minimum rows of saved data
309A|
309A| 31C0 0522            @5                MOVE.W  D0,SavedX            ; saved data X-coordinate
309E| 31C1 0524            MOVE.W  D1,SavedY            ; saved data Y-coordinate
30A2| 31C4 0526            MOVE.W  D4,SavedRows        ; rows of saved data
30A6|
30A6|                ;   Display the cursor on the screen.
30A6|
30A6| E648                LSR.W  #3,D0                ; convert X-coord to bytes
30A8| D2C0                ADD.W  D0,A1                ; and add to screen address
30AA| 7A5A                MOVEQ  #MaxX/8,D5            ; bytes per row on screen
30AC| C2C5                MULU   D5,D1                ; * Y-coord
30AE| D3C1                ADD.L  D1,A1                ; added to screen address
30B0| 21C9 0528            MOVE.L  A1,SavedAddr        ; saved data screen address
30B4| 6016                BRA.S  @7                ; test for rows=0
30B6|
30B6| 301A                @6                MOVE.W  (A2)+,D0            ; cursor data
30B8| E5B8                ROL.L  D2,D0                ; shift to proper bit position
30BA| C083                AND.L  D3,D0                ; eliminate unwanted bits
30BC| 321B                MOVE.W  (A3)+,D1            ; cursor mask
30BE| E5B9                ROL.L  D2,D1                ; shift to proper bit position
30C0| C283                AND.L  D3,D1                ; eliminate unwanted bits
30C2| 4681                NOT.L  D1                ; invert cursor mask
30C4|
30C4| 20D1                MOVE.L  (A1),(A0)+            ; from screen to saved data
30C6| C391                AND.L  D1,(A1)            ; screen and (not mask)
30C8| B191                EOR.L  D0,(A1)            ; xor cursor data
30CA| D2C5                ADD.W  D5,A1                ; screen address of next row
30CC| 51CC FFE8            @7                DBF     D4,@6                ; loop 'SavedRows' times
30D0|
30D0| 4CDF 0F3F            MOVEM.L (SP)+,D0-D5/A0-A3    ; restore registers
30D4| 4E75                RTS                        ; return
30D6|
30D6|                .ENDC                        ;{USERINT}
30D6|                .IF AAPL = 1
30D6|                .ENDC                        ;{USERINT}
30D6|                .IF AAPL = 1
30D6|                .ENDC
30D6|                .ENDC                        ;{ROM4K}
30D6|
30D6|
30D6|                .INCLUDE RM248.G.TEXT
30D6|

```



```

30D6|          .PAGE
30D6|          .IF      EXTERNAL = 1
30D6|          .IF      USERINT = 1
30D6|          ;*****
30D6|          ; Mini-Graphics Package      *
30D6|          ; Contributed by Mike Urquhart *
30D6|          ; Copyright 1983, 1984 Apple  *
30D6|          ;*****
30D6|
30D6|          ;
30D6|          ;
30D6|          ;      DRAWDESK:
30D6|          ;
30D6|          ;      this routine performs the following sequences:
30D6|          ;
30D6|          ;      1.      clears the screen to a white background.
30D6|          ;      2.      makes a menu bar by drawing a single pixel horizontal
30D6|          ;                  line across the screen.
30D6|          ;      3.      fills the screen area below the menu bar with the standart
30D6|          ;                  gray shade.
30D6|          ;
30D6|          ;      Also has an entry point called CLRDESK that only does item (3).
30D6|          ;
30D6|          ;      Destroys regs D0-D2,A1,A5-A6
30D6|          ;
30D6|          ;
30D6|          ;
30D6|          ;      1.      clear screen to white background
30D6|          ;
30D6|          DRAWDESK
30D6|          4282      clr.l    d2
30D8| 612E          bsr.s     whiten
30DA|
30DA|          ;
30DA|          ;      2.      make menu bar border
30DA|
30DA|          CLRDESK moveq    #ROWBYTES,d0      ;set length of line = 90 bytes
30DC| 74FF          moveq    #-1,d2             ;draw a black pattern
30DE| 327C 05A0      move.w    #MENULINE,A1      ;start at offset 1440 address
30E2| 7201          moveq    #1,d1             ;draw only 1 pixel line
30E4| 6142          bsr.s     paint_box
30E6|
30E6|          ;
30E6|          ;      set a6 to starting pixel address for grey routine below
30E6|          ;      set a5 to limit
30E6|          ;
30E6| 3C7C 05FA          move     #DESKLINE,a6
30EA| 3A7C 7FF8          move     #DESKLMT,a5      ;set limit

```



```

30EE|
30EE| ;
30EE| ; 3. make the grey background: to make the gray background, alternating
30EE| ; rows of $5555 and $AAAA starting with $AAAA are written to the
30EE| ; screen area.
30EE| ;
30EE|
30EE| 243C AAAA 5555 gray MOVE.L #DESKPATRN,D2 ;set up grey pattern
30F4|
30F4| 4842 gray1 swap d2 ;start with the $AAAA pattern
30F6| 7201 moveq #1,d1 ;draw a one pixel high line
30F8| 224E move.l a6,a1 ;starting pixel address must be in a1 for call
30FA| 6100 002C bsr paint_box
30FE|
30FE| DCC0 @1 add d0,a6 ;bump to next pixel line
3100| BCCD cmp.w a5,a6 ;is a6 less then max?
3102| 6DF0 blt.s gray1 ;loop if yes
3104| 4E75 rts ;else done
3106|
3106| ;
3106| ;
3106| ; subroutine blacken and whiten: clears full screen to black or white
3106| ;
3106| ; Calls: bsr blacken (no parameters)
3106| ;
3106| ; clr.l d2
3106| ; bsr whiten
3106| ;
3106| ; registers used: d2.l - contains either FFFFFFFF or 0 on return
3106| ; d0.b - destroyed but contains $5A (90) on return
3106| ; d1.w - destroyed-->should contain 0 on return
3106| ; a1.l - destroyed
3106| ;
3106| ;
3106| ; This routine calls paint_box, so other registers may be destroyed.
3106| ;
3106| ;
3106| ;
3106| ;
3106| BLACKEN
3106| 74FF moveq #-1,d2 ;black fill pattern
3108|
3108| whiten
3108| 323C 016B move.w #MaxY-1,d1 ;number of pixels in screen box heigth
310C| 705A moveq #ROWBYTES,d0 ;length of screen box is 90 bytes
310E| 93C9 suba.l A1,A1 ;clear A1
3110| 6116 bsr.s paint_box

```



Computer Boot ROM 2.48 Listing • 205 of 133

```

3128|      ; _____
3128|
3128|
3128|      PAINT_BOX
3128|
3128|      paintb1 lea    movinst,A4
312C|      6004          bra.s   cont
312E|
312E|      inverse lea    exclusive,A4
3132|      D3F8 0110     cont    ADD.L   SCRNBASE,A1      ;add video address to starting pixel address offset.
3136|
3136|      paintb2
3136|      2F0A          MOVE.L   A2,-(SP)      ;save reg
3138|      ;            do some setup steps
3138|      ;
3138|      4283          CLR.L    D3            ;clear for use
313A|      3600          MOVE.W   D0,D3        ;modify width/length
313C|      4483          NEG.L    D3          ;negate the width/length (D3 = -width/length)
313E|      ;
313E|      ;            add the length of one horizontal pixel row-2 to the -width/length
313E|      ;            to derive an offset which can be added to the updated address pointer
313E|      ;            (when it reflects the right corner or end point of the box/line)
313E|      ;            in order to arrive at the next row starting address.
313E|      ;
313E|      0683 0000 005A ADDI.L   #ROWBYTES,D3      ;create displacement for following sequence
3144|      ;
3144|      ;
3144|      2449          MOVE.L   A1,A2        ;create ending column check address
3146|      D5C0          ADD.L    D0,A2        ;add length of line to obtain ending column address
3148|      ;
3148|      ;            A1 now points to the left top coordinate of the box or line.
3148|      ;            A2 now points to the right top coordinate of the box or line.
3148|      ;
3148|      startop
3148|      4ED4          jmp      (A4)         ;execute the correct operation (EOR or MOVE)
314A|
314A|      movinst
314A|      32C2          MOVE.w   D2,(A1)+     ;start the sequence
314C|      6002          bra.s    compare
314E|
314E|      exclusive
314E|      B559          eor.w    D2,(A1)+
3150|
3150|      B5C9          compare  CMP.L   A1,A2      ;reached the right most point?
3152|      6702          BEQ.S    nextline     ;yes
3154|      60F2          BRA.S    startop
3156|      ;

```

```

3156|          ;      YES
3156|          ;
3156|      nextline
3156| D3C3          ADD.L   D3,A1          ;add 1 horizontal line length (5A) minus line length
3158| D4FC 005A      ADDA   #ROWBYTES,A2      ;to reach the left most point of the box on the      RM000
315C| 5341          SUBQ.W  #1,D1          ;next horizontal scan line.
315E| 66E8          BNE.S   startop        ;loop until done.
3160|          ;
3160| 245F          MOVE.L  (SP)+,A2        ;restore and
3162| 4E75          RTS          ;return
3164|
3164|          ;-----
3164|          ; Entry point to set cursor ptrs and make alert box for power cycling
3164|          ;-----
3164|
3164|      MAKEPCALRT
3164| 7A59          MOVEQ   #PCROW,D5        ;set row ptr
3166| 7C0C          MOVEQ   #PCCOL,D6        ;and col ptr
3168|          ;then drop into alert box routine
3168|          ;
3168|          ;
3168|          ;      MAKEALERT
3168|          ;
3168|          ;      This routine creates an alert box with no title.
3168|          ;
3168|          ;-----
3168|      MAKEALERT
3168| 48E7 C040      movem.l d0-d1/a1,-(sp)
316C|
316C| 704E          MOVEQ   #ALRTWIDTH,D0        ;set parameters for box
316E| 223C 0000 00A4  MOVE.L  #ALRTHIGH,D1
3174| 327C 1140      MOVEA   #ALRTSTRT,A1
3178| 616C          bsr.s   makebox
317A|
317A| 4CDF 0203      movem.l (sp)+,d0-d1/a1
317E| 4E75          rts
3180|
3180|          ;-----
3180|          ;
3180|          ;      MAKETEST
3180|          ;
3180|          ;      This routine creates an alert box for test icon display.
3180|          ;
3180|          ;-----
3180|      MAKETEST
3180| 48E7 C040      movem.l d0-d1/a1,-(sp)
3184|

```

```

3184| 7046          MOVEQ    #TSTWIDTH,D0          ;set parameters for alert box
3186| 7254          MOVEQ    #TSTWHIGH,D1
3188| 327C 1144      MOVEA    #TSTWSTRT,A1
318C| 6158          BSR.S     MAKEBOX                ;go draw box
318E|
318E| 7A40          MOVEQ    #TSTMROW,D5            ;set cursor ptrs
3190| 7C0E          MOVEQ    #TSTMCOL,D6
3192| 47FA 0CA6      LEA      CHKMSG,A3            ;set message ptr
3196| 72FF          MOVEQ    #-1,D1                ;append '...' string
3198| 6100 049A      BSR      DSPSTRING            ;and go display it
319C|
319C| 6100 03EA      BSR      DSPCPU                ;display test icons
31A0| 6100 03FA      BSR      DSPIOB
31A4| 6100 03EC      BSR      DSPMBRD
31A8| 6100 03FC      BSR      DSPXCRD
31AC|
31AC| 4CDF 0203      movem.l  (sp)+,d0-d1/a1
31B0| 4E75          rts
31B2|
31B2| ;
31B2| ;
31B2| ;          MAKEDBOX
31B2| ;
31B2| ;          This routine creates a dialog box.
31B2| ;
31B2| ;
31B2| ;
31B2| MAKEDBOX
31B2| 48E7 C040      movem.l  d0-d1/a1,-(sp)
31B6|
31B6| 7042          MOVEQ    #DBOXWIDTH,D0          ;set parameters for dialog box
31B8| 7214          MOVEQ    #DBOXHIGH,D1
31BA| 327C 071E      MOVEA    #DBOXSTRT,A1
31BE| 6126          bsr.s     makebox
31C0|
31C0| 4CDF 0203      movem.l  (sp)+,d0-d1/a1
31C4| 4E75          rts
31C6|
31C6| ;
31C6| ;
31C6| ;          routine makewindow
31C6| ;
31C6| ;          This routine creates a fixed window of the folder type.
31C6| ;          The calling routine must provide the address of the
31C6| ;          string which will be used to fill in the title box.
31C6| ;
31C6| ;          Call:
31C6| ;          move    <window width>,d0

```



Computer Boot ROM 2.48 Listing • 209 of 133

```

31E6|      ; _____
31E6|
31E6|
31E6|      MAKEBOX
31E6|      ;
31E6|      ;      make the basic window--->no edges  D2 must be set to 0 on call
31E6|      ;                                          d0 must be set to the width (90 maximum)
31E6|      ;                                          d1 must be set to the height (364 maximum)
31E6|      ;                                          A1 must be the offset address (0..32670)
31E6|      ;
31E6|      ;
31E6| 48E7 F878      movem.l d0-d4/a1-a4,-(sp)
31EA| 48E7 C040      movem.l d0-d1/a1,-(sp)      ;save the height and the starting pixel address
31EE|
31EE| 4282          clr.l   d2                  ;the pattern
31F0| 6100 FF36      bsr     paint_box
31F4|      ;
31F4|      ;      now draw the individual edges including the shadow edges on the right
31F4|      ;      and bottom of the window
31F4|      ;
31F4|      ;
31F4|      ;      draw bottom horizontal edge
31F4|      ;
31F4| 93C3          sub.l   d3,a1
31F6| 93C0          sub.l   d0,a1              ;a1 is currently equal to the lower right point
31F8|              ; of the box so we can subtract the width to
31F8|              ; calculate the lower left point of the box.
31F8|
31F8| 7201          moveq   #1,d1              ;set 1 pixel height
31FA| 74FF          moveq   #-1,d2             ;set the line pattern
31FC| 6100 FF38      bsr     paintb2           ;remember a4 is set up to point to the move.w
3200|              ; instruction
3200|      ;
3200|      ;      draw bottom horizontal shadow
3200| @1      addq    #2,a1          ;shadow begin offset by 2
3202| 7201          moveq   #1,d1              ;draw a one pixel line
3204| 5540          subq    #2,d0
3206| 6100 FF2E      bsr     paintb2           ;go
320A|      ;
320A|      ;      draw top horizontal edge
320A|      ;
320A| 4CDF 0203      @2      movem.l (sp)+,d0-d1/a1 ;restore original parameters
320E| 48E7 4040      movem.l d1/a1,-(sp)
3212| 7201          moveq   #1,d1              ;draw a 1 pixel line
3214| 6100 FF12      bsr     paint_box
3218|      ;

```



```

3218|          ;      now draw the right edge plus the right edge's shadow, use a1
3218|          ;      with a column parameter of 0
3218|          ;
3218| 4CDF 0410      movem.l  (sp)+,d4/a2
321C| 7401      moveq   #1,d2
321E| 5544      subq    #2,d4
3220| 5540      subq    #2,d0
3222| D3C0      add.l   d0,a1
3224|
3224| 2204      move.l   d4,d1
3226| 4280      clr.l   d0
3228| 6130      bsr.s   paint_v
322A|
322A| 303C 00B6      move.w   #182,d0          ;set one byte offset from right edge
322E| 7407      moveq   #7,d2
3230| 2204      move.l   d4,d1
3232| 6136      bsr.s   paintbit          ;draw first pixel line of the right shadow
3234|
3234| 303C 0110      move.w   #272,d0
3238| 7406      moveq   #6,d2
323A| 2204      move.l   d4,d1
323C| 5341      subq    #1,d1          ;reduce height by 1 to compensate for shadow offset
323E| 612A      bsr.s   paintbit
3240|
3240|          ;
3240|          ;      now draw the left vertical edge of the box
3240|          ;
3240| 224A      move.l   a2,a1          ;restore the starting pixel address
3242| D3F8 0110      add.l   SCRNBASE,a1      ;add in video address
3246| 705A      moveq   #90,d0
3248| D3C0      add.l   d0,a1
324A| 343C 8000      move    #32768,d2          ;set the pattern
324E|
324E| 2204      move.l   d4,d1          ;the height minus two
3250| 4280      clr.l   d0          ;column 0 offset
3252| 6106      bsr.s   paint_v
3254| 4CDF 1E1F      movem.l  (sp)+,d0-d4/a1-a4
3258|
3258| 4E75      rts
325A|
325A|          ;
325A|          ;
325A|          ;      Routine paint_v
325A|          ;
325A|          ;      This routine "paints" a vertical column one word wide.
325A|          ;
325A|          ;      Call:   BSR      paint_v

```



```

325A|      ;
325A|      ;      register setup:
325A|      ;
325A|      ;      A1 must contain the screen base address.
325A|      ;      D2 must contain a one word pattern.
325A|      ;      D1 must contain the number of pixels in the vertical length of the line
325A|      ;      in the range 1..364
325A|      ;      D0 must contain the screen word column in the range 0..44
325A|      ;
325A|      ;      Assumptions:  location SCRNBASE contains the video address
325A|      ;
325A|      ;      registers used: D0-D1
325A|      ;
325A|      ;
325A|      PAINT_V
325A|      paintv1
325A|
325A| 3382 0000      @1      MOVE.W  D2,0(A1,D0.W)      ;write to screen
325E| 5341          SUBQ.W  #1,D1          ;decrement count
3260| 6706          BEQ.S   @2              ;return
3262| 0640 005A      ADDI.W  #ROWBYTES,D0      ;bump to next horizontal line
3266| 60F2          BRA.S   @1              ;loop to continue writing vertical.
3268|
3268| 4E75          @2      RTS              ;return
326A|
326A|      ;
326A|      ;
326A|      ;      Routine paintbit
326A|      ;
326A|      ;      This routine "paints" a vertical line one bit wide.
326A|      ;
326A|      ;      Call:   BSR      paintbit
326A|      ;      register setup:
326A|      ;
326A|      ;      A1 must contain the screen base address.
326A|      ;      D2 must contain the bit number to set
326A|      ;      D1 must contain the number of pixels in the vertical length of the line
326A|      ;      in the range 1..364
326A|      ;      D0 must contain the screen word column in the range 0..44
326A|      ;
326A|      ;      Assumptions:  location SCRNBASE contains the video address
326A|      ;
326A|      ;      registers used: D0-D1
326A|      ;
326A|      ;
326A|      PAINTBIT

```

Computer Boot ROM 2.48 Listing • 213 of 133

```

3294| 3800                MOVE    D0,D4                ;save for later display
3296|
3296|                ; compute X,Y pixel coordinates from starting address
3296|
3296| 6100 016E            BSR      GETROWCOL            ;get pixel row, byte col
329A| CCFC 0008            MULU    #8,D6                ;convert to pixel col
329E| 3586 2002            MOVE    D6,2(A2,D2.W)        ;save upper left X
32A2| 3585 2004            MOVE    D5,4(A2,D2.W)        ; and Y coordinates
32A6|
32A6| 700A                MOVEQ    #BTNWIDTH,D0        ;set button parameters
32A8| 721C                MOVEQ    #BTNHIGH,D1
32AA| 614C                BSR.S    DRAWBTN            ;draw the button
32AC|
32AC| 2F09                MOVE.L    A1,-(SP)            ;save original val
32AE| 3278 052C            MOVE    LwrRight,a1        ;compute lower right coordinates
32B2| 6100 0152            BSR      GETROWCOL            ;get pixel row
32B6| CCFC 0008            MULU    #8,D6                ;and pixel col
32BA| 3586 2006            MOVE    D6,6(A2,D2.W)        ;save as X and
32BE| 3585 2008            MOVE    D5,8(A2,D2.W)        ; Y coordinates
32C2| 225F                MOVE.L    (SP)+,A1            ;restore starting value
32C4|
32C4|                ; Add the button label and description
32C4|
32C4| 48E7 F870            movem.l d0-d4/a1-a3,-(sp)    ;save parameters
32C8| E288                lsr.l     #1,d0                ;divide window width
32CA| 6100 013A            bsr      getrowcol            ;get the row and column coordinate of
32CE|                                ;the window corner point
32CE| DC80                add.l     d0,d6                ;adjust column pointer
32D0|
32D0| E289                lsr.l     #1,d1                ;divide window heigth
32D2| 5981                subq.l    #4,d1                ;decrement by half font heigth
32D4| DA81                add.l     d1,d5                ;adjust row coordinate
32D6|
32D6| 5346                SUBQ      #1,D6                ;go back 1 and
32D8| 4240                CLR       D0                    ;display apple icon
32DA| 6100 045E            BSR      DSPVAL              ;
32DE| 2004                MOVE.L    D4,D0                ;get char to display
32E0| 6100 0458            bsr      dspval              ;and go display alternate keycode
32E4|
32E4| 4CDF 0E1F            movem.l (sp)+,d0-d4/a1-a3    ;retrieve parameters
32E8| 4CDF 0402            MOVEM.L (SP)+,D1/A2          ;retrieve location and string indicator
32EC|
32EC| 224A                MOVE.L    A2,A1                ;compute output pt for description
32EE| 6100 0116            BSR      GETROWCOL            ;
32F2| 6100 0340            BSR      DSPSTRING           ;and display it
32F6|
32F6| 4E75                RTS

```



Computer Boot ROM 2.48 Listing • 215 of 133

Computer Boot ROM 2.48 Listing • 216 of 133

Computer Boot ROM 2.48 Listing • 217 of 133

```

3388| CCFC 0008          MULU    #8,D6          ;convert to pixels
338C| 30C6          MOVE     D6,(A0)+        ;save upper X coordinate
338E| 30C5          MOVE     D5,(A0)+        ;save upper Y coordinate
3390| DC40          ADD      D0,D6          ;compute and save
3392| 30C6          MOVE     D6,(A0)+        ; lower X coordinate
3394| 0645 000B      ADD      #MENUSPC/90,D5 ;compute and save
3398| 30C5          MOVE     D5,(A0)+        ; lower Y coordinate
339A| 2F09          MOVE.L   A1,-(SP)        ;save "box" address
339C| 224A          MOVE.L   A2,A1          ;get address for menu message
339E| 6166          BSR.S    GETROWCOL       ;compute msg output pt
33A0| 6100 035E      BSR      DSPMSG        ;do display
33A4| 225F          MOVE.L   (SP)+,A1        ;restore address for box
33A6| D3C3          ADD.L    D3,A1          ;incr to next start pt for rectangle
33A8| D5C3          ADD.L    D3,A2          ;also incr message address
33AA|
33AA| 5344          SUBQ     #1,D4          ;loop until done
33AC| 66D2          BNE.S    @1
33AE|
33AE| 31C1 053A      MOVE     D1,RectCnt      ;save new rectangle count
33B2| 4CDF 010F      MOVEM.L  (SP)+,D0-D3/A0 ;restore regs and exit
33B6| 4E75          RTS
33B8|
33B8| ;
33B8| ; routine writetitle --> writes the title for the window whose
33B8| ; left top corner is addressed by a1,
33B8| ; whose width is indicated by d0.
33B8| ;
33B8| ; To arrive at the correct address to write the title string,
33B8| ; the window width and string length must be divided by 2, the
33B8| ; result of the string division is subtracted from the width
33B8| ; division to arrive at the general byte column position to
33B8| ; begin writing the title.
33B8| ;
33B8| ;
33B8| ; inputs:          d0=window width
33B8| ;                  a1=window left/top corner point as a
33B8| ;                  0..32670 screen offset address.
33B8| ;                  a3=title string address
33B8| ;
33B8| ; output:         all registers used are restored.
33B8| ;
33B8| ;
33B8| ;
33B8| WRITETITLE
33B8| 48E7 FEFE      movem.l  d0-d6/a0-a6,-(sp) ;save as many registers as possible
33BC| 615A          bsr.s    getlength        ;calculate the length of the string
33BE|

```



```

33BE| E288          lsr.l  #1,d0          ;divide window width
33C0| E28A          lsr.l  #1,d2          ;divide string length
33C2| 9082          sub.l  d2,d0          ;1/2 width - 1/2 length --->d0
33C4|
33C4| 6140          bsr.s   getrowcol     ;get the row and column coordinate of
33C6|              ;the window corner point
33C6| 5845          addq   #4,d5          ;adjust for 4 pixel rows
33C8| DC80          add.l  d0,d6          ;adjust column pointer
33CA|
33CA| 48E7 0600      movem.l d5/d6,-(sp)    ;stack line/column pointers for later ref
33CE|
33CE| 6100 0330      bsr     dspmsg
33D2|
33D2| 4CDF 0060      movem.l (sp)+,d5/d6    ;restore the parameters
33D6| 5945          subq   #4,d5          ;reduce the pixel pointer by 4
33D8| 5546          subq   #2,d6          ;backspace the column pointer by 2
33DA| 0246 FFFE      andi.w #$FFFF,d6     ;round off to nearest word
33DE| 9DCE          sub.l  a6,a6          ;clear a6
33E0| 6100 033C      bsr     setcrsr2      ;convert standard row/col to pixel offset
33E4| 224E          move.l a6,a1          ;move to a1
33E6|              ;
33E6|              ; restore the string pointer so that its length can be recalculated.
33E6|              ;
33E6| 266F 0028      move.l 40(sp),a3
33EA| 6100 002C      bsr     getlength     ;the length is returned in a2 and d2
33EE| 5242          addq   #1,d2          ;must round off the length to a word boundary
33F0| 0242 FFFE      andi.w #$FFFF,d2
33F4| 2002          move.l d2,d0          ;set width of inverted window to length..
33F6| 5840          addq   #4,d0          ;..of string plus 4
33F8| 720F          moveq  #15,d1         ;set the height of the window
33FA| 74FF          moveq  #-1,d2         ;set the mask
33FC| 6100 FD30      bsr     inverse       ;invert the title
3400|
3400| 4CDF 7F7F      movem.l (sp)+,d0-d6/a0-a6
3404| 4E75          rts
3406|
3406|
3406| ; routine getrowcol --> converts a screen address displacement
3406| ; to row and column coordinates.
3406| ;
3406| ; inputs:      a1=screen address in the range 0..32759
3406| ; outputs:     d5 set to row
3406| ;              d6 set to byte col (the remainder)
3406| ;
3406| ; registers used: d2 (but saved and restored)
3406| ; registers destroyed: d5,d6 (set to new values)
3406|

```

```

3406|
3406|      GETROWCOL
3406| 2F02      move.l  d2,-(sp)      ;save d2
3408| 2409      move.l  a1,d2        ;move the address to d2
340A| 84FC 005A      divu    #ROWBYTES,d2    ;divide the address by 90 bytes
340E| 3A02      move    d2,d5        ;new row
3410| 4842      swap    d2
3412| 3C02      move.w  d2,d6        ;remainder is column
3414|
3414|      ;      MOVE.L  A1,D2      ;now calculate row
3414|      ;      SUBI.L  #TOPOFFSET,D2    ;decr for offset
3414|      ;      DIVU    #RBYTES,D2    ;divide by bytes per char row
3414|      ;      MOVE    D2,D5
3414|      ;      ADDQ    #1,D5        ;incr to next row
3414|
3414| 241F      move.l  (sp)+,d2      ;restore d2
3416| 4E75      rts                ;return
3418|
3418|      ; -----
3418|      ;
3418|      ;      routine getlength --> returns a value which is the length of
3418|      ;                          the string referenced by a3.  The
3418|      ;                          string is assumed to be terminated by
3418|      ;                          the null character value.
3418|      ;
3418|      ;      inputs:      a3=string address
3418|      ;      outputs:     d2,a2: string length
3418|      ;
3418|      ;      registers destroyed: d2,a2 (set to new values)
3418|      ; -----
3418|
3418|      GETLENGTH
3418| 244B      move.l  a3,a2
341A| 141A      @1      move.b  (a2)+,d2      ;read a byte
341C| 66FC      bne.s   @1          ;continue
341E| 95CB      sub.l   a3,a2        ;subtract the beginning from the end.
3420| 240A      move.l  a2,d2        ;move it to d2 for reference
3422| 4E75      rts                ;return
3424|
3424|      ; -----
3424|      ;
3424|      ;      Routine to display uncompressed icon.                      CHG008
3424|      ; INPUTS:
3424|      ;      A2 = pointer to uncompressed icon (48 x 32 bitmap)
3424|      ;      A6 = pointer to (even!) screen address for upper left hand
3424|      ;              corner of icon
3424|      ; SIDE EFFECTS:

```

```

3424|          ;      D0-D1 are trashed
3424|          ;-----
3424|
3424|      DSPRGICON
3424| 7005      MOVEQ    #ICONWIDTH-1,D0 ;set default width          CHG008
3426| 721F      MOVEQ    #ICONHIGH-1,D1 ; and heighth              CHG008
3428| 6102      BSR.S    OUTPUT      ;and do display                CHG008
342A| 4E75      RTS                      ;                          CHG008
342C|
342C|          ;-----
342C|          ;
342C|          ;      Subroutine to display message or icon.
342C|          ;
342C|          ;      The calling routine must provide the pointer to the raw bit map.
342C|          ;
342C|          ;      Call:   load d0 with number of bytes - 1 in the x axis
342C|          ;               load d1 with the number of pixels - 1 in the y axis
342C|          ;
342C|          ;               lea  sourcebytes,a2
342C|          ;               lea  destination,a6
342C|          ;               jsr  or bsr output
342C|          ;
342C|          ;      Returns with d6 updated.
342C|          ;
342C|          ;-----
342C|
342C|      OUTPUT
342C| 48E7 3802      MOVEM.L D2-D4/A6,-(SP) ;save regs
3430| 765A          moveq    #90,d3
3432|
3432| 4284      loop0    clr.l    d4
3434| 3400          move.w    d0,d2
3436| 6002          bra.s     loop2
3438|
3438| 5244      loop1    addq     #1,d4
343A|
343A| 1D9A 4000      loop2    move.b  (a2)+,0(a6,d4)
343E| 51CA FFEC      dbf      d2,loop1
3442|
3442| DDC3          add.l     d3,a6
3444| 51C9 FFEC      dbf      d1,loop0
3448| 5246          addq     #1,d6
344A|
344A| 4CDF 401C      MOVEM.L (SP)+,D2-D4/A6 ;restore
344E| 4E75          rts
3450|
3450|          .PAGE

```

```

3450| ;-----
3450| ;
3450| ; Routine to display error icon with id #.
3450| ;
3450| ; INPUTS:
3450| ;     D1 = id # to display
3450| ;     A2 = pointer to compressed icon
3450| ;     A5 = upper left corner address for icon as
3450| ;         offset from screen address
3450| ;
3450| ; SIDE EFFECTS:
3450| ;     D5-D6 are trashed
3450| ;-----
3450| DSPNUMICON
3450| 6100 00DA      BSR      DSPALRTICON      ;display the icon
3454| 3A7C 287E      MOVE     #ERRSTRT,A5      ;get icon address offset
3458| 610A          BSR.S     DSPNUM          ;display id # on icon
345A| DBF8 0110      ADD.L     SCRNBASE,A5     ;get screen address
345E| 6100 008C      BSR      DSPBAD          ;display bad mark over icon
3462| 4E75          RTS
3464|
3464| ;-----
3464| ;
3464| ; Routine to display icon id #.
3464| ; INPUTS:
3464| ;     D1 = id # to display
3464| ;     A2 = pointer to compressed icon
3464| ;     A5 = upper left corner address for icon as
3464| ;         offset from screen address
3464| ;
3464| ; SIDE EFFECTS:
3464| ;     D1/D5-D6/A2-A3 are trashed
3464| ;-----
3464|
3464| 2F00          DSPNUM  MOVE.L  D0,-(SP)      ;save reg
3466| 224D          MOVE.L  A5,A1              ;convert icon address to row, col
3468| 619C          BSR      GETROWCOL
346A|
346A| 47FA 05A8      LEA      XCARD,A3          ;slot card icon?
346E| B7CA          CMPA.L   A2,A3
3470| 6608          BNE.S    @1                ;skip if not
3472| 0645 0016      ADD      #SLOTROW,D5      ;add offsets for card id to cursor ptrs
3476| 5646          ADDQ     #SLOTCOL,D6
3478| 6038          BRA.S    @6
347A|
347A| 47FA 0552      @1     LEA      MEMBRD,A3    ;memory board?
347E| B7CA          CMPA.L   A2,A3              ;

```



```

3480| 6608                BNE.S  @2          ;skip if not
3482| 0645 0010          ADD    #MEMROW,D5    ;add offsets for card id
3486| 5846                ADDQ   #MEMCOL,D6
3488| 6028                BRA.S  @6          ;and go display
348A|
348A| 4A38 02AF          @2    TST.B  SYSTYPE    ;Lisa 2 system?          CHG009
348E| 6646                BNE.S  @99          ;exit if yes - no id #'s      CHG009
3490|
3490| 47FA 088C          LEA     DISKETTE,A3    ;diskette icon?
3494| B7CA                CMPA.L  A2,A3
3498| 0645 0012          ADD     #DISKROW,D5    ;compute posn for id #
349C| 5846                ADDQ   #DISKCOL,D6
349E| 6012                BRA.S  @6
34A0|
34A0| 47FA 067C          @3    LEA     DRIVEN,A3    ;drive icon?          CHG009
34A4| B7CA                CMPA.L  A2,A3          ;                      CHG009
34A6| 6606                BNE.S  @4          ;skip if not          CHG009
34A8| 5C45                ADDQ   #DRVROW,D5    ;add offsets for id #      CHG009
34AA| 5646                ADDQ   #DRVCOL,D6    ;                      CHG009
34AC| 6004                BRA.S  @6          ;and go display      CHG009
34AE|
34AE|                    @4
34AE| 5B45                SUBQ   #INSRTROW,D5    ;must be insert icon      CHG009/CHG024
34B0| 5846                ADDQ   #INSRTCOL,D6    ;                      CHG009
34B2|
34B2| 6100 0266          @6    BSR     SETCRSR    ;get screen address in A6
34B6| 5341                SUBQ   #1,D1          ;check number to display
34B8| 6606                BNE.S  @7
34BA| 45FA 0480          LEA     ONE,A2          ;set ptr to # icon
34BE| 600E                BRA.S  @9
34C0|
34C0| 5341                @7    SUBQ   #1,D1          ;is it 2?
34C2| 6606                BNE.S  @8
34C4| 45FA 047B          LEA     TWO,A2
34C8| 6004                BRA.S  @9
34CA|
34CA| 45FA 047A          @8    LEA     THREE,A2    ;must be 3
34CE|
34CE| 7000                @9    MOVEQ   #0,D0          ;set width - 1
34D0| 7204                MOVEQ   #4,D1          ;and heigth - 1
34D2| 6100 FF58          BSR.S  OUTPUT    ;display it
34D6|
34D6| 201F                @99   MOVE.L  (SP)+,D0    ;restore reg
34D8| 4E75                RTS          ;and exit
34DA|
34DA|                    ;-----
34DA|                    ;

```



```

34DA|           ; Routine to display error icon.
34DA|           ;
34DA|           ; INPUTS:
34DA|           ;     A2 = pointer to compressed icon
34DA|           ;
34DA|           ; SIDE EFFECTS:
34DA|           ;     A2/A6, D5-D6 are trashed
34DA|           ;-----
34DA|
34DA|           DSPERRICON
34DA| 6100 FC8C           BSR     MAKEALERT      ;open alert box
34DE| 7A73             MOVEQ    #ERRROW,D5     ;set screen ptrs
34E0| 7C10             MOVEQ    #ERRCOL,D6
34E2| 6100 0236        BSR     SETCRSR        ;get screen address in A6
34E6| 2A4E             MOVE.L   A6,A5         ;save it
34E8| 6100 00F8        BSR     DSPICON        ;go do display of component icon
34EC|
34EC|           DSPBAD
34EC| 45FA 079A          LEA      CHECKMRK,A2    ;get ptr to check icon
34F0| 6108             BSR.S    MRGICON        ;do the merge
34F2| 45FA 07E2          LEA      BADMRK,A2    ;get ptr to slash icon
34F6| 6102             BSR.S    MRGICON        ;merge it
34F8| 4E75             RTS                    ;and exit
34FA|
34FA|           ;-----
34FA|           ;
34FA|           ; Routine to merge two icons, one over the other.
34FA|           ;
34FA|           ; INPUTS:
34FA|           ;     A2 = ptr to icon to be merged
34FA|           ;     A5 = pointer to base icon
34FA|           ;
34FA|           ; SIDE EFFECTS:
34FA|           ;     None
34FA|           ;-----
34FA|
34FA| 48E7 F006          MRGICON MOVEM.L D0-D3/A5-A6,-(SP) ;save regs
34FE| 2C4D             MOVE.L   A5,A6           ;get start address
3500| 5C4E             ADDQ     #6,A6           ;first display new icon next to other icon
3502| 48E7 0006          MOVEM.L A5-A6,-(SP)    ;save ptrs
3506| 6100 00DA        BSR     DSPICON
350A| 4CDF 6000          MOVEM.L (SP)+,A5-A6     ;restore ptrs
350E|
350E| 7654             MOVEQ    #ROWBYTES-6,D3   ;set up row offset constant
3510| 741F             MOVEQ    #32-1,D2         ;icon heighth in pixel lines - 1
3512| 7202             @1      MOVEQ    #3-1,D1    ;icon width in words - 1
3514| 3016             @2      MOVE     (A6),D0    ;get from byte

```




```

3516| 815D          OR      D0,(A5)+      ;do the merge
3518| 425E          CLR      (A6)+        ;erase the old
351A| 51C9 FFF8      DBF      D1,@2        ;do full row
351E| DBC3          ADDA.L   D3,A5        ;bump ptrs to next row
3520| DDC3          ADDA.L   D3,A6
3522| 51CA FFEE      DBF      D2,@1        ;go to next row
3526|
3526| 4CDF 600F      MOVEM.L (SP)+,D0-D3/A5-A6 ;restore and exit
352A| 4E75          RTS
352C|
352C| ;-----
352C| ;
352C| ; Routine to display alert icon.
352C| ;
352C| ; INPUTS:
352C| ;     A2 = pointer to icon
352C| ;     MSB set if uncompressed icon, else compressed assumed   CHG008
352C| ;
352C| ; SIDE EFFECTS:
352C| ;     A6, D5-D6 are trashed
352C| ;-----
352C|
352C| DSPALRTICON
352C| 48E7 C020      MOVEM.L D0-D1/A2,-(SP) ;save regs           CHG008
3530| 6100 FC36      BSR      MAKEALERT    ;open alert box
3534| 7A73          MOVEQ     #ALRTROW,D5   ;set screen ptrs
3536| 7C10          MOVEQ     #ALRTC0L,D6
3538| 6100 01E0      BSR      SETCSR      ;get screen address in A6
353C| 220A          MOVE.L    A2,D1         ;check icon address   CHG008
353E| 6A0C          BPL.S     @1           ;skip if for compressed icon CHG008
3540| 0881 001F      BCLR     #31,D1       ;clear indicator bit   CHG008
3544| 2441          MOVE.L    D1,A2         ;and restore ptr      CHG008
3546| 6100 FEDC      BSR      DSPRGICON    ;display an uncompressed icon CHG008
354A| 6004          BRA.S     @2           ;skip to exit         CHG008
354C| 6100 0094      @1      BSR      DSPICON ;go do display       CHG008
3550| 4CDF 0403      @2      MOVEM.L (SP)+,D0-D1/A2 ;restore regs   CHG008
3554| 4E75          RTS                    ;                       CHG008
3556|
3556| ;-----
3556| ;
3556| ; Routine to display icon with question mark along side.
3556| ;
3556| ; INPUTS:
3556| ;     A2 = pointer to compressed icon
3556| ; OUTPUTS:
3556| ;     A5 = icon screen address
3556| ; SIDE EFFECTS:

```



```

3556|          ;      A2/A6, D5-D6 are trashed
3556|          ;-----
3556|
3556|      DSPQICON
3556|      BSR      MAKEALERT      ;open alert box
355A|      MOVEQ    #ERRROW,D5     ;set screen ptrs
355C|      MOVEQ    #ERRCOL,D6
355E|      BSR      SETCRSR        ;get screen address in A6
3562|      MOVE.L   A6,A5          ;save it
3564|      BSR.S    DSPICON        ;go do display of component icon
3566|      LEA      QUESTION,A2    ;get ptr to ? icon
356A|      MOVE.L   A5,A6          ;restore start address
356C|      ADDA     #6,A6          ;display next to component
3570|      BSR.S    DSPICON
3572|      RTS
3574|
3574|          ;-----
3574|          ;
3574|          ;   Routine to hilite (invert) a test icon.
3574|          ;
3574|          ; INPUTS:
3574|          ;      A1 = address of icon
3574|          ;
3574|          ; SIDE EFFECTS:
3574|          ;      None
3574|          ;-----
3574|
3574|      INVICON  MOVEM.L D0-D2,-(SP) ;save regs
3578|      MOVEQ    #ICONWIDTH,D0    ;set parms for icon
357A|      MOVEQ    #ICONHIGH,D1
357C|      MOVEQ    #-1,D2          ;set fill pattern
357E|      BSR      INVERSE          ;and go invert selected one
3582|      MOVEM.L  (SP)+,D0-D2      ;restore
3586|      RTS
3588|
3588|          ;-----
3588|          ;
3588|          ;   Routine to display test icons.
3588|          ;
3588|          ; INPUTS:
3588|          ;      None
3588|          ;
3588|          ; SIDE EFFECTS:
3588|          ;      A2/A6 trashed
3588|          ;-----
3588|
3588|      DSPCPU

```



```

3588| 45FA 0401          LEA      CPUBRD,A2          ;set ptr for CPU board icon
358C| 3C7C 1DF6          MOVEA   #CPUSTRT,A6         ;and address
3590| 601C                BRA.S   DODSPLY           ;go do display
3592|
3592| DSPMBRD
3592| 45FA 043A          LEA      MEMBRD,A2          ;set ptr for Memory board icon
3596| 3C7C 1E04          MOVEA   #MEMSTRT,A6         ;and address
359A| 6012                BRA.S   DODSPLY           ;go do display
359C|
359C| DSPIOB
359C| 45FA 03AD          LEA      IOBRD,A2           ;set ptr for I/O board icon
35A0| 3C7C 1E12          MOVEA   #IOSTRT,A6          ;and address
35A4| 6008                BRA.S   DODSPLY           ;go do display
35A6|
35A6| DSPXCRD
35A6| 45FA 046C          LEA      XCARD,A2           ;set ptr for I/O slot card icon
35AA| 3C7C 1E20          MOVEA   #XCRDSTRT,A6        ;and address
35AE|
35AE| DODSPLY
35AE| DDF8 0110          ADDA.L   SCRNBASE,A6        ;compute screen address for display
35B2| 612E                BSR.S   DSPICON           ;go do display
35B4| 4E75                RTS
35B6|
35B6| ;-----
35B6| ;
35B6| ; Routine to display icon with check mark.
35B6| ;
35B6| ; Inputs:
35B6| ;     None
35B6| ; Outputs:
35B6| ;     None
35B6| ; Side Effects:
35B6| ;     A6 trashed
35B6| ;-----
35B6|
35B6| CHKCPU
35B6| 61D0                BSR.S   DSPCPU           ;redisplay CPU icon
35B8| 3A7C 1DF6          MOVEA   #CPUSTRT,A5         ;get start address for it
35BC| 6016                BRA.S   DSPCHECK          ;and go add check mark
35BE|
35BE| CHKMBRD
35BE| 61D2                BSR.S   DSPMBRD          ;redisplay Memory board icon
35C0| 3A7C 1E04          MOVEA   #MEMSTRT,A5         ;get start address for it
35C4| 600E                BRA.S   DSPCHECK          ;and go add check mark
35C6|
35C6| CHKIOBRD
35C6| 61D4                BSR.S   DSPIOB           ;redisplay I/O icon

```



```

35C8| 3A7C 1E12          MOVEA    #IOSTRT,A5      ;get start address for it
35CC| 6006              BRA.S     DSPCHECK      ;and go add check mark
35CE|
35CE|                  CHKXCRD
35CE| 61D6              BSR.S     DSPXCRD      ;redisplay I/O slot card icon
35D0| 3A7C 1E20          MOVEA    #XCRDSTRT,A5    ;get start address for it
35D4|
35D4|                  DSPCHECK
35D4| 45FA 06B2          LEA       CHECKMRK,A2    ;get ptr to check icon
35D8| DBF8 0110          ADDA.L    SCRNBASE,A5    ;compute screen address for display
35DC| 6100 FF1C          BSR      MRGICON      ;and go do merge
35E0| 4E75              RTS
35E2|
35E2|                  ;-----
35E2|                  ;
35E2|                  ; Routine to display compressed icon.
35E2|                  ;
35E2|                  ; INPUTS:
35E2|                  ;       A2 = pointer to compressed icon
35E2|                  ;       A6 = pointer to (even!) screen address for upper left hand
35E2|                  ;               corner of icon
35E2|                  ; SIDE EFFECTS:
35E2|                  ;       A6 is trashed
35E2|                  ;-----
35E2|
35E2|                  DSPICON
35E2| 48E7 E0A0          MOVEM.L D0-D2/A0/A2,-(SP)      ;
35E6| 204E          MOVE.L    A6,A0                  ; save screen start address
35E8|
35E8|                  MOVEQ    #23,D1                ; There are 24 octals in an icon
35EA| 7405          MOVEQ    #5,D2                  ; reset row bytes counter
35EC|
35EC| 303C 0100          DLOOP   MOVE    #$100,D0      ; prime D0 for 8 bit count count
35F0| 101A          MOVE.B    (A2)+,D0              ; load map byte from compressed image
35F2|
35F2|                  MLOOP   LSR.W    #1,D0          ; shift off map bit
35F4| 6714          BEQ.S     DONE                    ; byte done when = 0
35F6| 6404          BCC.S     BLACK                  ; dispatch on the bit
35F8| 421E          CLR.B     (A6)+                  ; store zero in new
35FA| 6002          BRA.S     CHECK                  ; continue for all 8 bits
35FC| 1CDA          BLACK    MOVE.B    (A2)+,(A6)+    ; store byte in new
35FE|
35FE| 51CA FFF2          CHECK   DBF      D2,MLOOP      ; see if on scanline seam(every 6 bytes)
3602| DCFC 0054          ADDA    #90-6,A6              ; bump to next scanline
3606| 7405          MOVEQ    #5,D2                  ; reset row bytes counter
3608| 60E8          BRA.S     MLOOP                  ; continue for all 8 bits
360A|

```



```

360A| 51C9 FFE0      DONE    DBF      D1,DLOOP      ; do the rest of the octals in ICON
360E|
360E|              ; Now unXOR the icon on the screen
360E| 2C48            MOVE.L   A0,A6              ; get screen pointer saved above
3610| 244E            MOVE.L   A6,A2              ; second pointer
3612| D4FC 005A      ADDA      #90,A2              ; scanline pointer          RM015
3616|
3616| 323C 001E      MOVE      #30,D1              ; do 31 scanlines
361A|              ; This is the cause of the even destination restriction
361A| 201E      XLOOP    MOVE.L   (A6)+,D0          ; get long from previous scanline
361C| B19A            EOR.L    D0,(A2)+            ; xor into this scanline
361E| 301E            MOVE.W   (A6)+,D0          ; get word from previous scanline
3620| B15A            EOR.W    D0,(A2)+            ; xor into this scanline
3622| D4FC 0054      ADDA      #90-6,A2            ; next scan line + rowbytes      RM015
3626| DCFC 0054      ADDA      #90-6,A6            ; next scan line + rowbytes      RM015
362A|
362A| 51C9 FFEE      DBF      D1,XLOOP
362E| 4CDF 0507      MOVEM.L   (SP)+,D0-D2/A0/A2      ;          CHG009
3632| 4E75            RTS
3634|
3634|              .PAGE
3634|              ;-----
3634|              ; Subroutine to display text string according to keyboard id
3634|              ; Inputs:
3634|              ;     A3 = ptr to message
3634|              ;     D1 = nonzero if '...' string to be appended
3634|              ; Outputs:
3634|              ;     A2 = ptr to start of string
3634|              ;     A3 = ptr to end of string
3634|              ; Side Effects:
3634|              ;     D5-D6, A3 trashed
3634|              ;-----
3634|
3634|              DSPSTRING
3634| 48E7 A000      MOVEM.L   D0/D2,-(SP)          ;save regs
3638| 45FA 0887      LEA      MENUHDG,A2          ;don't translate service mode messages
363C| B7CA            CMPA.L   A2,A3
363E| 6770            BEQ.S    DSPOUT              ;skip if it is
3640| 244B            MOVE.L   A3,A2              ;else save starting point
3642| 1038 01B2      MOVE.B    KEYID,D0          ;get keyboard id
3646| 6768            BEQ.S    DSPOUT              ;skip if no id available
3648| 0200 003F      ANDI.B    #$3F,D0            ;clear mfg code
364C| 1400            MOVE.B    D0,D2              ;move to working reg
364E|
364E|              ; Search for US, UK or Canadian keyboard
364E|
364E| 0202 00F0      ANDI.B    #$F0,D2            ;old US keyboard?

```



```

3652| 675C          BEQ.S  DSPOUT      ;yes - go do English display
3654| 0C02 0030      CMPI.B  #$30,D2    ;US or Canadian layout?
3658| 6608          BNE.S   @1
365A| 0C00 003D      CMPI.B  #$3D,D0    ;Canadian?
365E| 6726          BEQ.S  DSPALL      ;yes - display all languages
3660| 604E          BRA.S   DSPOUT      ;else just English
3662|
3662| 0C02 0020      @1    CMPI.B  #$20,D2    ;European keyboard?
3666| 661E          BNE.S   DSPALL      ;no - display all languages
3668| 0C00 002F      CMPI.B  #$2F,D0    ;UK?
366C| 6742          BEQ.S  DSPOUT      ;yes - display English
366E|
366E|              ; Search for German type keyboard
366E|
366E| 0C00 002E          CMPI.B  #$2E,D0    ;German?
3672| 6732          BEQ.S  DSPGERMN
3674| 0C00 0026      CMPI.B  #$26,D0    ;Swiss-German?
3678| 672C          BEQ.S  DSPGERMN
367A|
367A|              ; Search for French type keyboard
367A| 0C00 002D      CMPI.B  #$2D,D0    ;French?
367E| 672A          BEQ.S  DSPFRNCH
3680| 0C00 0027      CMPI.B  #$27,D0    ;Swiss-French?
3684| 6724          BEQ.S  DSPFRNCH
3686|
3686|              ; Display all languages for any other keyboard (e.g., Italian, Spanish, etc.)
3686|
3686| 0838 0007 02A2    DSPALL  BTST    #MENU,STATFLGS ;doing menu?
368C| 6612          BNE.S   @1            ;skip if yes
368E| 0445 000A      SUB     #CHRSPC,D5    ;back up one row
3692| 6124          BSR.S   DSPIT          ;display English string
3694| 0645 000A      ADD     #CHRSPC,D5    ;incr to next row
3698| 611E          BSR.S   DSPIT          ;display French translation
369A| 0645 000A      ADD     #CHRSPC,D5    ;bump another row
369E| 6010          BRA.S   DSPOUT          ;go do final display of German
36A0|
36A0| 6130          @1    BSR.S   DSPMSLSH    ;display English followed by /
36A2| 612E          BSR.S   DSPMSLSH    ;display French followed by /
36A4| 600A          BRA.S   DSPOUT          ;and go do final German display
36A6|
36A6|              DSPGERMN
36A6| 4A1B          TST.B   (A3)+          ;skip two strings before output
36A8| 66FC          BNE.S   DSPGERMN
36AA|
36AA|              DSPFRNCH
36AA| 4A1B          TST.B   (A3)+          ;skip one string before output
36AC| 66FC          BNE.S   DSPFRNCH

```



```

36AE| 244B                MOVE.L  A3,A2                ;save new beginning ptr
36B0|
36B0| 6106                DSPOUT  BSR.S  DSPIT                ;do display
36B2| 4CDF 0005          MOVEM.L  (SP)+,D0/D2          ;restore regs
36B6| 4E75                RTS                            ; and exit
36B8|
36B8| ;-----
36B8| ; Subroutine to display text string followed by '...'
36B8| ; Inputs:
36B8| ;     A3 = ptr to message
36B8| ;     D1 = nonzero if '...' string to be appended
36B8| ; Outputs:
36B8| ;     None
36B8| ; Side Effects:
36B8| ;     A3 updated to next location after 0 byte
36B8| ;-----
36B8|
36B8| 48E7 0600          DSPIT  MOVEM.L  D5-D6,-(SP)        ;save cursor ptrs
36BC| 6142                BSR.S  DSPMSG                ;output message
36BE| 2F0B                MOVE.L  A3,-(SP)                ;save msg ptr
36C0| 4A41                TST     D1                    ;check if periods needed
36C2| 6706                BEQ.S  @9                    ;skip if not
36C4| 47FA 07F7          LEA     PERIODS,A3              ;else do display
36C8| 6136                BSR.S  DSPMSG
36CA| @9
36CA| 265F                MOVE.L  (SP)+,A3                ;restore regs and exit
36CC| 4CDF 0060          MOVEM.L  (SP)+,D5-D6
36D0| 4E75                RTS
36D2|
36D2| ;-----
36D2| ; Subroutine to display text string followed by '/'
36D2| ; Inputs:
36D2| ;     A3 = ptr to message
36D2| ; Outputs:
36D2| ;     None
36D2| ; Side Effects:
36D2| ;     A3 updated to next location after 0 byte
36D2| ;-----
36D2|
36D2| DSPMSLSH
36D2| 612C                BSR.S  DSPMSG                ;output message
36D4| 702F                MOVEQ   #'/',D0                ;display /
36D6| 6162                BSR.S  DSPVAL
36D8| 4E75                RTS
36DA|
36DA| ;-----
36DA| ; Subroutine to display alert box message

```



```

36DA|      ; Inputs:
36DA|      ;   A3 = ptr to message
36DA|      ; Outputs:
36DA|      ;   None
36DA|      ;-----
36DA|      DSPALRTMSG
36DA| 48E7 0E00      MOVEM.L D4-D6,-(SP)      ;save regs      CHG005
36DE| 3A3C 007E      MOVE      #MSGROW,D5      ;set screen ptrs
36E2| 7C18      MOVEQ     #MSGCOL,D6
36E4| 3805      MOVE      D5,D4      ;set left margin in case of CR      CHG005
36E6| 6118      BSR.S     DSPMSG      ;go do display
36E8| 4CDF 0070      MOVEM.L (SP)+,D4-D6      ;restore regs      CHG005
36EC| 4E75      RTS
36EE|
36EE|      ;-----
36EE|      ; Routine to convert row coordinate to pixel row coordinate before
36EE|      ; doing message display
36EE|      ;
36EE|      ; Expects D5 = row coordinate from 0-32
36EE|      ;-----
36EE|      CONVRTD5
36EE| CAFC 000A      MULU      #10,D5      ;multiply by pixel lines per char row
36F2| 610C      BSR.S     DSPMSG      ;then go do message
36F6|      RTS
36F6|      .ENDC      ;{USERINT}
36F6|      ;-----
36F6|      ; Subroutine to display message followed by a line feed, CR.
36F6|      ; Calls DSPMSG routine, with same assumptions as that routine.
36F6|      ;-----
36F6| 6108      DSPMSGR BSR.S     DSPMSG      ;go display msg
36F8|
36F8|      .IF USERINT = 0
36F8|      .ELSE
36F8| 0645 000A      ADD      #CHRSPC,D5      ;do "line feed"
36FC|      .ENDC
36FC|
36FC| 7C01      MOVEQ     #1,D6      ;and "carriage return"
36FE| 4E75      RTS
3700|
3700|      ;-----
3700|      ; Subroutine to display message on screen.
3700|      ; Requires inputs:
3700|      ;   A3 - ptr to ASCII character string ended by 0 byte
3700|      ;   D4 = left margin if message has a CR

```




```

3700|          ;      D5 = cursor row position (0 - 32 decimal)
3700|          ;      D6 = cursor column position (1 - 88 decimal)
3700|          ; Uses regs:
3700|          ;      D0 - for character to display
3700|          ;
3700|          ; NOTE: ONLY UPPER CASE ALPHA, NUMERIC AND CERTAIN SPECIAL CHARS SUPPORTED!
3700|          ;-----
3700| 3700| 2F00          DSPMSG  MOVE.L  D0,-(SP)          ;save reg
3702|
3702|          .IF      ROM4K = 0
3702| 3702| 0C45 014C      @1      CMPI      #LASTROW,D5          ;check if out of bounds
3706| 3706| 6F04          BLE.S      @2          ;skip if OK
3708| 3708| 6100 F3FE      BSR      SCROLL          ;else scroll page
370C|          .ENDC
370C|
370C| 370C| 4280          @2      CLR.L      D0          ;clear for use
370E| 370E| 101B          MOVE.B      (A3)+,D0          ;get a char to display
3710| 3710| 6704          BEQ.S      DSPDONE          ;exit if done
3712| 3712| 6126          BSR.S      DSPVAL          ;go do display
3714| 3714| 60EC          BRA.S      @1          ;continue until done
3716| 3716| 201F      DSPDONE MOVE.L      (SP)+,D0          ;restore and exit
3718| 3718| 4E75          RTS
371A|
371A|          .PAGE
371A|          ;-----
371A|          ; Subroutine to set cursor position in video page for message display.
371A|          ; Requires inputs:
371A|          ;      D5 = row position (0 - 32 decimal)
371A|          ;      D6 = column position (1 - 88 decimal)
371A|          ;      location SCRNBASE = base address for video page
371A|          ; Provides output:
371A|          ;      A6 = address for new cursor location
371A|          ;-----
371A|
371A|          SETCSR
371A| 371A| 2C78 0110      MOVE.L      SCRNBASE,A6          ;get base address for screen
371E|          SETCSR2
371E| 371E| 2F00          MOVE.L      D0,-(SP)          ;save reg
3720| 3720| 4280          CLR.L      D0          ;use as working reg
3722|
3722|          .IF      USERINT = 0
3722|          .ELSE
3722| 3722| 3005          MOVE      D5,D0          ;get pixel row
3724|          .ENDC
3724| 3724| C0FC 005A      MULU      #RBYTES,D0          ;compute byte offset
3728|

```



```

3728|          ;      .IF USERINT = 0
3728| 0640 005A      ADD      #RBYTES,D0          ;add one more
372C|          ;      .ELSE
372C|          ;      ADD      #TOPOFFSET,D0      ;adjust for offset from top of screen
372C|          ;      .ENDC
372C|
372C| D08E          ADD.L   A6,D0          ;add in base screen address
372E| 2C40          MOVE.L  D0,A6          ;and save new value
3730| 4280          CLR.L   D0
3732|
3732|          .IF USERINT = 0
3732|          .ELSE
3732| 3006          MOVE     D6,D0          ;get pixel col
3734|          .ENDC
3734|
3734| DDC0          ADDA.L  D0,A6          ;add to cursor address
3736| 201F          MOVE.L  (SP)+,D0      ;restore and exit
3738| 4E75          RTS
373A|
373A|          ;-----
373A|          ; Subroutine to display single ASCII character.
373A|          ; Requires input:
373A|          ;      D0 = character to display
373A|          ;      D4 = left margin if CR char
373A|          ;      D5 = cursor row position (0 - 32 decimal)
373A|          ;      D6 = cursor column position (1 - 88 decimal)
373A|          ; Returns output:
373A|          ;      D6 = new cursor col position (incremented by 1)
373A|          ;-----
373A|
373A| 48E7 C022      DSPVAL  MOVEM.L A2/A6/D0-D1,-(SP)      ;save regs
373E| 61DA          BSR.S   SETCRSR          ;set cursor position
3740| 0240 007F      ANDI    #$7F,D0        ;ensure valid
3744|          .IF USERINT = 1
3744| 4A00          TST.B   D0          ;apple icon?
3746| 6738          BEQ.S   @4
3748|          .ENDC
3748|
3748|          .IF ROM4K = 0
3748| 0C00 0020      CMPI.B  #$20,D0        ;space?
374C| 672C          BEQ.S   @3          ;skip if yes
374E| 0C00 000D      CMPI.B  #RET,D0       ;carriage return?
3752| 6732          BEQ.S   @5          ;skip if yes
3754| 0C00 003F      CMPI.B  #QUESTN,D0    ;'?' char?
3758| 6734          BEQ.S   @6
375A|
375A| 0400 002D      SUB.B   #$2D,D0        ;else check if in table

```



```

375E| 6D14          BLT.S    @1
3760| 0C00 000C      CMPI.B   #$C,D0          ;numeric char?
3764| 6F2E          BLE.S    @2          ;skip if yes
3766| 5F00          SUB.B    #7,D0          ;else decr for alpha check
3768| 0C00 000D      CMPI.B   #$D,D0          ;check if in alpha range
376C| 6D06          BLT.S    @1          ;skip if invalid
376E| 0C00 0026      CMPI.B   #$26,D0        ;last valid char = 'Z'
3772| 6F20          BLE.S    @2
3774|                                     ;skip if OK
3774|          .ELSE
3774|          .ENDC
3774|
3774|          .IF  USERINT = 0
3774|          .ELSE
3774|
3774| 45FA 013A      @1    LEA     INVCHAR,A2          ;else set for invalid character
3778| 6026          BRA.S    OUT
377A|
377A| 45FA 003E      @3    LEA     SPACE,A2          ;set ptr to space char
377E| 6020          BRA.S    OUT
3780|
3780| 45FA 0134      @4    LEA     APPLICON,A2        ; display apple icon
3784| 601A          BRA.S    OUT
3786|
3786| 0645 000A      @5    ADD     #CHRSPC,D5        ;set cursor for next row
378A| 3C04          MOVE     D4,D6          ;set to left margin
378C| 6026          BRA.S    DSPVXIT
378E|
378E| 45FA 011A      @6    LEA     QUESTCH,A2        ;set ptr to '?' char
3792| 600C          BRA.S    OUT
3794|
3794| 3200          @2    MOVE     D0,D1          ;convert for table (multiply by 6)  CHG017
3796| E748          LSL      #3,D0          ;mult by 8
3798| 9041          SUB      D1,D0          ; then subtract twice
379A| 9041          SUB      D1,D0          ;
379C|                                     CHG017
379C| 45FB 0022      LEA      FONTTBL(D0) ,A2        ;get ptr to char in table
37A0|
37A0| 4280          out    clr.l    d0          ;set number of bytes-1 in x direction
37A2| 7205          moveq   #5,d1          ;set number of pixels-1 in y direction  CHG017
37A4| 4216          MOVE.B  #0,R0 (A6)        ;first line always 0  CHG017
37A6| DDFC 0000 005A  ADDA.L  #RBYTES,A6        ;bump ptr to next row  CHG017
37AC| 6100 FC7E      bsr      output          ;output char
37B0| 422E 0276      MOVE.B  #0,R7 (A6)        ;and add final 0 byte
37B4|          .ENDC
37B4|
37B4|          DSPVXIT

```



```

37B4| 4CDF 4403          MOVEM.L (SP)+,A2/A6/D0-D1      ;restore and exit
37B8| 4E75              RTS
37BA|
37BA|          .IF      EXTERNAL = 1
37BA|          .ENDC
37BA|          .PAGE
37BA|          ;-----
37BA|          ; CHARACTER FONT TABLE
37BA|          ;-----
37BA|          SPACE
37BA| 00 00 00 00 00 00      .BYTE      $00,$00,$00,$00,$00,$00      ; (space)
37C0|
37C0|          FONTTBL
37C0|          .IF ROM4K = 0
37C0| 00 00 00 7C 00 00      .BYTE      $00,$00,$00,$7C,$00,$00      ; code = 45 '-'
37C6| 00 00 00 00 00 30      .BYTE      $00,$00,$00,$00,$00,$30      ; code = 46 '.'
37CC| 04 08 10 20 40 80      .BYTE      $04,$08,$10,$20,$40,$80      ; code = 47 '/'
37D2|          .ENDC
37D2|
37D2| 38 44 44 44 44 38      .BYTE      $38,$44,$44,$44,$44,$38      ; code = 48 '0'
37D8| 08 38 08 08 08 08      .BYTE      $08,$38,$08,$08,$08,$08      ; code = 49 '1'
37DE| 38 44 08 10 20 7C      .BYTE      $38,$44,$08,$10,$20,$7C      ; code = 50 '2'
37E4| 38 44 18 04 44 38      .BYTE      $38,$44,$18,$04,$44,$38      ; code = 51 '3'
37EA| 08 18 28 48 7C 08      .BYTE      $08,$18,$28,$48,$7C,$08      ; code = 52 '4'
37F0| 7C 40 78 04 44 38      .BYTE      $7C,$40,$78,$04,$44,$38      ; code = 53 '5'
37F6| 38 40 78 44 44 38      .BYTE      $38,$40,$78,$44,$44,$38      ; code = 54 '6'
3802| 38 44 38 44 44 38      .BYTE      $38,$44,$38,$44,$44,$38      ; code = 56 '8'
3808| 38 44 44 3C 04 38      .BYTE      $38,$44,$44,$3C,$04,$38      ; code = 57 '9'
380E|
380E| 30 48 84 FC 84 84      .BYTE      $30,$48,$84,$FC,$84,$84      ; code = 65 'A'
3814| F8 84 F8 84 84 F8      .BYTE      $F8,$84,$F8,$84,$84,$F8      ; code = 66 'B'
381A| 78 84 80 80 84 78      .BYTE      $78,$84,$80,$80,$84,$78      ; code = 67 'C'
3820| F8 84 84 84 84 F8      .BYTE      $F8,$84,$84,$84,$84,$F8      ; code = 68 'D'
3826| FC 80 F8 80 80 FC      .BYTE      $FC,$80,$F8,$80,$80,$FC      ; code = 69 'E'
382C| FC 80 F8 80 80 80      .BYTE      $FC,$80,$F8,$80,$80,$80      ; code = 70 'F'
3832| 78 84 80 9C 84 7C      .BYTE      $78,$84,$80,$9C,$84,$7C      ; code = 71 'G'
3838| 84 84 FC 84 84 84      .BYTE      $84,$84,$FC,$84,$84,$84      ; code = 72 'H'
383E| 38 10 10 10 10 38      .BYTE      $38,$10,$10,$10,$10,$38      ; code = 73 'I'
3844| 1C 08 08 08 88 70      .BYTE      $1C,$08,$08,$08,$88,$70      ; code = 74 'J'
384A| 88 90 A0 D0 88 84      .BYTE      $88,$90,$A0,$D0,$88,$84      ; code = 75 'K'
3850| 80 80 80 80 80 FC      .BYTE      $80,$80,$80,$80,$80,$FC      ; code = 76 'L'
3856| 84 CC B4 84 84 84      .BYTE      $84,$CC,$B4,$84,$84,$84      ; code = 77 'M'
385C| 84 C4 A4 94 8C 84      .BYTE      $84,$C4,$A4,$94,$8C,$84      ; code = 78 'N'
3862| 78 84 84 84 84 78      .BYTE      $78,$84,$84,$84,$84,$78      ; code = 79 'O'
3868| F8 84 84 F8 80 80      .BYTE      $F8,$84,$84,$F8,$80,$80      ; code = 80 'P'
386E| 78 84 84 84 94 78      .BYTE      $78,$84,$84,$84,$94,$78      ; code = 81 'Q'
3874| F8 84 84 F8 88 84      .BYTE      $F8,$84,$84,$F8,$88,$84      ; code = 82 'R'

```

```

387A| 78 84 60 18 84 78      .BYTE    $78,$84,$60,$18,$84,$78      ; code = 83 'S'
3880| FE 10 10 10 10 10      .BYTE    $FE,$10,$10,$10,$10,$10      ; code = 84 'T'
3886| 84 84 84 84 84 78      .BYTE    $84,$84,$84,$84,$84,$78      ; code = 85 'U'
388C| 44 44 28 28 10 10      .BYTE    $44,$44,$28,$28,$10,$10      ; code = 86 'V'
3892| 82 82 92 AA 44 44      .BYTE    $82,$82,$92,$AA,$44,$44      ; code = 87 'W'
3898| 44 28 10 28 44 82      .BYTE    $44,$28,$10,$28,$44,$82      ; code = 88 'X'
389E| 82 44 28 10 10 10      .BYTE    $82,$44,$28,$10,$10,$10      ; code = 89 'Y'
38A4| FC 08 10 20 40 FC      .BYTE    $FC,$08,$10,$20,$40,$FC      ; code = 90 'Z'
38AA|
38AA|      .IF ROM4K = 0
38AA|      QUESTCH
38AA| 38 44 08 10 00 10      .BYTE    $38,$44,$08,$10,$00,$10      ; code = 63 '?'
38B0|      .ENDC
38B0|
38B0|      INVCHAR
38B0| C7 BB F7 EF FF EF      .BYTE    $C7,$BB,$F7,$EF,$FF,$EF      ; inverse of ?
38B6|
38B6|      .IF ROM4K = 0
38B6|      APPLICON
38B6| 08 77 FE FE 7F 3E      .BYTE    $08,$77,$FE,$FE,$7F,$3E      ; apple icon
38BC|
38BC|      .PAGE
38BC|      .ALIGN 2
38BC|
38BC|      ;-----
38BC|      ; Keycode to Ascii Table (assumes alpha-lock so upper case only)
38BC|      ;-----
38BC|
38BC|      AsciiTable
38BC| 1B 2D 11 12 37 38 39      .BYTE    $1B,$2D,$11,$12,$37,$38,$39,$14      ;
38C3| 14                        ;Pad : Clear - Left Right 789 Up
38C4| 34 35 36 13 2E 32 33      .BYTE    $34,$35,$36,$13,$2E,$32,$33,$03      ;Pad : 456 Down .23 Enter
38CB| 03
38CC| 00 00 00 00 00 00 00      .BYTE    $00,$00,$00,$00,$00,$00,$00,$00      ;unused
38D3| 00
38D4| 00 00 00 00 00 00 00      .BYTE    $00,$00,$00,$00,$00,$00,$00,$00      ;unused
38DB| 00
38DC| 2D 3D 00 00 50 08 00      .BYTE    $2D,$3D,$00,$00,$50,$08,$00,$00      ;- = 2*unused P BackSp 2*unused
38E3| 00
38E4| 0D 30 00 00 2F 31 00      .BYTE    $0D,$30,$00,$00,$2F,$31,$00,$00      ;Ret Pad:0 2*unused / Pad:1 2*unused
38EB| 00
38EC| 39 30 55 49 4A 4B 5B      .BYTE    $39,$30,$55,$49,$4A,$4B,$5B,$5D      ;90UIJK[]
38F3| 5D
38F4| 4D 4C 3B 27 20 2C 2E      .BYTE    $4D,$4C,$3B,$27,$20,$2C,$2E,$4F      ;ML;' Space ,.O
38FB| 4F
38FC| 45 36 37 38 35 52 54      .BYTE    $45,$36,$37,$38,$35,$52,$54,$59      ;E6785RTY
3903| 59
3904| 00 46 47 48 56 43 42      .BYTE    $00,$46,$47,$48,$56,$43,$42,$4E      ;Option FGHVCBN

```

RM000



```

390B| 4E
390C| 41 32 33 34 31 51 53 .BYTE $41,$32,$33,$34,$31,$51,$53,$57 ;A2341QSW
3914| 00 5A 58 44 00 00 00 .BYTE $00,$5A,$58,$44,$00,$00,$00,$00 ;Tab ZXD unused Alpha Shift Cmd
391C|
391C| .ENDC
391C| .IF USERINT = 1
391C| ;-----
391C| ; Icons
391C| ;-----
391C|
391C| CrsrData ;arrow for mouse cursor
391C| CrsrMask ;same for mask
391C| 8000 C000 E000 F000 .WORD $8000,$C000,$E000,$F000
3924| F800 FC00 FE00 FF00 .WORD $F800,$FC00,$FE00,$FF00
392C| F800 F800 CC00 8C00 .WORD $F800,$F800,$CC00,$8C00
3934| 0600 0600 0300 0300 .WORD $0600,$0600,$0300,$0300
393C|
393C| E0 60 60 60 F0 ONE .BYTE $E0,$60,$60,$60,$F0 ;icon id = 1
3941| E0 30 60 C0 F0 TWO .BYTE $E0,$30,$60,$C0,$F0 ;icon id = 2
3946| E0 30 60 30 E0 THREE .BYTE $E0,$30,$60,$30,$E0 ;icon id = 3
394B|
394B| FF FF FF 3F 03 FF 00 IObrd .BYTE $FF,$FF,$FF,$3F,$03,$FF,$00,$FF
3952| FF
3953| FF FF E0 01 FF FF FF .BYTE $FF,$FF,$E0,$01,$FF,$FF,$FF,$E4
395A| E4
395B| FF C0 70 2F F9 50 66 .BYTE $FF,$C0,$70,$2F,$F9,$50,$66,$9E
3962| 9E
3963| C0 01 80 F7 03 79 56 .BYTE $C0,$01,$80,$F7,$03,$79,$56,$06
396A| 06
396B| 74 FE 0F FF FF FF FF .BYTE $74,$FE,$0F,$FF,$FF,$FF,$FF,$FF
3973| FF FF 3F 01 AA 60 AA .BYTE $FF,$FF,$3F,$01,$AA,$60,$AA,$AF
397A| AF
397B| FF C0 03 07 C4 FF E0 .BYTE $FF,$C0,$03,$07,$C4,$FF,$E0,$55
3983| 55 50 F1 FF FF F8 FF .BYTE $55,$50,$F1,$FF,$FF,$F8,$FF,$FF
398A| FF
398B|
398B| FF FF FF 3F 03 FF 00 CPUbrd .BYTE $FF,$FF,$FF,$3F,$03,$FF,$00,$FF
3992| FF
3993| FF FF E0 01 FF FF FF .BYTE $FF,$FF,$E0,$01,$FF,$FF,$FF,$C4
399A| C4
399B| FF C0 79 F2 40 F9 38 .BYTE $FF,$C0,$79,$F2,$40,$F9,$38,$E0
39A2| E0
39A3| BF E0 EF F0 71 38 01 .BYTE $BF,$E0,$EF,$F0,$71,$38,$01,$80
39AA| 80
39AB| 79 FC 03 C0 FF FF FF .BYTE $79,$FC,$03,$C0,$FF,$FF,$FF,$FF
39B2| FF
39B3| FF FF FF 3F 01 FE C0 .BYTE $FF,$FF,$FF,$3F,$01,$FE,$C0,$AA

```



39BA	AA		
39BB	AA BF C0 03 FC 84 1F		.BYTE \$AA,\$BF,\$C0,\$03,\$FC,\$84,\$1F,\$E0
39C2	E0		
39C3	01 55 55 40 E1 03 FF		.BYTE \$01,\$55,\$55,\$40,\$E1,\$03,\$FF,\$FF
39CE			.BYTE \$E0,\$FF,\$FF
39CE	FF FF FF FF FF FF C3	MEMbrd	.BYTE \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$C3,\$1F
39D5	1F		
39D6	FF FF C0 80 0F FF FF		.BYTE \$FF,\$FF,\$C0,\$80,\$0F,\$FF,\$FF,\$80
39DE	07 FF E0 E3 03 FF F0		.BYTE \$07,\$FF,\$E0,\$E3,\$03,\$FF,\$F0,\$FF
39E5	FF		
39E6	FF FF 71 04 5E 88 02		.BYTE \$FF,\$FF,\$71,\$04,\$5E,\$88,\$02,\$1C
39ED	1C		
39EE	8E 50 03 8C 70 C7 01		.BYTE \$8E,\$50,\$03,\$8C,\$70,\$C7,\$01,\$0C
39F5	0C		
39F6	20 7B 0E 04 0C 5E 88		.BYTE \$20,\$7B,\$0E,\$04,\$0C,\$5E,\$88,\$03
39FE	FF D5 57 00 FF 80 07		.BYTE \$FF,\$D5,\$57,\$00,\$FF,\$80,\$07,\$FF
3A06	80 03 FF C0 F3 2A A8		.BYTE \$80,\$03,\$FF,\$C0,\$F3,\$2A,\$A8,\$FC
3A0D	FC		
3A0E	7F FC FF FF FF FF		.BYTE \$7F,\$FC,\$FF,\$FF,\$FF,\$FF,\$FF
3A14			
3A14	FF FF FF FF FF FF E1	Xcard	.BYTE \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$E1,\$01
3A1C	FF FF 80 FC FF FF FF		.BYTE \$FF,\$FF,\$80,\$FC,\$FF,\$FF,\$FF,\$FF
3A24	FF FF FF FF FF FF FF		.BYTE \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF
3A2C	FF FF DF 03 63 EA AD		.BYTE \$FF,\$FF,\$DF,\$03,\$63,\$EA,\$AD,\$80
3A34	01 3C C0 06 15 50 CF		.BYTE \$01,\$3C,\$C0,\$06,\$15,\$50,\$CF,\$3F
3A3B	3F		
3A3C	F8		.BYTE \$F8
3A3D	FF FF FF 61 1F FF FF	waiticon	.BYTE \$FF,\$FF,\$FF,\$61,\$1F,\$FF,\$FF,\$F8
3A45	0F 18 FF FF F0 0F FF		.BYTE \$0F,\$18,\$FF,\$FF,\$F0,\$0F,\$FF,\$FF
3A4D	86 F0 11 FF FF 88 6D		.BYTE \$86,\$F0,\$11,\$FF,\$FF,\$88,\$6D,\$05
3A54	05		
3A55	A0 02 18 80 01 40 01		.BYTE \$A0,\$02,\$18,\$80,\$01,\$40,\$01,\$4F
3A5C	4F		
3A5D	F2 CE 80 A8 15 F3 54		.BYTE \$F2,\$CE,\$80,\$A8,\$15,\$F3,\$54,\$2A
3A64	2A		
3A65	3C 2A 54 14 28 CF 08		.BYTE \$3C,\$2A,\$54,\$14,\$28,\$CF,\$08,\$10
3A6C	10		
3A6D	FF 3C 08 10 14 28 CF		.BYTE \$FF,\$3C,\$08,\$10,\$14,\$28,\$CF,\$28
3A74	28		
3A75	14 F3 50 0A 1C A0 05		.BYTE \$14,\$F3,\$50,\$0A,\$1C,\$A0,\$05,\$01
3A7C	01		
3A7D	46 62 86 80 02 88 11		.BYTE \$46,\$62,\$86,\$80,\$02,\$88,\$11,\$40
3A84	40		
3A85	61 05 10 08 A0 11 18		.BYTE \$61,\$05,\$10,\$08,\$A0,\$11,\$18,\$E0
3A8C	E0		
3A8D	07 88 0F FF FF 86 F0		.BYTE \$07,\$88,\$0F,\$FF,\$FF,\$86,\$F0,\$0F
3A94	0F		



3A95	FF FF F0 E1 1F FF FF		.BYTE \$FF,\$FF,\$F0,\$E1,\$1F,\$FF,\$FF,\$F8	
3A9C	F8			
3A9D	FF FF		.BYTE \$FF,\$FF	
3A9F				
3A9F	FF FF FF FF FF FF FF	proicon	.BYTE \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF	
3AA6	FF			
3AA7	FF 00 1F FF FF FF FF		.BYTE \$FF,\$00,\$1F,\$FF,\$FF,\$FF,\$FF,\$F8	
3AAE	F8			
3AAF	3F FF F0 FF FF FF FC		.BYTE \$3F,\$FF,\$F0,\$FF,\$FF,\$FF,\$FC,\$FF	
3AB7	CC 03 7F 01 80 CF 03		.BYTE \$CC,\$03,\$7F,\$01,\$80,\$CF,\$03,\$7F	
3ABE	7F			
3ABF	FC 01 80 00 3F FF FF		.BYTE \$FC,\$01,\$80,\$00,\$3F,\$FF,\$FF,\$FF	
3AC6	FF			
3AC7	FF FC 1C FF 60 FF F3		.BYTE \$FF,\$FC,\$1C,\$FF,\$60,\$FF,\$F3,\$FF	
3ACE	FF			
3ACF	38 03 0C FD C0 FF FF		.BYTE \$38,\$03,\$0C,\$FD,\$C0,\$FF,\$FF,\$FF	
3AD6	FF			
3AD7	FF FF FF		.BYTE \$FF,\$FF,\$FF	
3ADA	FF FF FF FF FF FF FF	upper	.BYTE \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF	;CHG024
3AE1	FF			
3AE2	FF 2F 08 1F FF 08 FF		.BYTE \$FF,\$2F,\$08,\$1F,\$FF,\$08,\$FF,\$FC	
3AE9	FC			
3AEA	10 1F FF FF FC 02 20		.BYTE \$10,\$1F,\$FF,\$FF,\$FC,\$02,\$20,\$1F	
3AF1	1F			
3AF2	FF FF FC 47 F8 20 1F		.BYTE \$FF,\$FF,\$FC,\$47,\$F8,\$20,\$1F,\$FF	
3AF9	FF			
3AFA	FF FC 80 1F FF 08 FF		.BYTE \$FF,\$FC,\$80,\$1F,\$FF,\$08,\$FF,\$FC	
3B01	FC			
3B02	80 1F FF FF FC 80 47		.BYTE \$80,\$1F,\$FF,\$FF,\$FC,\$80,\$47,\$F8	
3B09	F8			
3B0A	1F FF FF FC 20 E0 1F		.BYTE \$1F,\$FF,\$FF,\$FC,\$20,\$E0,\$1F,\$FF	
3B11	FF			
3B12	FF FC 10 FB 08 FF FF		.BYTE \$FF,\$FC,\$10,\$FB,\$08,\$FF,\$FF,\$FF	
3B19	FF			
3B1A	FF FF FF FF		.BYTE \$FF,\$FF,\$FF,\$FF	
3B1E				
3B1E	FF FF FF FF FF FF FF	driven	.BYTE \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF	;CHG024
3B25	FF			
3B26	FF EF 02 38 03 FF 04		.BYTE \$FF,\$EF,\$02,\$38,\$03,\$FF,\$04,\$07	
3B2D	07			
3B2E	FF 22 88 3F FF F8 11		.BYTE \$FF,\$22,\$88,\$3F,\$FF,\$F8,\$11,\$FE	
3B35	FE			
3B36	E8 7F FF FC 20 8B 20		.BYTE \$E8,\$7F,\$FF,\$FC,\$20,\$8B,\$20,\$7F	
3B3D	7F			
3B3E	FF FC A0 11 FE 3F FF		.BYTE \$FF,\$FC,\$A0,\$11,\$FE,\$3F,\$FF,\$F8	
3B46	08 E3 07 FF 84 F8 03		.BYTE \$08,\$E3,\$07,\$FF,\$84,\$F8,\$03,\$FF	
3B4D	FF			



3B4E	02 FF FF FF FF FF FF		.BYTE \$02,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF
3B55	FF		
3B56			
3B56	FF 79 07 FE 0F C6 FF	insertd	.BYTE \$FF,\$79,\$07,\$FE,\$0F,\$C6,\$FF,\$7F
3B5D	7F		
3B5E	FF F0 71 FF FF F8 0A		.BYTE \$FF,\$F0,\$71,\$FF,\$FF,\$F8,\$0A,\$18
3B65	18		
3B66	AA AA A8 FA AA AA F6		.BYTE \$AA,\$AA,\$A8,\$FA,\$AA,\$AA,\$F6,\$A0
3B6D	A0		
3B6E	60 E3 03 C0 07 38 04		.BYTE \$60,\$E3,\$03,\$C0,\$07,\$38,\$04,\$20
3B75	20		
3B76	06 08 10 FF F3 08 10		.BYTE \$06,\$08,\$10,\$FF,\$F3,\$08,\$10,\$3C
3B7D	3C		
3B7E	04 20 03 C0 FF FF FF		.BYTE \$04,\$20,\$03,\$C0,\$FF,\$FF,\$FF,\$FF
3B85	FF		
3B86	61 1F FF FF F8 0F 38		.BYTE \$61,\$1F,\$FF,\$FF,\$F8,\$0F,\$38,\$FF
3B8D	FF		
3B8E	FF F0 01 80 CF 02 40		.BYTE \$FF,\$F0,\$01,\$80,\$CF,\$02,\$40,\$F3
3B95	F3		
3B96	04 20 3C 08 10 10 08		.BYTE \$04,\$20,\$3C,\$08,\$10,\$10,\$08,\$CF
3B9E	1C 38 FF FF CF 03 C0		.BYTE \$1C,\$38,\$FF,\$FF,\$CF,\$03,\$C0
3BA5	CF 01 C0 FF 3F 0E 38	keybdout	.BYTE \$CF,\$01,\$C0,\$FF,\$3F,\$0E,\$38,\$CF
3BAD	08 08 F3 04 10 3C 02		.BYTE \$08,\$08,\$F3,\$04,\$10,\$3C,\$02,\$20
3BB4	20		
3BB5	01 40 DF 80 F3 01 C0		.BYTE \$01,\$40,\$DF,\$80,\$F3,\$01,\$C0,\$3F
3BBC	3F		
3BBD	01 40 20 1B 6D B6 DB		.BYTE \$01,\$40,\$20,\$1B,\$6D,\$B6,\$DB,\$60
3BC4	60		
3BC5	16 DB 08 6D B6 DF 3B		.BYTE \$16,\$DB,\$08,\$6D,\$B6,\$DF,\$3B,\$6D
3BCC	6D		
3BCD	B6 DB 80 67 80 06 DB		.BYTE \$B6,\$DB,\$80,\$67,\$80,\$06,\$DB,\$6D
3BD4	6D		
3BD5	B6 D8 FF 0F 4F FF FF		.BYTE \$B6,\$D8,\$FF,\$0F,\$4F,\$FF,\$FF,\$FF
3BDC	FF		
3BDD	00 FF F0 FF FF FF FF		.BYTE \$00,\$FF,\$F0,\$FF,\$FF,\$FF,\$FF,\$FF
3BE5	F8 00 1B 6D B6 D8 36		.BYTE \$F8,\$00,\$1B,\$6D,\$B6,\$D8,\$36,\$C0
3BED	1B 6D 00 B6 D8 36 C0		.BYTE \$1B,\$6D,\$00,\$B6,\$D8,\$36,\$C0,\$1B
3BF5	6D B6 D8 00 36 C0 1B		.BYTE \$6D,\$B6,\$D8,\$00,\$36,\$C0,\$1B,\$6D
3BFD	B6 D8 36 C0 00 1B 6D		.BYTE \$B6,\$D8,\$36,\$C0,\$00,\$1B,\$6D,\$B6
3C04	B6		
3C05	D8 36 C0 1B 6D 00 B6		.BYTE \$D8,\$36,\$C0,\$1B,\$6D,\$00,\$B6,\$D8
3C0D	36 C0 1B FF FF D8 00		.BYTE \$36,\$C0,\$1B,\$FF,\$FF,\$D8,\$00,\$36
3C15	C0 1B FF FF D8 36 C0		.BYTE \$C0,\$1B,\$FF,\$FF,\$D8,\$36,\$C0,\$00
3C1D	FF FF FF FF FF F8 7F		.BYTE \$FF,\$FF,\$FF,\$FF,\$FF,\$F8,\$7F,\$FF
3C24	FF		
3C25	F0 FF FF FF F0 FF		.BYTE \$F0,\$FF,\$FF,\$FF,\$F0,\$FF
3C2B	FD E0 FF F3 07 1C 3C	mouseout	.BYTE \$FD,\$E0,\$FF,\$F3,\$07,\$1C,\$3C,\$04



3C3A	A0		.BYTE \$04,\$02,\$08,\$CF,\$01,\$10,\$F7,\$A0
3C3B	3D 40 01 E0 FF F3 01		.BYTE \$3D,\$40,\$01,\$E0,\$FF,\$F3,\$01,\$20
3C42	20		
3C43	7F 3F 9C FF 80 FF FE		.BYTE \$7F,\$3F,\$9C,\$FF,\$80,\$FF,\$FE,\$FF
3C4A	FF		
3C4B	FF 3C 7E 7C FF FE CF		.BYTE \$FF,\$3C,\$7E,\$7C,\$FF,\$FE,\$CF,\$1F
3C52	1F		
3C53	F0 F3 0F E0 3C 0F E0		.BYTE \$F0,\$F3,\$0F,\$E0,\$3C,\$0F,\$E0,\$1F
3C5B	F0 FF FF FF FF FF FF		.BYTE \$F0,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$CF
3C62	CF		
3C63	FF FE		.BYTE \$FF,\$FE
3C65			
3C65	FF FF FF FF FF FF FF	Question	.BYTE \$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF,\$FF
3C6C	FF		
3C6D	CF 0F E0 F3 03 80 FE		.BYTE \$CF,\$0F,\$E0,\$F3,\$03,\$80,\$FE,\$0C
3C74	0C		
3C75	CF 03 80 F7 E0 BF 03		.BYTE \$CF,\$03,\$80,\$F7,\$E0,\$BF,\$03,\$EF
3C7D	03 FF FE 03 FF FF FF		.BYTE \$03,\$FF,\$FE,\$03,\$FF,\$FF,\$FF,\$FF
3C84	FF		
3C85	FF FF FF		.BYTE \$FF,\$FF,\$FF
3C88	CF 07 80 F3 0E 80 BE	checkmrk	.BYTE \$CF,\$07,\$80,\$F3,\$0E,\$80,\$BE,\$1B
3C8F	1B		
3C90	36 EF 6C 7B D8 01 9E		.BYTE \$36,\$EF,\$6C,\$7B,\$D8,\$01,\$9E,\$B0
3C98	03 60 E7 06 C0 79 0D		.BYTE \$03,\$60,\$E7,\$06,\$C0,\$79,\$0D,\$80
3C9F	80		
3CA0	1B DF 36 F7 6C 3D D8		.BYTE \$1B,\$DF,\$36,\$F7,\$6C,\$3D,\$D8,\$01
3CA7	01		
3CA8	B0 CF 03 60 F3 06 C0		.BYTE \$B0,\$CF,\$03,\$60,\$F3,\$06,\$C0,\$BC
3CAF	BC		
3CB0	0D 80 1B EF 36 FB 6C		.BYTE \$0D,\$80,\$1B,\$EF,\$36,\$FB,\$6C,\$9E
3CB8	D8 01 B0 E7 03 60 38		.BYTE \$D8,\$01,\$B0,\$E7,\$03,\$60,\$38,\$F8
3CBF	F8		
3CC0	06 C0 DC 0D CE 80 36		.BYTE \$06,\$C0,\$DC,\$0D,\$CE,\$80,\$36,\$1B
3CC8	F3 1B 36 3C 0D EC 06		.BYTE \$F3,\$1B,\$36,\$3C,\$0D,\$EC,\$06,\$D8
3CD0	CF 03 30 F3 01 E0		.BYTE \$CF,\$03,\$30,\$F3,\$01,\$E0
3CD6			
3CD6	3C 03 C0 02 E0 CF 01	badmrk	.BYTE \$3C,\$03,\$C0,\$02,\$E0,\$CF,\$01,\$B0
3CDD	B0		
3CDE	F7 D8 7D 6C 36 DF 1B		.BYTE \$F7,\$D8,\$7D,\$6C,\$36,\$DF,\$1B,\$E7
3CE5	E7		
3CE6	0D 80 79 06 C0 03 9E		.BYTE \$0D,\$80,\$79,\$06,\$C0,\$03,\$9E,\$60
3CEE	01 B0 EF D8 FB 6C BE		.BYTE \$01,\$B0,\$EF,\$D8,\$FB,\$6C,\$BE,\$36
3CF6	1B CF 0D 80 F3 06 C0		.BYTE \$1B,\$CF,\$0D,\$80,\$F3,\$06,\$C0,\$3C
3CFE	03 60 01 B0 DF D8 F7		.BYTE \$03,\$60,\$01,\$B0,\$DF,\$D8,\$F7,\$6C
3D05	6C		
3D06	7D 36 1B 9F 0D 80 E7		.BYTE \$7D,\$36,\$1B,\$9F,\$0D,\$80,\$E7,\$06
3D0E	C0 79 03 60 01 BE B0		.BYTE \$C0,\$79,\$03,\$60,\$01,\$BE,\$B0,\$D8



```

3D15| D8
3D16| EF 6C FB 36 3E 1B 0E .BYTE $EF,$6C,$FB,$36,$3E,$1B,$0E,$80
3D1D| 80
3D1E| FF FF FF FF 1F 03 FF diskette .BYTE $FF,$FF,$FF,$FF,$1F,$03,$FF,$FF
3D25| FF
3D26| 86 F8 07 FF FF F4 EF .BYTE $86,$F8,$07,$FF,$FF,$F4,$EF,$0A
3D2D| 0A
3D2E| FB 05 BC 02 80 01 FF .BYTE $FB,$05,$BC,$02,$80,$01,$FF,$7D
3D36| 78 84 CF 01 02 FF 7C .BYTE $78,$84,$CF,$01,$02,$FF,$7C,$01
3D3D| 01
3D3E| 02 84 DF 78 FF FF FF .BYTE $02,$84,$DF,$78,$FF,$FF,$FF,$7F
3D45| 7F
3D46| 07 10 FF FF FF 80 03 .BYTE $07,$10,$FF,$FF,$FF,$80,$03,$FF
3D4D| FF
3D4E| FF FE FF FF FF FF .BYTE $FF,$FE,$FF,$FF,$FF,$FF,$FF
3D54|
3D54| FF 0F 03 FF FF FF 00 lisa .BYTE $FF,$0F,$03,$FF,$FF,$FF,$00,$FF
3D5B| FF
3D5C| C0 07 FF FF FF FF E0 .BYTE $C0,$07,$FF,$FF,$FF,$FF,$E0,$10
3D64| 0C FF FF F0 30 01 FF .BYTE $0C,$FF,$FF,$F0,$30,$01,$FF,$70
3D6B| 70
3D6C| FF F8 07 80 01 9E FF .BYTE $FF,$F8,$07,$80,$01,$9E,$FF,$01
3D73| 01
3D74| FF CF 07 80 FF 3F 07 .BYTE $FF,$CF,$07,$80,$FF,$3F,$07,$80
3D7C| E7 01 FF 09 01 FF 01 .BYTE $E7,$01,$FF,$09,$01,$FF,$01,$FF
3D83| FF
3D8B| FF .BYTE $FF,$F8,$40,$07,$80,$0C,$FF,$FF
3D8C| F0 30 00 07 FF FF FF .BYTE $F0,$30,$00,$07,$FF,$FF,$FF,$FF
3D94| E0 03 FF 00 FF FF FF .BYTE $E0,$03,$FF,$00,$FF,$FF,$FF,$C0
3D9B| C0
3D9C| 03 FF FF FF 00 FF E0 .BYTE $03,$FF,$FF,$FF,$00,$FF,$E0,$07
3DA3| 07
3DA4| FF FF FF FF F0 61 AA .BYTE $FF,$FF,$FF,$FF,$F0,$61,$AA,$AA
3DAB| AA
3DAC| AA 95 FF 18 FF FF 9F .BYTE $AA,$95,$FF,$18,$FF,$FF,$9F,$5F
3DB3| 5F
3DB4| FF FD 86 1F 0A AA A8 .BYTE $FF,$FD,$86,$1F,$0A,$AA,$A8,$15
3DBB| 15
3DBC| 3F 07 FF 00 FF FF FF .BYTE $3F,$07,$FF,$00,$FF,$FF,$FF,$F0
3DC3| F0
3DC4| 03 FF FF FF FC FF E0 .BYTE $03,$FF,$FF,$FF,$FC,$FF,$E0,$FF
3DCB| FF
3DCC| FF FF .BYTE $FF,$FF
3DCE|
3DCE| .ENDC
3DCE|
3DCE| .PAGE

```



```

3DCE|                                     .LIST
3DCE|                                     ;-----
3DCE|                                     ;  Message Table
3DCE|                                     ;-----
3DCE|
3DCE|                                     .IF  USERINT = 0
3DCE|                                     .ENDC                                     ;{USERINT}
3DCE|                                     .IF      BURNIN = 1
3DCE|
3DCE| 50 4F 57 45 52 20 43  BRNMSG  .ASCII  'POWER CYCLING AT '
3DDC| 41 54 20
3DDF| 00                                     .BYTE  0
3DE0| 54 49 4D 45 20 49 53  TIMMSG  .ASCII  'TIME IS '                                     ; RM000
3DE7| 20
3DE8| 00                                     .BYTE  0
3DE9| 44 52 49 56 45 20 54  TWGMSG  .ASCII  'DRIVE TEST'                                     ; RM000
3DF0| 45 53 54
3DF3| 00                                     .BYTE  0
3DF4| 4C 4F 4F 50 20 43 4F  LOOPMSG  .ASCII  'LOOP COUNT IS '
3DFB| 55 4E 54 20 49 53 20
3E02| 00                                     .BYTE  0
3E03| 50 4D 20 42 55 53 20  PMMSG   .ASCII  'PM BUS ERROR'
3E0F| 00                                     .BYTE  0
3E10| 46 4C 4F 50 50 59 20  TWGFAIL .ASCII  'FLOPPY TEST FAILED'
3E1E| 49 4C 45 44
3E22| 00                                     .BYTE  0
3E23| 46 4C 4F 50 50 59 20  TWGRSLT .ASCII  'FLOPPY ERROR COUNT IS '
3E31| 4F 55 4E 54 20 49 53
3E38| 20
3E39| 00                                     .BYTE  0
3E3A|
3E3A|                                     .ENDC
3E3A|
3E3A|                                     .IF  ROM4K = 0
3E3A|                                     .IF  USERINT = 0
3E3A|                                     .ELSE
3E3A|
3E3A| 54 45 53 54 49 4E 47  CHKMSG  .ASCII  'TESTING'
3E41| 00                                     .BYTE  0
3E42| 54 45 53 54                                     .ASCII  'TEST'                                     ;French translation
3E46| 00                                     .BYTE  0
3E47| 45 53 20 57 49 52 44                                     .ASCII  'ES WIRD GETESTET'                                     ;German translation
3E4E| 20 47 45 54 45 53 54
3E55| 45 54
3E57| 00                                     .BYTE  0
3E58| 52 45 53 54 41 52 54  RTRYMSG  .ASCII  'RESTART'
3E5F| 00                                     .BYTE  0

```



```

3E60| 52 45 43 4F 4D 4D 45      .ASCII  'RECOMMENCER'      ;French
3E67| 4E 43 45 52
3E6B| 00                        .BYTE    0
3E6C| 4E 45 55 20 53 54 41      .ASCII  'NEU STARTEN'      ;German
3E73| 52 54 45 4E
3E77| 00                        .BYTE    0
3E78|
3E78| 43 4F 4E 54 49 4E 55  CONTMSG .ASCII  'CONTINUE'
3E7F| 45
3E80| 00                        .BYTE    0
3E81| 43 4F 4E 54 49 4E 55      .ASCII  'CONTINUER'      ;French
3E88| 45 52
3E8A| 00                        .BYTE    0
3E8B| 57 45 49 54 45 52 4D      .ASCII  'WEITERMACHEN'    ;German
3E92| 41 43 48 45 4E
3E97| 00                        .BYTE    0
3E98|
3E98| 53 54 41 52 54 55 50  STRMSG  .ASCII  'STARTUP FROM'
3EA4| 00                        .BYTE    0
3EA5| 44 45 4D 41 52 52 45      .ASCII  'DEMARRER DE'      ;French
3EAC| 52 20 44 45
3EB0| 00                        .BYTE    0
3EB1| 53 54 41 52 54 45 4E      .ASCII  'STARTEN VON'      ;German
3EB8| 20 56 4F 4E
3EBD|                          .BYTE    0
3EBD| 2E 2E 2E                PERIODS .ASCII  '...'
3EC0| 00                        .BYTE    0
3EC1|
3EC1| 4F 50 54 49 4F 4E 53  MENUHDG .ASCII  'OPTIONS'
3EC8| 00                        .BYTE    0
3EC9|
3EC9| 44 49 53 50 4C 41 59  DISPMSG .ASCII  'DISPLAY MEM   1'
3ED0| 20 4D 45 4D 20 20 20
3ED7| 31
3ED8| 00                        .BYTE    0
3ED9| 53 45 54 20 4D 45 4D  SETMSG  .ASCII  'SET MEMORY   2'
3EE7| 32
3EE8| 00                        .BYTE    0
3EE9| 43 41 4C 4C 20 50 52  CALLMSG  .ASCII  'CALL PROGRAM  3'
3EF0| 4F 47 52 41 4D 20 20
3EF7| 33
3EF8| 00                        .BYTE    0
3EF9|                          .IF  ROM16K = 1
3EF9| 4C 4F 4F 50 20 4F 4E  LPMSG    .ASCII  'LOOP ON TEST   4'
3F00| 20 54 45 53 54 20 20
3F07| 34
3F08| 00                        .BYTE    0

```



```

3F09|                                     .ENDC
3F09|
3F09| 41 44 4A 55 53 54 20 VIDMSG .ASCII 'ADJUST VIDEO 5'
3F10| 56 49 44 45 4F 20 20
3F17| 35
3F18| 00                                     .BYTE 0
3F19|
3F19|                                     .IF BURNIN = 1
3F19| 50 4F 57 45 52 20 43 CYCLMSG .ASCII 'POWER CYCLE 6'
3F20| 59 43 4C 45 20 20 20
3F27| 36
3F28| 00                                     .BYTE 0
3F29|                                     .ENDC
3F29|
3F29| 51 55 49 54 20 20 20 QUITMSG .ASCII 'QUIT 7'
3F30| 20 20 20 20 20 20 20
3F37| 37
3F38| 00                                     .BYTE 0
3F39|
3F39| F4 F1 F2 MENUID .BYTE KEY1,KEY2,KEY3 ;menu id table
3F3C|
3F3C|                                     .IF ROM16K = 1
3F3C| F3 .BYTE KEY4
3F3D|                                     .ENDC
3F3D|
3F3D| E4 .BYTE KEY5
3F3E|
3F3E|                                     .IF BURNIN = 1
3F3E| E1 .BYTE KEY6
3F3F|                                     .ENDC
3F3F|
3F3F| E2 .BYTE KEY7
3F40|
3F40|                                     .ENDC
3F40|                                     .IF ROM16K = 1
3F40|                                     .IF FULLSCC = 0
3F40|                                     .ENDC
3F40| 31 20 2D 20 52 4F 4D TSTMENU .ASCII '1 - ROM'
3F47| 0D .BYTE RET
3F48| 32 20 2D 20 4D 4D 55 .ASCII '2 - MMU'
3F4F| 0D .BYTE RET
3F50| 33 20 2D 20 56 49 44 .ASCII '3 - VIDEO' ;RM000
3F57| 45 4F
3F59| 0D .BYTE RET
3F5A| 34 20 2D 20 50 41 52 .ASCII '4 - PARITY' ;RM000
3F61| 49 54 59
3F64| 0D .BYTE RET

```



```

3F65| 35 20 2D 20 50 41 52      .ASCII  '5 - PARA VIA'          ;RM000
3F6C| 41 20 56 49 41
3F71| 0D                        .BYTE    RET
3F72| 36 20 2D 20 4B 59 42      .ASCII  '6 - KYBD VIA'          ;RM000
3F79| 44 20 56 49 41
3F7E| 0D                        .BYTE    RET
3F7F| 37 20 2D 20 43 4F 50      .ASCII  '7 - COPS'
3F86| 53
3F87| 0D                        .BYTE    RET
3F88| 38 20 2D 20 53 43 43      .ASCII  '8 - SCC'
3F8F| 0D                        .BYTE    RET
3F90| 39 20 2D 20 44 49 53      .ASCII  '9 - DISK'
3F97| 4B
3F98| 0D                        .BYTE    RET
3F99| 41 20 2D 20 43 4C 4F      .ASCII  'A - CLOCK'
3FA0| 43 4B
3FA2| 0D                        .BYTE    RET
3FA3| 42 20 2D 20 4D 45 4D      .ASCII  'B - MEMORY'
3FAA| 4F 52 59
3FAD| 0D                        .BYTE    RET
3FAE| 43 20 2D 20 49 4F 20      .ASCII  'C - IO SLOTS'
3FB5| 53 4C 4F 54 53
3FBA| 00                        .BYTE    0
3FBB|                          .ENDC
3FBB|
3FBB|                          .IF  USERINT = 0
3FBB|                          .ELSE
3FBB| 41 44 44 52 45 53 53  ADDRMSG .ASCII  'ADDRESS ?'
3FC2| 20 3F
3FC4| 00                        .BYTE    0
3FC5| 44 41 54 41 20 3F      DATAMSG .ASCII  'DATA ?'
3FCB| 00                        .BYTE    0
3FCC| 43 4F 55 4E 54 20 3F  CNTMSG  .ASCII  'COUNT ?'
3FD3| 00                        .BYTE    0
3FD4| 54 45 53 54 20 3F      TSTMSG  .ASCII  'TEST ?'
3FDA| 00                        .BYTE    0
3FDB|                          .ENDC
3FDB| 57 48 41 54 20 3F      WHATMSG .ASCII  'WHAT ?'
3FE1| 00                        .BYTE    0
3FE2|                          .ENDC          ; {ROM4K}
3FE2|
3FE2|                          .ENDCROM4K = 1
3FE2|
3FE2|                          .IF  ROM8K = 1
3FE2|                          .ENDC

```

```

3FE2|
3FE2|                                     .IF ROM16K = 1
3FE2| 00 00 00 00 00 00 00          .ORG    $3FF4                ;                CHG005
3FF4|
3FF4|                                     ;***** COPYRIGHT NOTICE *****
3FF4| 43 38 34 41 50 50 4C  HDGMSG  .ASCII  'C84APPLE'          ;                CHG005
3FFB| 45
3FFC|                                     ;*****
3FFC|
3FFC| 02                VRSN      .BYTE    $02                ;version 2                CHG001
3FFD| 48                REV      .ASCII  'H'                ; rev H                CHG001
3FFE|                .ENDC
3FFE|
3FFE|
3FFE| 0000                LAST    .WORD    $0000                ;checksum word for ROM test
4000|                .END

```

----- SYMBOLTABLE DUMP

```

AB - Absolute      LB - Label      UD - Undefined      MC - Macro
RF - Ref           DF - Def         PR - Proc           FC - Func
A5 - A5 Global     32 - 32-bit Global

```

```

A6SAV   AB 000001F8| AAPL      AB 00000000| ACHK1   LB 00000286
ACHK2    LB 0000029C| ACR1      AB 00000016| ACTADDR  AB 00000270
ACTDATA  AB 00000274| ACTL      AB 00000002| ADDRMSG  LB 00003FBB
ADR128K  AB 00020000| ADRCHK    LB 00000EF6| ADRCLR   LB 00000F00
ADREXCP  AB 00000006| ADRLTCH   AB 000001AA| ADRMSK   AB 00000003
ADRREGS  AB 000001E0| ADRTST    LB 00000EE6| ADRVCT   LB 0000000C
ADRVCTR  AB 0000000C| AERR       LB 0000074E| AKEY     AB 000000F0
ALBOXCOL AB 00000006| ALBOXROW  AB 00000031| ALPHA    LB 000016A8
ALPHKEY  AB 000000FD| ALRMSAV   AB 00FCC1B1| ALRTCOL  AB 00000010
ALRTHIGH AB 000000A4| ALRTROW   AB 00000073| ALRTSTRT AB 00001140
ALRTWIDT AB 0000004E| ALTEMSK   AB 70000000| ALTBOOT  AB 0000001C
ALTCOL   AB 00000014| ALTKYADD  AB 00000445| APPLNET  AB 00008001
APPLICON LB 000038B6| APPLQUAL  AB 00009FFF| ASCIITAB LB 000038BC
B96DATA  LB 00001038| B96LTH    AB 00000010| BADBRD   LB 000021A2
BADHDR   AB 00000054| BADMRK    LB 00003CD6| BADRSP   AB 00000052
BADSM    AB 0000005C| BADST     AB 0000005D| BADTHDR  AB 00000026
BASE     LB 00000000| BASICTST  LB 00000E6A| BEEP     LB 00000AE8
BEGIN    LB 000000F6| BEGIN2    LB 00000152| BEGIN3   LB 0000018A
BERR     LB 00000742| BFAIL2    LB 00001F3E| BKEY     AB 000000EE
BLACK    LB 000035FC| BLACKEN   LB 00003106| BLKH     AB 00000001
BLKL     AB 00000003| BLKM      AB 00000002| BLKNUM   AB 00000536
BLKSIZE  AB 00000200| BMENU     AB 00000001| BMENULEN AB 00000022
BMENUSPC AB 00000BF4| BMENUWID  AB 00000012| BOOTCHK  LB 000016E6
BOOTDATA AB 000001B4| BOOTDVCE  AB 000001B3| BOOTFAIL LB 00001F3A

```



BOOTMEM	AB	00000188	BOOTMENU	LB	000018F8	BOOTMSK	AB	008FFFFFF
BOOTPAT	AB	0000AAAA	BOTSIDE	AB	00000001	BOUNDS	LB	00002F7C
BRNMSG	LB	00003DCE	BS	AB	00000008	BSR2	MC	-----
BSR4	MC	-----	BSR6	MC	-----	BSRS2	MC	-----
BSRS4	MC	-----	BSRS6	MC	-----	BSY	AB	00000001
BSYTIME	AB	00000500	BTEXTY	AB	00020002	BTERR	LB	00001D2A
BTMENU	AB	0000001D	BTN	AB	00000006	BTN1MSG	AB	00001C08
BTN3MSG	AB	00003DC8	BTN3STRT	AB	00003A36	BTNCOL	AB	00000034
BTNHIGH	AB	0000001C	BTNMSPC	AB	00000392	BTNROW	AB	00000045
BTNSPC	AB	000010E0	BTNWIDTH	AB	0000000A	BURNIN	AB	00000001
BUSEXCP	AB	00000005	BUSVCT	LB	00000008	BUSVCTR	AB	00000008
BUTN	AB	00000003	BUTN1	AB	00000001	BUTN2	AB	00000005
BYTESPER	AB	00000007	CALL3	LB	00000E6C	CALLBASE	AB	00000480
CALLMSG	LB	00003EE9	CALLRTN	LB	00002908	CFGEXIT	LB	00001370
CHARROWS	AB	0000001B	CHECK	LB	000035FE	CHECKMRK	LB	00003C88
CHECKSUM	LB	00000CB2	CHKBASE	LB	0000025E	CHKBSY	LB	00001F82
CHKCMD	AB	00000005	CHKCNT	AB	000000BA	CHKCNT2	AB	000000C4
CHKCPU	LB	000035B6	CHKDRIVE	LB	00001D5C	CHKFDIR	LB	00001E96
CHKFIN	LB	00001E3E	CHKHI	LB	000004E2	CHKICONS	LB	00001B68
CHKID	LB	00001380	CHKINPUT	LB	00002EA2	CHKIOBRD	LB	000035C6
CHKIT	LB	00002C52	CHKIT2	LB	00002CBE	CHKLO	LB	0000045E
CHKMADR	LB	0000153C	CHKMBRD	LB	000035BE	CHKMEM	LB	00000508
CHKMSG	LB	00003E3A	CHKPAS2	LB	00002244	CHKPASS	LB	0000223A
CHKPM	LB	00001836	CHKPOSN	LB	00002E46	CHKPROFI	LB	00001A36
CHKPXIT	LB	00002E9C	CHKROW	AB	0000004B	CHKRW	LB	000002B4
CHKS2	LB	000019EE	CHKS3	LB	00001A00	CHKSLOT	LB	00001B44
CHKSXIT	LB	00001B94	CHKTIM	LB	0000229A	CHKVCT	LB	00000018
CHKXCRD	LB	000035CE	CHRHIGH	AB	00000008	CHRSPC	AB	0000000A
CHRWIDTH	AB	00000001	CKEY	AB	000000ED	CKLOOP	LB	00001890
CKKIT	LB	000018AE	CLAMP	AB	00000009	CLICK	LB	00000AEE
CLK	AB	0000000E	CLKDATA	AB	000001BA	CLKERR	LB	000012E2
CLKSAVE	AB	00FCC1A1	CLKTST	LB	0000128C	CLMPERR	AB	00000016
CLOCKBYT	AB	00000480	CLRDBOX	LB	00002C28	CLRDESK	LB	000030DA
CLRFDIR	LB	00001E72	CLRINT	LB	00001C24	CLRIT	LB	00002B7A
CLRMENU	LB	00003114	CLRPM	LB	00002250	CLRRST	LB	00000AC4
CLRSCRN	LB	000026F0	CLRSTAT	AB	00000085	CMD	AB	00000002
CMDBUFR	AB	00000304	CMDCHK	LB	00001E04	CMDOWN	AB	000000FF
CMDERR	LB	0000245E	CMDFLG	AB	00000003	CMDKEY	AB	000000FF
CMDTIME	AB	00120000	CMDUP	AB	0000007F	CNFRM	AB	0000000E
CNTINC	LB	0000238A	CNTMSG	LB	00003FCC	COARSE	LB	00002F6C
CODECOL	AB	00000012	CODEROW	AB	00000097	COL1STRT	AB	00001B72
COL2MID	AB	000039BC	COL2STRT	AB	00001B7E	COL3STRT	AB	00001B8A
COMPARE	LB	00003150	CONCHK	LB	00000410	CONFIG	LB	000012EE
CONFIG2	LB	000012F6	CONOFF	LB	00000800	CONOK	LB	0000043E
CONSET	LB	000007FA	CONSET2	LB	00000802	CONT	LB	00003132
CONTCHK	LB	0000265E	CONTMSG	LB	00003E78	CONTMSK	AB	001E3FFA
CONXT	AB	00000538	CONVERT	LB	00002BF4	CONVRTD5	LB	000036EE



COPS0	LB	00002D38	COPS1	LB	00002D48	COPS2	LB	00002D5C
COPS4	LB	00002D9E	COPSBAD	LB	0000090A	COPSCHK	LB	000011C0
COPSCMD	LB	00000956	COPSENB	LB	000008EA	COPSVCT	LB	00000916
COPY6	LB	00002092	COPY6LP	LB	000020A0	CPIOMSK	AB	001FFFFFF
CPSINIT	LB	00000920	CPUBRD	LB	0000398B	CPUINTR	AB	00000004
CPUMSK	AB	0000000F	CPUSEL	AB	00000001	CPUSTRT	AB	00001DF6
CRSRBUSY	AB	0000049B	CRSRDATA	LB	0000391C	CRSRHEIG	AB	00000494
CRSRHIDD	AB	0000049E	CRSRHOTX	AB	00000490	CRSRHOTY	AB	00000492
CRSRMASK	LB	0000391C	CRSRBOSC	AB	000004A0	CRSRTRAC	AB	0000049A
CRSRVISI	AB	0000049C	CRSRX	AB	00000496	CRSRY	AB	00000498
CRTCOL	AB	00000302	CRTRW	AB	00000300	CSBIT	AB	00000005
CSTRB	AB	00FCD01C	CURSORDI	LB	0000300E	CURSORHI	LB	00002FEA
CUSORIN	LB	00002FCC	CYCLCNT	AB	00FCC1C1	CYCLMSG	LB	00003F19
CYCLVAL	AB	00FCC1C3	D7SAV	AB	000001AC	DATABFR	AB	00020000
DATAMSG	LB	00003FC5	DATARGS	AB	000001C0	DBOXCOL	AB	00000018
DBOXDSPL	LB	00002C0A	DBOXHIGH	AB	00000014	DBOXLEFT	AB	00000014
DBOXROW	AB	00000018	DBOXSTRT	AB	0000071E	DBOXTOP	AB	00000168
DBOXWIDT	AB	00000042	DDRA1	AB	00000006	DDRA2	AB	00000018
DDRB1	AB	00000004	DDRB2	AB	00000010	DEBUG	AB	00000000
DEFCOL	AB	0000003E	DEFROW	AB	000000A4	DEFSTRT	AB	000039E6
DEFVID	AB	0000002F	DEFVID2	AB	000000AF	DELAY	LB	00000AE2
DELAY5	LB	00000AD4	DELAY_1	LB	00000ACC	DESKLINE	AB	000005FA
DESKLMT	AB	00007FF8	DESKPATR	AB	AAAA5555	DG2OFF	AB	00FCE004
DG2ON	AB	00FCE006	DIAGS	AB	00000001	DIE	AB	00000089
DISABLE	MC	-----	DISINT	LB	00002364	DISK	AB	00000011
DISKCOL	AB	00000004	DISKETTE	LB	00003D1E	DISKMEM	AB	00FCC001
DISKROM	AB	00FCC031	DISKROW	AB	00000012	DISPMSG	LB	00003EC9
DIV0VCT	LB	00000014	DLCNT	AB	FFFFFFFFC	DLOOP	LB	000035EC
DLYCNST	AB	00000009	DLYTIME	AB	00100000	DOBOOT	LB	000016E2
DOCRES	LB	00002146	DODSPLY	LB	000035AE	DOMENU	LB	000025EC
DONE	LB	0000360A	DOREAD	LB	00001C2C	DORESET	LB	00002652
DOSUM	LB	0000019E	DRAWBTN	LB	000032F8	DRAWDESK	LB	000030D6
DRAWSIDE	LB	00003342	DRIVE	AB	00000535	DRIVEN	LB	00003B1E
DRV	AB	00000004	DRV1	AB	00000000	DRV2	AB	00000080
DRVCOL	AB	00000003	DRVERR	AB	00000007	DRVROW	AB	00000006
DRVTYPE	AB	00FCC015	DRWHORZ	LB	00002A26	DRWVERT	LB	00002A40
DSABLDSK	LB	00001D46	DSABLINT	AB	00000087	DSAVARRY	AB	FFFFFFFF8
DSCONT	LB	000023FC	DSCRACH	AB	FFFFFFFF18	DSK1IN	AB	00000000
DSK2IN	AB	00000004	DSKBAD	LB	00001CAA	DSKBSY	AB	00000051
DSKBUFF	AB	000003E8	DSKCHK	LB	00001C9A	DSKCNTH	AB	00FCC19D
DSKCNTL	AB	00FCC19F	DSKDATA	AB	00000400	DSKDIAG	AB	00000006
SKDIS	LB	00001CC0	SKERR	LB	00001CC4	SKERR2	LB	00001D0E
SKERR3	LB	00001D14	SKIN	AB	00000002	SKOUT	LB	00001CBC
SKRSLT	AB	000002AE	SKSIZE	AB	000006A6	SKTIMER	LB	00001C98
SKTMOUT	AB	001C8000	SKTST	LB	0000110C	SKVCT	LB	00001186
SKXIT	LB	00001180	DSPALL	LB	00003686	DSPALRTI	LB	0000352C
DSPALRTM	LB	000036DA	DSPBAD	LB	000034EC	DSPCH	LB	000016B0



DSPCHECK	LB	000035D4	DSPCLK	LB	000024CA	DSPCODE	LB	00001622
DSPCPU	LB	00003588	DSPCPURM	LB	000008DC	DSPCXIT	LB	00001668
DSPDEC	LB	00001630	DSPDONE	LB	00003716	DSPERR	LB	0000244E
DSPERRIC	LB	000034DA	DSPFRNCH	LB	000036AA	DSPGERMN	LB	000036A6
DSPICON	LB	000035E2	DSPIOB	LB	0000359C	DSPIT	LB	000036B8
DSPMBRD	LB	00003592	DSPMEM	LB	00002836	DSPMENU	LB	00002744
DSPMENUB	LB	000027D0	DSPMNTY	LB	00001A94	DSPMSG	LB	00003700
DSPMSGR	LB	000036F6	DSPMSLSH	LB	000036D2	DSPNUM	LB	00003464
DSPNUMIC	LB	00003450	DSPOUT	LB	000036B0	DSPQICON	LB	00003556
DSPRGICO	LB	00003424	DSPSTRIN	LB	00003634	DSPTIM	LB	00002398
DSPVAL	LB	0000373A	DSPVXIT	LB	000037B4	DSPWTICO	LB	00001EA6
DSPXCRD	LB	000035A6	DSTACK	AB	FFFFFF18	DVCCODE	AB	00FCC189
DVCECHK	LB	00001754	EADREXCP	AB	0000002E	EBOOT	AB	0000004B
EBUSEXCP	AB	0000002D	ECLK	AB	00000036	ECPAR	AB	0000002B
ECPUINTR	AB	0000002C	ECPUSEL	AB	00000029	EDISK	AB	00000039
EILLEXCP	AB	00000030	EIOCOP	AB	00000034	EIOCOP2	AB	0000003B
EIOEXCP	AB	0000003A	EIOKBD	AB	0000003C	EJCTDSK	LB	00001E56
EJCTTIME	AB	00180000	EKBDCOP	AB	00000035	EMEM	AB	00000046
EMISEXCP	AB	0000002F	EMMU	AB	00000028	ENABLE	MC	-----
ENBLDRVS	LB	00002E2A	ENBLINT	AB	00000086	ENDPM	AB	00FCC1FF
ENQKBD	LB	00002B86	ENTRKEY	AB	000000AF	EPAR	AB	00000047
ERRCOL	AB	00000010	ERRDISP	LB	000014EA	ERRMSK	AB	0E7FFFFF
ERRROW	AB	00000073	ERRSTRT	AB	0000287E	ERS232A	AB	00000037
ERS232B	AB	00000038	ETREXCP	AB	00000031	EVIA1	AB	00000032
EVIA2	AB	00000033	EVID	AB	0000002A	EXCADR	AB	00000282
EXCFC	AB	00000280	EXCHK	LB	0000140E	EXCIR	AB	00000286
EXCLUSIV	LB	0000314E	EXCP0	LB	00000758	EXCP1	LB	00000764
EXCPC	AB	0000028A	EXCPERR	LB	00000030	EXCSR	AB	00000288
EXCTYPE	AB	0000028E	EXIT	LB	00000CF8	EXMEM	AB	00000006
EXMSK	AB	000003F0	EXPAND	LB	000018B0	EXRW	AB	00000081
EXTERNAL	AB	00000000	FASTMR	AB	00000006	FDIR	AB	00000004
FDIRTIME	AB	00C00000	FILEID	AB	00000004	FINDD2	LB	000020BC
FINDERR	LB	000020E6	FINDSYNC	LB	00000D0A	FINE	LB	00002F56
FINKBD	AB	00000001	FINLISA	AB	00000000	FIRSTCOL	AB	00000018
FIRSTROW	AB	0000003E	FIVESEC	AB	001312D0	FMT	AB	00000003
FMTIME	AB	01800000	FNDXIT	LB	000020F6	FONTTBL	LB	000037C0
FULLSCC	AB	00000001	GET0	LB	000009F0	GET1	LB	00002C4E
GET2	LB	00002CBA	GET3	LB	00002D0C	GETA	LB	00002BA4
GETBITS1	LB	00000C1C	GETBITS2	LB	00000C50	GETBYTES	LB	00000C82
GETCH	LB	00002B96	GETDATA	LB	00000A7E	GETDIG	LB	00001634
GETERR	LB	000026E6	GETEXIT	LB	00002BE8	GETINPUT	LB	00002C46
GETIT	LB	00000AA2	GETJMP	LB	00000A38	GETL1	LB	0000262A
GETL1XIT	LB	000026DA	GETLENGT	LB	00003418	GETLEV2	LB	0000274C
GETNIBBL	LB	00000D34	GETNTRY	LB	00002E5A	GETPADDR	LB	00000FF0
GETPARM	LB	00002BB2	GETROWCO	LB	00003406	GETRSP	LB	000020CE
GETSTAT	LB	00002060	GETXIT2	LB	00002BF2	GLOBALS	AB	00000480
GOTOMON	LB	000015CC	GRAY	LB	000030EE	GRAY1	LB	000030F4



HALFMEG	AB	00080000	HALFSEC	AB	0001E848	HALFSIZE	AB	00000070
HDERR2	LB	00001F5E	HDERR3	LB	00001F66	HDGMSG	LB	00003FF4
HDRBUFR	AB	0001FFEC	HDRLEN	AB	0000000C	HDRSIZE	AB	00000014
HDSKERR	LB	00001F10	HEX128K	AB	00020000	HEX2K	AB	00000800
HEX32K	AB	00008000	HEX512K	AB	00080000	HEX8K	AB	00002000
HEX96K	AB	00018000	HIP7CH	LB	000016C8	HOURL	AB	000001BC
HOURSAY	AB	00FCC193	ICBIT	AB	0000000D	ICERR	AB	000000FE
ICONADDR	AB	00000532	ICONCHK	LB	00001986	ICONCNT	AB	00000534
ICONCSPC	AB	00001680	ICONHIGH	AB	00000020	ICONMENU	LB	00001AAA
ICONMSPC	AB	00000440	ICONPTR	AB	00020004	ICONRSPC	AB	0000000C
ICONWIDT	AB	00000006	IER1	AB	0000001C	IERR	LB	000006F8
IFR1	AB	0000001A	ILLEXCP	AB	00000008	ILLVCT	LB	00000010
ILLVCTR	AB	00000010	INCSR	AB	00000004	INDATA	AB	00000000
INIT1	LB	00002544	INIT2	LB	0000255C	INIT3	LB	00002570
INITB2	LB	000010CE	INITB2L	AB	00000002	INITBDAT	LB	000010CA
INITBLTH	AB	00000004	INITFLG	AB	00FCC191	INITMEM	LB	0000066C
INITMON	LB	00002534	INITVCT	LB	000006A6	INSERTD	LB	00003B56
INSRTCOL	AB	00000004	INSRTROW	AB	00000005	INSRTTIM	AB	00C00000
INTERR	LB	00001174	INTLV	AB	00000012	INTSTAT	AB	0000005E
INV	AB	0000005B	INVALID	LB	00002A74	INVCHAR	LB	000038B0
INVERSE	LB	0000312E	INVERT	LB	00002EC2	INVERTCK	AB	00000000
INVICON	LB	00003574	INVID	LB	0000219E	INVIDBIT	AB	00000006
INVPAG	AB	00000C00	INVPARM	LB	00002BEE	INVSUM	LB	0000222A
INTVST	LB	000008A2	INVKIT	LB	00002A86	IO1ERR	AB	00000019
IO1ID	AB	00000298	IO1PORT1	AB	00000003	IO1PORT2	AB	00000004
IO1STAT	AB	0000029E	IO2ERR	AB	0000001A	IO2ID	AB	0000029A
IO2PORT1	AB	00000006	IO2PORT2	AB	00000007	IO2STAT	AB	0000029F
IO3ERR	AB	0000001B	IO3ID	AB	0000029C	IO3PORT1	AB	00000009
IO3PORT2	AB	0000000A	IO3STAT	AB	000002A0	IOBRD	LB	0000394B
IOCERR	LB	00000A66	IOCHK	LB	0000145E	IOCOPS	AB	0000000C
IOCOPS2	AB	00000013	IOEXCP	AB	00000012	IOKBD	AB	00000014
IOIMT	AB	00000900	IOIMT2	AB	00000901	IOMSK	AB	001FDC00
IOROM	AB	000002A1	IOS1	AB	00000034	IOS2	AB	00000037
IOS3	AB	00000041	IOSBOOT	LB	0000215E	IOSCHK	LB	00001590
IOSMSK	AB	0E000000	IOSPACE	AB	00FC0000	IOSTRT	AB	00001E12
IOTST	LB	00001000	IOVCT	LB	00000918	IRA2	AB	00000008
IRB2	AB	00000000	JMPTBL	LB	00000080	KBDBFR	AB	000002C0
KBDCHK	LB	000014F8	KBDCOPS	AB	0000000D	KBDDELAY	LB	00000ADC
KBDDLY	AB	00067C28	KBDEND	AB	00000300	KBDOUT	AB	00000017
KBDQ	AB	000002B0	KBDQPTR	AB	00000260	KCERR	AB	000000FF
KEY1	AB	000000F4	KEY2	AB	000000F1	KEY3	AB	000000F2
KEY4	AB	000000F3	KEY5	AB	000000E4	KEY6	AB	000000E1
KEY7	AB	000000E2	KEY8	AB	000000E3	KEY9	AB	000000D0
KEYBDOUT	LB	00003BA5	KEYID	AB	000001B2	KEYSCAN	LB	000011EA
KEYTBL	LB	0000127A	KEYTOASC	LB	0000271A	KUNPLG	AB	000000FD
L10VCT	LB	00000028	L10VCTR	AB	00000028	L11VCT	LB	0000002C
L11VCTR	AB	0000002C	LAST	LB	00003FFE	LASTBLK	AB	000006A6



LASTCOL	AB	00000058	LASTROW	AB	0000014C	LCNTHI	AB	00FCC195
LCNTLO	AB	00FCC197	LEV1LOOP	LB	000026DE	LEV2LOOP	LB	00002A82
LEVEL1	LB	000025A4	LEVEL2	LB	0000273C	LISA	LB	00003D54
LISAROM	PR	-----	LOADLMT	LB	00000316	LOADORG	LB	000002E4
LOADPGM	LB	000021EA	LOMEM	AB	00000800	LOOP	AB	0000001F
LOOP0	LB	00003432	LOOP1	LB	00003438	LOOP2	LB	0000343A
LOOPMSG	LB	00003DF4	LOPTBL	LB	000029DC	LOPTST	LB	00002934
LOPTCH	LB	000016CC	LOTONE	LB	00000554	LPMSG	LB	00003EF9
LPTEST	LB	00001054	LSTCHK	LB	0000182E	LVL1VCT	LB	00000064
LVL2VCT	LB	00000068	LVL3VCT	LB	0000006C	LVL4VCT	LB	00000070
LVL5VCT	LB	00000074	LVL6VCT	LB	00000078	LVL7VCT	LB	0000007C
LWRRIGHT	AB	0000052C	MADRERR	LB	000002B0	MAKEALER	LB	00003168
MAKEBOX	LB	000031E6	MAKEBTN	LB	0000327A	MAKEDBOX	LB	000031B2
MAKEMENU	LB	0000336A	MAKEPCAL	LB	00003164	MAKESVCW	LB	00002814
MAKETEST	LB	00003180	MAKEWIND	LB	000031C6	MAPINV	LB	000005A4
MAXADR	AB	00200000	MAXMEM	AB	00000294	MAXTEST	AB	0000000C
MAXX	AB	000002D0	MAXY	AB	0000016C	MBARLEN	AB	00000010
MEALTCH	AB	00FCF000	MEM	AB	00000015	MEMBRD	LB	000039CE
MEMCHK	LB	00001514	MEMCODE	AB	00FCC18D	MEMCOL	AB	00000004
MEMERR	LB	00001582	MEMLMT	AB	00000700	MEMLOOP	LB	00000E0E
MEMMSK	AB	00600000	MEMROW	AB	00000010	MEMSLT	AB	00000186
MEMSIZ	LB	00000446	MEMSLOT	AB	000002AD	MEMSTRT	AB	00001E04
MEMTST1	LB	00000620	MEMTST2	LB	00000E02	MEMTST3	LB	000029CA
MENU	AB	00000007	MENU1MSG	AB	00000658	MENUBASE	AB	00000530
MENUEND	AB	000020B4	MENUHDG	LB	00003EC1	MENUID	LB	00003F39
MENULEN	AB	0000000B	MENULINE	AB	000005A0	MENULOC	AB	00000111
MENUSPC	AB	000003DE	MENUSTRT	AB	000005A2	MENUWIDT	AB	00000012
MERRCHK	LB	00001572	MIDALCOL	AB	0000002D	MIDALROW	AB	00000083
MIDTSTRO	AB	0000005B	MINCNT	AB	00FCC1C5	MINMEM	AB	000002A4
MINSAV	AB	00FCC19B	MINUTE	AB	000001BD	MISC	LB	000006EC
MISEXCP	AB	00000007	MITEMS	AB	00000007	MLOOP	LB	000035F2
MMU	AB	00000000	MMU0B	AB	00008008	MMU0L	AB	00008000
MMU126B	AB	00FC8008	MMU126L	AB	00FC8000	MMU127B	AB	00FE8008
MMU127L	AB	00FE8000	MMUACHK	LB	00000270	MMUEADRB	AB	00FE8008
MMUEADRL	AB	00FE8000	MMUERR	LB	000001D8	MMUERR2	LB	000003F4
MMUERR3	LB	000003FA	MMUINIT	LB	00000216	MMULP	LB	000001E0
MMULPCHK	LB	00000408	MMURSLT	AB	000001B0	MMURW	LB	0000022A
MMUSADRB	AB	00008008	MMUSADRL	AB	00008000	MMUSET	LB	00000290
MMUTST	LB	000001B0	MMUTST2	LB	00000348	MMUTSTE1	LB	000029C0
MON	AB	00000010	MONITOR	LB	0000259C	MOUSDWN	AB	00000086
MOUSDY	AB	0000048A	MOUSDY	AB	0000048B	MOUSE	AB	00000004
MOUSEMOV	LB	00002F2A	MOUSEON	AB	00000007	MOUSEOUT	LB	00003C2B
MOUSINIT	LB	00002FB2	MOUSOUT	AB	00000018	MOUSSCAL	AB	0000048C
MOUSTHRE	AB	0000048E	MOUSUP	AB	00000006	MOUSX	AB	00000486
MOUSY	AB	00000488	MOVINST	LB	0000314A	MPAR	AB	00000016
MRGICON	LB	000034FA	MSBTN	AB	00000002	MSCHK	LB	00000A52
MSGCOL	AB	00000018	MSGLEN	AB	0000052E	MSGROW	AB	0000007E



MSPLG	AB	00000087	MSRCHSZ	AB	00000040	MSUNPLG	AB	00000007
NEWLISA	AB	00000001	NEWTWIG	AB	00000001	NEXTLINE	LB	00003156
NIOLMT	AB	000006FF	NMEMLMT	AB	000008FF	NMI	LB	00000704
NMIEXCP	LB	000000CA	NMIVCT	AB	0000007C	NOC	AB	0000005A
NOCHG	LB	000023BE	NOCNT	AB	00000001	NOCRD	LB	00002194
NOCRD1	LB	00001332	NOCRD2	LB	00001350	NOCRD3	LB	0000136E
NODISK	AB	00000007	NODSK	AB	00000050	NOIO	LB	00000B5A
NOIO2	LB	00000B76	NOIO3	LB	00000B8E	NORESET	AB	00000001
NORSTR	AB	00000000	NOTIFY	LB	000016B8	NOTPE	LB	00000730
NROWS	AB	0000001B	NSPLMT	AB	000000FF	OCD	AB	00000000
OK	AB	000000FF	OKCH	LB	00002BDC	ONE	LB	0000393C
ONEHOUR	AB	00E0F000	ONEMEG	AB	00100000	ONEMIN	AB	0003C000
ONESEC	AB	0003D090	ORA1	AB	00000002	ORA2	AB	00000008
ORB1	AB	00000000	ORB2	AB	00000000	OTHER	LB	000015E6
OTHRBTNS	LB	000025EC	OTHRMSK	AB	01800000	OUT	LB	000037A0
OUTCH	LB	00001678	OUTCHR	LB	0000166E	OUTCSR	AB	00000006
OUTDATA	AB	00000002	OUTNIB	LB	00001696	OUTPUT	LB	0000342C
PAG128K	AB	00000100	PAINTB1	LB	00003128	PAINTB2	LB	00003136
PAINTBIT	LB	0000326A	PAINTV1	LB	0000325A	PAINT_BO	LB	00003128
PAINT_V	LB	0000325A	PAR	AB	00000003	PARERR	LB	00000DD2
PAROFF	AB	00FCE01C	PARON	AB	00FCE01E	PARTST	LB	00000D5C
PARXIT	LB	00000DE8	PATRN	AB	AA55A55A	PATRN2	AB	0000A55A
PBIT	AB	00000001	PBOOT	LB	00001EFC	PC	AB	0000000F
PCCOL	AB	0000000C	PCERR	LB	00000FD0	PCHIGH	AB	000000C0
PCHIP	AB	0000027D	PCHPROW	AB	0000027C	PCMD	AB	00000000
PCMSZ	AB	00000005	PCR1	AB	00000018	PCR2	AB	00000060
PCROW	AB	00000059	PCSTR	AB	0000070A	PCWIDTH	AB	00000056
PEADDR	AB	000001A6	PEADR2	AB	00000278	PERIODS	LB	00003EBD
PHYTOLOG	AB	00080000	PIABASE	AB	00FCA001	PKEY	AB	000000C4
PMCHKSM	AB	00FCC1FD	PMERR	LB	00001820	PMEXIT	LB	00001818
PMMSG	LB	00003E03	PMSTR	AB	00FCC181	PMVCT	LB	000015DC
PMWRDS	AB	00000020	PORTA1	AB	0000001E	PORTA2	AB	00000078
POWERCYC	LB	00002A6C	POWEROFF	LB	00002DD8	PRIVCT	LB	00000020
PRIXIT	LB	00000FC6	PRO	AB	00000033	PROBOOT	LB	00001ECE
PROERR	LB	00001FDE	PROFILE	AB	00000002	PROFLE	AB	00000001
PROICON	LB	00003A9F	PROINIT	LB	00001FF0	PROMPT	LB	00002A92
PROREAD	LB	00001F70	PROXIT	LB	00001FE6	PROXIT2	LB	00001FEA
PRTYINT1	LB	00000F14	PRTYINT2	LB	00000F72	PUTBS	LB	00002B6C
PUTLF	LB	00002B50	QEND	AB	000002C0	QTRMEG	AB	00040000
QTRSEC	AB	0000F424	QUESTCH	LB	000038AA	QUESTION	LB	00003C65
QUESTN	AB	0000003F	QUITMSG	LB	00003F29	R0	AB	00000000
R1	AB	0000005A	R2	AB	000000B4	R3	AB	0000010E
R4	AB	00000168	R5	AB	000001C2	R6	AB	0000021C
R7	AB	00000276	RAMCHK2	LB	00000ECC	RAMNXT	LB	00000EDE
RAMRW	LB	00000EC0	RAMTEST	LB	00000EB0	RBYTES	AB	0000005A
RCNT	AB	0000000A	RDCLK0	LB	000012B2	RDCLK1	LB	000012D4
RDCNT	LB	0000285E	RDDATA	LB	00001FD2	RDDTA	LB	000028C6



RDENTRY	LB	00002EAC	RDERR	LB	00000F0C	RDINPUT	LB	00002ABA
RDIOSLT	LB	000021CC	RDIOXIT	LB	00002234	RDRETRY	LB	00001C48
RDSCTR1	LB	00001C6C	RDSERN	LB	00000BF6	RDSLOTS	LB	00001306
RDSLIT	LB	00001B9A	RDTIME	AB	00180000	RDWRERR	AB	00000017
READCLK	LB	000012A0	READCOPS	LB	00002DBE	READIN	LB	00002AC4
READIT	LB	00002032	READKEY	LB	00002702	READMMU	LB	000005D0
READQ	LB	00002BB6	READS	AB	00000000	RECTCNT	AB	0000053A
RECTTABL	AB	0000053A	REGTST	LB	00000202	REMAP	LB	00000590
RET	AB	0000000D	RETRY	AB	00000004	RETRYCNT	AB	00000020
REV	LB	00003FFD	RLONGS	AB	000000E1	ROM16K	AB	00000001
ROM4K	AB	00000000	ROM8K	AB	00000000	ROMBASE	AB	00FE0000
ROMIDCOL	AB	00000050	ROMIDROW	AB	00000003	ROMSLCT	AB	000000FE
ROMTST	LB	00000194	ROMV	AB	00000030	ROW2ADR	AB	00020000
ROWBYTES	AB	0000005A	ROWLEN	AB	00000042	ROWLINES	AB	0000000A
ROWSLEFT	AB	0000012C	RS232A	AB	0000000F	RS232B	AB	00000010
RSPOK	LB	000020DC	RSPTIME	AB	0000FFFF	RST0	LB	000009DC
RST1	LB	00000A06	RST2	LB	000009F6	RSTCODE	AB	00000080
RSTKBD	LB	00000AAA	RSTLMT	AB	00000FFE	RSTMMU	LB	00000566
RSTRTIME	AB	00900000	RSTSCAN	LB	000009C2	RSTSCC	LB	000010D0
RSTXIT	LB	00000A3C	RTRYCNT	AB	00000058	RTRYMSG	LB	00003E58
RTS2	MC	-----	RTS4	MC	-----	RTS6	MC	-----
RUNTESTS	LB	00000E6A	RWCHK1	LB	0000023C	RWCHK2	LB	0000024C
RWCHK3	LB	00000254	RWERR	LB	000002C2	RWF1	AB	00000002
RWF2	AB	00000006	RXBF	AB	00000000	SAV2PM	LB	0000184E
SAVEDADD	AB	00000528	SAVEDDAT	AB	000004A2	SAVEDROW	AB	00000526
SAVEDX	AB	00000522	SAVEDY	AB	00000524	SAVEHI	LB	000004FA
SAVELO	LB	000004BC	SAVEREG2	LB	0000003E	SAVEREGS	LB	0000003A
SAVERR	LB	0000222C	SAVEXCP	LB	00001D1C	SAVRSLT	LB	00000E98
SCALE	LB	00002F42	SCANCPS	LB	000011E2	SCANERR	LB	00000A60
SCANMSK	AB	00183000	SCANXIT	LB	00000A6A	SCCBCTL	AB	00FCD241
SCCDATA	AB	00000004	SCCEXIT	LB	0000108E	SCCIN	LB	0000107A
SCCLERR	LB	0000108C	SCCLOOP	LB	00001058	SCCLOOP2	LB	0000106C
SCCLXIT	LB	0000108A	SCCOUT	LB	00001066	SCCRSLT	AB	000002AC
SCCSET	LB	00000774	SCCTEST	LB	00001008	SCCVCT	LB	000010EE
SCNRSPTS	LB	0000154C	SCNSPTS	LB	000019D6	SCRACHSI	AB	000000E0
SCRNBASE	AB	00000110	SCRNERR	LB	0000086A	SCRNOK	LB	0000086C
SCRNSAV	LB	00000874	SCRNTST	LB	00000822	SCROLL	LB	00002B08
SCTR	AB	00000008	SEARCH	LB	000018D6	SECLN	AB	00000200
SEEK	AB	00000083	SEG1OFF	AB	00FCE008	SEG1ON	AB	00FCE00A
SEG2OFF	AB	00FCE00C	SEG2ON	AB	00FCE00E	SELF	LB	00002444
SENDMSG	LB	000021A6	SENDSP	LB	0000210E	SERNUM	AB	00000240
SERR1	AB	0000003D	SERR2	AB	0000003E	SET1	AB	00000000
SET2	AB	10000000	SETBUSVC	LB	000006E2	SETCRSR	LB	0000371A
SETCRSR2	LB	0000371E	SETDUR	LB	000016CE	SETERR1	LB	00002448
SETERR2	LB	0000244C	SETMEM	LB	000028A2	SETMMU	LB	000002C6
SETMSG	LB	00003ED9	SETSCC	LB	00001048	SETTYPE	LB	00001198
SETUP	AB	00FCE012	SETUPON	AB	00FCE010	SETVCTRS	LB	000006AC



SETVLTCH	LB	00000886	SETXIT	LB	0000247C	SFER	AB	00000000
SHFTKEY	AB	000000FE	SHR1	AB	00000014	SHUTDOWN	LB	000023E8
SIDE	AB	00000006	SILENCE	LB	00000B52	SIZRSLT	AB	00000184
SIZXIT	LB	00000502	SKEY	AB	000000F6	SLEEP	AB	00000088
SLOT1L	AB	00FC0001	SLOT2	LB	00001334	SLOT2L	AB	00FC4001
SLOT3	LB	00001352	SLOT3L	AB	00FC8001	SLOTCOL	AB	00000003
SLOTMR	AB	00000005	SLOTROW	AB	00000016	SNDRI	LB	000020DE
SNUM	AB	00FE8000	SPACE	LB	000037BA	SPEED	AB	0000000C
SPIN	LB	000000C8	SPLMT	AB	00000F00	SPURVCT	LB	00000060
SQUAWK	LB	0000270C	START	LB	00000440	STARTOP	LB	00003148
STAT	AB	00000010	STAT01	LB	00002076	STAT1	AB	000001B4
STAT2	AB	000001B5	STAT3	AB	000001B6	STAT4	AB	000001B6
STATBFR	AB	000001B4	STATERR	LB	000020B2	STATFLGS	AB	000002A2
STATMSK	AB	C140C000	STATNZ	AB	00000053	STATOK	LB	00002186
STATREG	AB	00FCF801	STATSAV	AB	00FCC161	STATSTRT	AB	00FCC161
STATSUM	AB	00FCC17D	STATUS	AB	00000180	STATWRDS	AB	00000008
STATXIT	LB	000020B6	STBIT	AB	0000000E	STENTRY	AB	00020000
STKBASE	AB	00000480	STRBOOT	LB	00001C82	STRTIME	AB	01200000
STRMSG	LB	00003E98	STRTRD	LB	00002048	STRTXIT	LB	00002074
STST	AB	00000016	SUPSTK	AB	00000290	SVCHIGH	AB	00000140
SVCLEFT	AB	00000014	SVCMSG	LB	00000052	SVCSTRT	AB	00000EDA
SVCTOP	AB	000007BC	SVCWIDTH	AB	00000042	SYSOK	LB	000016DE
SYSTYPE	AB	000002AF	T1LH1	AB	0000000E	T1LH2	AB	00000038
T1LL1	AB	0000000C	T1LL2	AB	00000030	T2CH1	AB	00000012
T2CH2	AB	00000048	T2CL1	AB	00000010	T2CL2	AB	00000040
TAG	LB	00000D58	TBLEND	LB	0000128B	TBOOTERR	LB	00001D0A
TCNT	AB	00000003	TENSECS	AB	00009000	TERR	LB	000024C4
THREE	LB	00003946	THRESH	AB	00000005	TIMFLG	AB	00FCC199
TIMMSG	LB	00003DE0	TIMOUT	AB	00000027	TKILLER	AB	000000AC
TMOUT	AB	00000055	TNTHSEC	AB	000061A8	TODSET	LB	00002466
TOPE	LB	00000AF6	TONE2	LB	00000B06	TONEDLY	LB	00000654
TOOLONG	LB	000024C2	TOPOFFSE	AB	0000010E	TOPSIDE	AB	00000000
TOTLMEM	AB	000002A8	TRAK	AB	0000000A	TRAPVCT	LB	0000001C
TRCVCT	LB	00000024	TRK1	AB	00000001	TRPERR	LB	00000736
TRPEXCP	AB	00000009	TRPVCT0	AB	00000080	TRYRD	LB	00001F90
TST2	LB	000013AA	TSTBIT	AB	0000000C	TSTCHK	LB	0000139A
TSTCOL	AB	0000000A	TSTCRD	AB	00001000	TSTDONE	LB	00000E4E
TSTERR	LB	00002344	TSTHI	LB	000004CE	TSTICOL	AB	00000014
TSTINIT	LB	00000E84	TSTIROW	AB	00000055	TSTISPC	AB	0000000E
TSTLOOP	LB	00000200	TSTMCOL	AB	0000000E	TSTMENU	LB	00003F40
TSTMROW	AB	00000040	TSTMSG	LB	00003FD4	TSTQUAL	AB	00001800
TSTROW	AB	00000031	TSTSTAT	LB	00000F68	TSTWHIGH	AB	00000054
TSTWSTRT	AB	00001144	TSTWIDT	AB	00000046	TSTXIT	LB	000015C0
TSTXIT2	LB	000015C4	TURNON	LB	0000093E	TWG1	AB	00000031
TWG2	AB	00000032	TWGBOOT	LB	00001BCC	TWGCHK	LB	000022EA
TWGDATA	AB	00020000	TWGDISP	LB	00002510	TWGERR	LB	00001DF2
TWGFAIL	LB	00003E10	TWGHDR	AB	0001FFF4	TWGLOOP	LB	0000249A




```

TWGMSG  LB 00003DE9| TWGOK   LB 00001DFA| TWGOUT  LB 00001DF0
TWGRD   LB 00001D70| TWGREAD LB 00001D76| TWGRSLT LB 00003E23
TWGRXIT LB 00001DFE| TWGTST  LB 0000247E| TWIG1   AB 00000000
TWIG2   AB 00000001| TWIGGY  AB 00000001| TWO     LB 00003941
TWOSSEC AB 0007A120| TXBE    AB 00000002| TYPE    AB 00000014
UCLMPERR AB 00000019| UNCLAMP AB 00000002| UPPER   LB 00003ADA
USERINT  AB 00000001| USPSAV  AB 000001FC| VCTRINIT LB 00001EB0
VECTLOOP LB 0000101A| VFY     AB 00000004| VFYCHKSM LB 0000188C
VFYTIME  AB 01800000| VIA1    AB 0000000A| VIA1BASE AB 00FCDD81
VIA1CHK  LB 000008B0| VIA1TST LB 000008A2| VIA1VCT  LB 00000DF8
VIA2     AB 0000000B| VIA2BASE AB 00FCD901| VIA2CHK  LB 00000780
VIA2TST  LB 00000780| VIA2VCT  LB 000007AE| VIAFAIL  LB 000007F4
VIARW    LB 000007D0| VIARWEND LB 000007F6| VIATST   LB 000007B8
VID       AB 00000002| VIDAJSST LB 000029F4| VIDBIT   AB 00000004
VIDCHK   LB 00000BA2| VIDERR   LB 00000BD4| VIDLTCH  AB 00FCE800
VIDMSG   LB 00003F09| VIDTST   LB 00000B96| VIDXIT   LB 00000BE0
VMSK     AB FFFF8000| VRBIT    AB 00000002| VRSN     LB 00003FFC
VSRCHSZ  AB 00008000| VTIRDIS  AB 00FCE018| VTIRENB  AB 00FCE01A
W14COL   AB 00000017| W34COL   AB 00000041| WAIT2    LB 00002CBA
WAIT3    LB 00002D0E| WAITALRT LB 00001EA6| WAITICON LB 00003A3D
WCOL     AB 00000002| WFBSY    LB 000020FC| WFBSY1   LB 00002100
WFNBSY   LB 00002122| WFNBSY1  LB 00002138| WFNBSY2  LB 0000212A
WFNBSY3  LB 00002132| WHATMSG  LB 00003FDB| WHITEN   LB 00003108
WINDHIGH AB 00000140| WINDSTRT AB 0000070A| WINDWIDT AB 00000056
WMIDCOL  AB 0000002D| WMIDROW  AB 000000B4| WORDSPER AB 0000000E
WRAPXIT  LB 00000500| WRITESCC LB 000010BE| WRITETIT LB 000033B8
WRMSTRT  AB 0000001E| WROW     AB 00000014| WRPERR   AB 00000014
WRT       AB 00000001| WRMENU   LB 000027A0| WRTMMU   LB 000005BA
WRTMSG   LB 00002376| WRTSCRN  LB 000026F2| WRTSUM   LB 0000187E
WT4BOOT  LB 00001A14| WT4INPUT LB 00002D38| WWPERR   LB 00000DF4
XCARD    LB 00003A14| XCRDSTRT AB 00001E20| XFRDATA  LB 00001DCE
XFRHDR   LB 00001DB4| XLATE    LB 0000125E| XLOOP    LB 0000361A
XPCTADDR AB 00000268| XPCTDATA AB 0000026C|

```

```

Assembly complete: 11840 lines
                   0 Warnings
                   0 Errors

```

----- ASSEMBLY COMPLETE.

```

;=====
;APPLE LISA COMPUTER 16K ROM DUMP
;=====
;
;CREATED BY DAVID T CRAIG
;71533.606@COMPUSERVE.COM
;09 JUNE 1998

```

```

;ROM NOTES:
;
; O STARTING ADDRESS IS $FE0000
;
; O ROM IS 16K BYTES IN SIZE
;

```

```

; O ROM MAPPED TO ADDRESS $0
; WHEN LISA STARTS SO THAT
; KEY VALUES AT THE START OF
; THE ROM OCCUPY THE 68000
; CPU'S LO-MEM VECTOR TABLE
;
; O THIS ROM HAS VERSION 2.48

```



```
; WHICH IS ALSO SEEN AS 2.H
;
; (SEE THE 2ND 2-BYTE WORD
; FROM THE ROM'S END, 0248)
;
; O LAST 2-BYTE WORD IS THE
; ROM CHECKSUM WHICH IN THIS
; DUMP IS INCORRECT AT $0000
;
; (USE A REAL LISA AND ITS
; BUILT-IN "SERVICE MODE" TO
; PEEK AT THE ROM AND FIND
; WHAT THE CHECKSUM SHOULD
; BE - OR LOOK IN THE ROM
; SOURCE FOR HOW THE CHECKSUM
; IS USED)
;
; CHECKSUM IS $3F7B
;
0000048000FE00F600FE003000FE0030
00FE003000FE003000FE003000FE0030
00FE003000FE003000FE003000FE0030
3E7C048042876000015C21CF029021CE
01F84E6E21CE01FC3C7C01F848E6FFFC
4E7553455256494345204D4F44450000
00FE003000FE003000FE003000FE0030
00FE003000FE003000FE003000FE00CA
4EFA25D04EFA24AE4EFA36644EFA052C
4EFA1EDE4EFA1CE04EFA0E164E714E75
4E714E754EFA052A4EFA08AC4EFA11F2
4EFA157E4EFA074C4EFA0A3C4EFA17CE
4EFA17BC4EFA0B3060FE423900FCE012
0839000100FCF801661431F900FCF000
01AA4A3900FCE01C4A3900FCE01E4A39
00FCE0104E73303900FC800002400FFF
0C400901664C02790FFF00FC80086642
33FC07000000800033FC090100FC8000
33FC0F0000FE8000427900FE80084239
00FCE01221CF02903E7C04806100FF00
49FA0006600006B495CA428097CB6000
23F44287303900FE800002400FFF0C40
0F00663002790FFF00FE8008662608C7
001E700033FC090000FC800033C000FC
800833FC0F0000FE80004E7049FA0006
6000066E428041FAFE6843FA3E62D058
E358B3C866F8D0586600FF1E4A876BE4
49FA000660000604DFA00066000006C
```

```
661649FA0004604E4DFA0006600000A2
6604600000F246474A4767024E704E70
207C000280007201740749FA00046010
207C00028008740549FA0004600260D8
200830813610E3494840E34848402040
534266EE4ED4303CA55A72007400247C
00020000007C07104ED4207C00008000
227C00FE8000267C00FE800849FA0006
60000072464049FA00046068464049FA
00046060E350B3C86704D1CA60DEB7C8
670A207C00008008224B60D04A424ED6
207C00008000227C00FE8000267C00FE
8008383C0C003210B14102410FFF6620
3084E350B3C86704D1CA60EAB7C8670C
207C00008008224B780060DA4A424ED6
844160DC30803210B14102410FFF6602
4ED484414ED4207C0000800870007200
380274007C00247C00020000267C0000
01007C1049FA000460CAD08BD1CA5346
66F2207C00FC8008700049FA000460B4
D1CA49FA000460AC207C00008000303C
070072007C1049FA00046098D1CA5346
66F4207C00FC8000303C090049FA0004
6082D1CA303C0F0049FA00066000FF76
4A426600FE9434047C0049FA00066000
FEC64A3900FCE00A7C0149FA00066000
00B0670000904DFA00066000FEFE6600
00844A3900FCE00E7C0349FA00066000
009067704DFA00066000FEA066664A39
00FCE0087C0249FA00046074675C4DFA
00066000FE866524A3900FCE00C49FA
00066000FE624A3900FCE00A7C014DFA
00066000FEAC662C4A3900FCE00E7C03
4DFA00066000FE9A661A4A3900FCE008
7C024DFA00066000FE88660E4A3900FC
E00C60144A3900FCE0084A3900FCE00C
E85E844608C700004A876B00FDA46030
383900FC800002440FFF0C440900661E
383900FE800002440FFF0C440F00660E
38390000800002440FFF0C4407004ED4
423900FCE01242802040224026402C40
72024841283CAA55A5A3604464349FA
0006600000A44A46675246464A466648
D7C1224BB3FC0020000066E213FC00AF
00FCE800303C61A8534066FC13FC002F
00FCE800207C00FCD814A1045FA0006
600000B249FA000660000350207C000F
```

```
FFFE2084261060FA46463C46204BB1FC
001000006F06207C00100000244BD7C1
224BB3FC00200000672820086604B651
671E49FA000460204A46670E46464A46
67DC300E464680463C40244B425360CE
4251D5C1224A605E7A20424632843343
0002B85167063011B9408C40B6690002
670830290002B7408C40334400023283
B8690002670830290002B9408C40B651
67063011B7408C404A4667064A995345
66BA4ED47060323C00FA740449FA0006
600005A44ED22A082C097009E0ADE0AE
247C00008008267C000080002A7C0002
0000263C00000100787E3005323C0700
49FA000460245344D083BC8066F24240
323C0C0049FA00046010538466F6EC8E
534613C600FCE80060664A3900FCE010
34803681D5CDD7CD423900FCE0124ED4
363C0FFF4A3900FCE0104A4267220C02
000167160C02000267084A3900FCE00E
60084A3900FCE00E60064A3900FCE00A
3012C0433213C243D5CDD7CD4A3900FC
E0084A3900FCE00C423900FCE0124ED4
2448264991C8327C080049FA00066000
08806738200AE088EE8813C000FCE800
49FA0006600001B445FA00066000FF06
303C61A8534066FC45FA00066000FEF6
91C8303CA55A3080321060FA307C0180
707F429851C8FFFC31C3018631CE0184
21CA02A421CB029497CA21CB02A8207C
0000800097C821CB011031C201B021FC
000002B002606104600000CA41FA003E
93C9704022C853406EFA612641FA0090
21C8000C41FA003221C8001041FA0036
21C8007C41FA006021C8002821C8002C
4E7547FA005E21CB00084E7521C701AC
7E0008C70007606C21C701AC7E0008C7
0008606021C701AC7E006100085C6620
08C70016610008DA4A3900FCE01C0801
000567060281FFFF800021C101A66034
08C70004602E21C701AC7E0008C70009
602221C701AC7E0008C70005600A21C7
01AC7E0008C7000631DF028021DF0282
31DF028631DF028821DF028A31C0028E
60000C2847FA03E421CB000861000952
47FA002C21CB0008207C00FCD9317008
4DFA00046022670A08C7000B4A876BE0
```



```

600A4A876BDA49FA0004605460747033
08C7000B600001622248D3C070004202
42104211363C00FF49FA000460024ED6
B4106620B411661C1083B4116616B610
66121283B610660CB6116608140351CB
FFE0600252404A404ED4103C00806002
70FF207C00FCD901117C0084001010BC
0004117C00FF00181140000808D00007
4ED420780110227802A849FA00066000
0680676E08C700156100064A61362278
01102049247C0000800091CA48E700E0
49FA00066000065A4CDF0700670E2808
6112224891CA2008EE2603621C80110
6114602E7211E2ACD844220348438641
877340004E7524380110223802A89282
203802949081E088EE8813C000FCE800
4E7561002832613449FA00066000FF4C
47FA054621CB0008207C00FCD8D7002
4DFA00066000FEF2670C08C7000A4A87
6BDE60000AC64A876BD6600E103A371F
7A037C5061002E544E7547FA002A21CB
0008612C65144A876BF020070280001F
FFFF670000BE60000A9208C7000C4A87
6BD860000A86703408C700126000FE3A
207C00FCD81117C0001001600280009
0018117C007F001C117C007F001A4240
611465107070610E650A705061086504
706061024E7548E7F8E040E7007C0700
207C00FCD8122482448D4FC00067440
76FF78061140001E323C061A53416732
091166F8C0FC0001323C061A53416722
091166F81483323C061A534167140911
67F8700A53406EFC4212117C0082001C
600846DF003C0001600246DF4CDF071F
4E7522780260347C02C0616C64FC6100
00DA428142834284615E65600C000080
67240C000087670E0C00000766027802
6146654860E67801613E65400C000007
66DA780260EA613065320C0000FD6604
760160DC0C0000DF620811C001B27602
60CE0C0000FF660408C7000D0C0000FE
660408C7000C60B861444E754A01660A
610000827201613664924A03660408C7
00174A0467145344671008C70018600A
08C70014600408C7000C21C902602007
0280001830004A8066000920606AB5C9
671A243C000001FF207C00FCD811028

```

```

001A08000001660A534266F2003C0001
4E751028000212C04E75207C00FCDD81
08900000002800010004203C00000BB8
61204E7508D0000061124E75203C0000
61A8600E203C001312D06006203C0006
7C28538066FC4E756104600000AA103C
00A07200740848E7108849FA00046006
4CDF11084E75207C00FCDD810028000E
0004021000F18510022800E300160028
001000164A3900FCC0316A1008390005
00FCC03166061600E40BD00311400010
117C000F0014363C00D051CBFFFE51C9
FFF6022800E300164ED408C700103E7C
048047FA001221CB0008207C00FCD901
4A106000FC0C08C7000B47FA001221CB
0008207C00FCD814A106000FC9608C7
000160000806610025E8327C1DF66100
29D4267C00FCE018287C00FCE01A2A7C
00FCF801303C0DF474024A534A540515
670651C8FFFA600C4A534A5405156704
4A53600C08C700024A876BC6600007BC
307C0240611064EC4A7900FCE0184A87
6BB06000016848E7018040E7007C0700
227C00FE8000247C00FCF8014E56FF18
47EEFF1849FA01422D48FFF872022D7C
00000007FFFC427900FCE018427900FC
E01A031266FC4C9100FF489300FF508B
508B4E71700853AEFFFC5FC8FFFE6EE6
2D7C00000007FFFC303C00AB51C8FFFE
4C9100FF489300FF508B508B4E717008
53AEFFFC5FC8FFFE6EE6427900FCE018
780147EEFF18284BD8FC00706100007C
4A4467646100009E47EEFF18D6FC0070
284BD8FC0070610000624A44674A6100
0084206EEFF8424010280018343C0064
C0C212280019343C000AC2D0411228
001AD04142414242424316301000D443
52410C41001866F21628001BD4430442
003CB440670242444E5E46DF4CDF0180
427900FCE01AE24C4E7542807202341B
E34AE310B9CB660AD7FCFFFFFFF905341
670E0C0000FF66E6E948E80830C04E75
42444E75740672084280E3DBE310B9CB
6606D7FCFFFFFFF90534166EEE948E808
30C0534266E04E754B4153002478007C
47FA009221CB007C2A7C00FCF8014A39
00FCE01C42824284303C01FF307C0300

```

```

36102248D3F802A44A3900FCE0063080
4A3900FCE0044A3900FCE01E4A426632
32104E714A42672A0815000166243839
00FCF0004A3900FCE01CEB8CB3C46612
21CA007C4240464030804A3900FCE01E
601608C700034A876B824A3900FCE01C
21CA007C600005B44A876B00FF706100
27C6600E74014E73703208C7000A6000
FB18327C1E046100276C6100F8D643FA
010421C9007C0807001E67047A016020
61000A14650E0839000600FCC18D6704
7A0260027A014A3900FCE01C612C660E
4A3900FCE01E61226604534566E84A87
6BBC08070015660005426100276247FA
F8A421CB007C60000198611849FA0006
6000003E670408C70015611C66EE0807
00154E757402484228380110307C0800
2242367C01864E75300348438640875B
B889670A2049D3C2B8896C0222444E75
2A48203CAA55A55A46807600007C0010
2080B09067064DFA0004604046802080
B098670A59884DFA000460305888E390
4680B3C866DA203CAA55A55A204D7200
4681007C0010B09067064DFA0004600C
20C1E390B3C866EE4A834ED42210B181
86814ED661526600F7EC08C7001621C0
026C21C80268263802A4610000C40801
00056604743F600A343C7FFF0281FFFF
800021C101A643FA002A21C9007C9283
22414A3900FCE01C4A3900FCE01E4284
181951CAFFFC605E0839000100FCF801
4E7561F46600F78E6100007621C10278
93FC0000000121C9027021C402740801
00056632220902810000000324016706
E188534166FAE1980280000000FFB900
671E080200006602E148367C01862809
26006100F8B04A3900FCE01C6000F796
220908010000670811FC0014027D6006
11FC0009027D7411E4A911C1027C60D6
4281323900FCF00031C101AAEB894E75
327C1E126100256E47FA00E421CB0008
610000BE54887200700010BC00023E97
1410B4016704700160643E9710BC0002
5281108166E460100900044D0B500C00
0D000E1303C105EA45FAFFEE323C0010
5588616A720076FF08100002660851CB
FFF8544060243E971141000408100000

```



```

660851CBFFF8584060103E9714280004
B401660876FF520166CE6002504011C0
02AC672008000000670408C7000FE248
4A00670408C7001061264A876B00FF5A
600002E8611A4A876B00FF4E604E1410
6002109A51C9FFFC4E75020009C00582
207C00FCD24145FAFFF2720461E0700C
6100FA0045FAFFE8720261D24E75B1FC
00FCD24166047038600270374A876A08
3E7C04806000FF026000F80E47FA0078
21CB0008207C00FCC0017A033C3C0051
702F610026161028003011C002A17202
6100054661624282227C00FCD9010229
00BF0010203C001C8000081100066606
538066F6743911E8001602AE66164A02
6612705511400002B028000266066100
0BD6640C08C700114A876B906000021C
4A876B88603A70394A876A083E7C0480
6000FF7A6000F7824280123900FCC031
4A016A160801000567047001600C0801
0006670470026002700311C002AF4E75
47FAF75421CB00086100F9246100F8FE
207C00FCD81117C00C9001861046000
00AC22780260347C02C06100F8926568
0C0000FF66246100F886655C0C0000FE
66366100F87A65500C0000C4660C11FC
000F01B308C7001C60D04A006A1E0C00
00FD67C60C000086660808F8000202A2
60B808C7001D60B2612460AE0C000006
660808B8000402A260A00C00007F6606
08B8000302A2609221C902604E7547FA
001A4282B01B670852424A1366F6747F
11C201B308C7001C4E75F4F1F2F3E401
E1E201E3D00101010101AF0061124A87
6BFA0807000E660001026100232A604E
40E7007C070070026100F6AC6534347C
01C0327C01B96100F7C665260C000080
66F06100F7BA651A020000F00C0000E0
66E072056100F7A86508534166F646DF
4E7508C7000E46DF003C00014E75327C
1E20610022807801610C4A876BF86100
22CE6000009648E740702C4F4281327C
0298247C00FC00012A78000847FA0014
21CB0008030A00006156640808C70019
60024259247C00FC400147FA001421CB
0008030A00006138640808C7001A6002
4259247C00FC800147FA001421CB0008

```

```

030A0000611A640808C7001B60024259
007C070021CD00082E4E4CDF0E024E75
0C41FFFF671032C16B060801000E6704
61000E3A4E7542594E756100EC9E47FA
000A21CB00086100FDF06100F3363E7C
04806100F4D261001D22200702800E7F
FFFF4A8067000220200702800000000F
4A80673A45FA25B508070001670A7029
610001086000F39A610002E2610002DE
610002D60807000067047028600C0807
00026704702A6002702B600001B42007
0280000003F04A806744610002B06100
02A845FA2930080700046704702C602A
080700056704702D6020080700066704
702E6016080700076704702F600C0807
00086704703060027031600001642007
0280001FDC004A806700008E45FA24DD
0807000A6708703261706000F4346100
024C61000244610002440807000B6704
703360520807000C67087034614C6000
F44A0807000E67047036603A0807000F
67047037603008070010670470386026
0807001167047039601C080700126704
703A6012080700136704703B60080807
00146702703C600000D861001FEE6100
013208C7001F4E750807000D67166100
01C8610001C8610001C045FA26997035
600000AE20070280006000004A806700
0070610001A8610001A06100019C0CB8
0008000002A86E14203802A40C800010
0006D04720160027202602608070016
6706203801A660E4307C018670084A58
6604534066F80C0000046E0472016002
7202080700156708704611C102AD6002
704721C7018045FA244661001EC46034
61000136610001366100013245FA2476
0807001B67047203600C0807001A6704
720260027201103801B461001E946004
61001F1821C70180610000586100FC14
6100F34E610019F660000FC23E7C0480
6100F10060E62007028018000004A80
672C08070017670645FA25AB60146100
023665160839000700FCC18D670C45FA
261B61001F42600000A0088700186000
00BE48E7F0003A3C00973C3C00126004
48E7F00002800000FFFF72014A406726
4282428380FC000A48401400E89A5243

```

```

424048404A40670260EAE99A10025343
6704611460F461064CDF000F4E756108
0645000A7C014E7548E7E0007408B401
6706E998534260F6E998610A534166F8
4CDF00074E752F000240000F0C000009
62060000003060080400000900000040
61002088201F4E75610E610C610E6100
F41461001A16601A702060027060323C
00FA74046100F4206100F3F24E7542B8
0180610018E842800807001C67101038
01B30C00000F665C6100015460560807
001D660001F46100012E64164A3802AF
670C74016100032067047001600E7002
600A103900FCC189E8086028323C1000
383C1800610001A0661A3602323C8001
383C9FFF61000190670434036A06C4FC
0003300211C001B34A00660E4A3802AF
67046000076A4200600A0C0000016608
103C0080600004560C00000267E40C00
00046E08227C00FC0001601A0C000007
6E08227C00FC4001600C0C00000A6E0A
227C00FC8001600009B60C00000F6700
0A8A0C000010667608B8000002A208F8
000102A26100194E6100199A47FA0052
21CB000861000060653E103900FCC189
E8080C00000F6630423900FCC18908B9
000600FCC18D47FA25FC61001F04207C
00FCC1950108000072046100FE627C0C
61000CB861000CFA6100EEC860000D7E
47FA25E145FA2125428060000D086100
0EDC600000C448E7C080207C00FCC181
303C001F320061444CDF01034E7548E7
C080E908123900FCC1890201000F8001
13C000FCC18908F9000600FCC18D207C
00FCC181701E61064CDF01034E753200
610A46435243078800004E7542824283
4A41670805080000588860023418D642
E35B51C8FFEC4A436704003C00014E75
48E7203045FAF9C447FAF9D14282B002
670C5242528AB7CA66F4700260021012
4CDF0C044E7548E790007400760241F8
029852423018C044B041670651CBFFF4
4A424CDF00094E750238000F02A24278
053A7C014A3802AF6602524674016100
01266602524642846100F9EC41F80298
30186A12247C00FC00016100026E6506
02430003DC4330186A12247C00FC4001

```



```

61000258650602430003DC4330186A12
247C00FC800161000242650602430003
DC430C06000A6F027C0A70122206C2FC
002247FA252461000E5831FC05A20530
31FC06580532103802AF67140C000003
661C4282610000A0661445FA213E6006
45FA217C7201740076FF610000E845FA
216E7202740176FF610000DA42820C00
0003671261000070660C45FA20D37402
76FF610000C0428041F8029830186A0E
247C00FC000172017403610001583018
6A0E247C00FC40017202740661000146
30186A10247C00FC80017203343C0009
61000132610015F808F8000502A26100
122665000CC26100F836610015BE6100
16AA6000FCB248E7AA80610005B4664E
610006F04A00671E4A02674042806100
0456610006D648E780806100167E4CDF
01014A00661E021000EF61000690660A
0C280001007867027052760061000690
610006A042280018001000184A404CDF
01554E7548E7804030026100FE143278
053061064CDF02014E7548E7F8A441F8
053A3410C4FC0005D442525831802000
61000C5838006100193ECCFC00083186
200231852004303C0090DC4031862006
06450022318520089DCE3C780532224E
DDF801104A036A1461001AE841FA2020
B5C8660C2A496100195C600461001916
D2FC0445610018F0424061001C1E3004
61001C1832780530D2FC0BF431C90530
32780532D2FC0BF431C905324E714CDF
251F4E7548E7F8E0080000D661A45FA
1EC476FF327805326100FF3A45FA1EB6
2A4961001900602C61306528227C0001
FFFC280302440003204A32182449D4C1
6100FF125344670C321852422449D4C1
6100FF024CDF071F4E7548E7A0000800
000D6720428461000624651A42833639
0002000408C30000247C0001FFFCDD5C3
161A600276014CDF00054E7547FAF5B8
21CB000811C00535610002CC207C00FC
C001227C0001FFFC4247C000200004281
0280000000FF4A406608117C00080002
6006117C00800002E098828010BC0086
610001F265000082267C00FCDD8108AB
000400046100024C6500006E42406100

```

```

0140640A0C00002767000060600A3029
00040C40AAAA6724123C000142406100
012065464201424061000116653C3029
00040C40AAAA67047026602E42B801B4
47FA009C49FA009E61000236247C0002
000048E780806207C00EE226100EE384238
02B04CDF01014ED2702711C001B40C00
00276720610001CC651211E800BA01B6
11E800C401B511E8005801B761000198
610000846100EA1C61000092103801B4
0C000007661445FA1E7E610018503A7C
287E61001780600002560C000017670C
0C00002667060C00004B660661001752
600845FA1C47610017D26000022E49FA
0004600849FA0004601060A031DF0280
21DF028231DF02864ED431DF028821DF
028A6100E3063E7C04806100E970704B
11C001B44ED4207C00FCC001117C0088
000210BC0087610000AC4E7545FA1FC0
123805354A0166047201600272024E75
243C00C0000048E7107840E7007C0700
6100011403C800041140000C42280002
10BC0081610000A8655610280010117C
00CC000210BC0085615A65444A006642
49E803E8074C000022C3074C000822C3
074C001022C349E80400303C001F074C
000024C3074C000824C3074C001024C3
074C001824C3D8FC002051C8FFE2600A
702746DF003C0001600446DF42804CDF
1E084E7548E71010263C00120000207C
00FCC001267C00FCD901022B00BF0010
4A106714081300066606538366F66004
538366EC003C00014CDF08084E752602
267C00FCDD81081300046608538366F6
003C00014E75611A6516243C00180000
117C0002000210BC008161D265026102
4E75117C00FF000210BC00856186267C
00FCDD817619081300046708534366F6
003C00014E75267C00FCDD8108130004
670261CE4E7545FA1B95610016804E75
307C000820CB20CB721420CC534166FA
307C0008722020CC534166FA4E7561D6
227C0001FFEC247C000200004281243C
01200000760A78036100008665223029
00040C40AAAA67047054601447FA0060
49FA006461AA247C000200006000FD74
0C38000302AF670645FA1B85600445FA

```

```

1BBA0C000050660E0807001C66086100
11AA6000F8FA610015A26100F6E66100
F7886100F7846100F780610011C80238
00FC02A208F8000002A26000064049FA
00066000FDB849FA00066000FDBE60A0
48E73F7E40E7007C0700617467047050
605C081000016608538266F67051604E
610000B664186100018A610000AC640E
610001A46100017C6100009E65304A78
01B66A066100009265244AB801B46712
223801B420010280C140C00067047053
600C7004615C707F224A615660084DF
003C0001600446DF42804CDF7EFC4E75
4280267C00FCDD81001300A0002B00A0
0004207C00FCD9010228007B00600028
006B00604228001800100018021000FB
022800FC00100028001C001008100000
4E7512E8000812E8000812E8000812E8
000851C8FFEE4E75367C030426811743
000417440005611E651A7402615E6514
367C01B416E8000816E8000816E80008
16E800084E7548E728007401613E6412
0C000055672C610000AA4A006624612C
6524021000F7117C00FF0018303C0005
115B000851C8FFFA0010000842280018
6004003C00014CDF00144E7548E77800
021000EF422800184280613066181228
00784203B2026704705260027655612E
4A006602613C42280018001000184A00
6704003C00014CDF001E4E75383CFFFF
081000016706534466F670554E750210
00E7117C00FF00181143007800100010
4E75283C00180000600E283C01200000
6006283C000005000810000166065384
66F670554E75267C00FCDD810213007F
6100E97A001300806100E9724E751800
2A7800082C4F47FA002C21CB00080309
00004A416A280C41FFFF67226100FD28
24496148651C103801B3247C00020002
6000FAF0705A21CD00082E4E6008705B
6004183801B311C001B445FA18680C04
00046E047201600C0C0400076E047202
600272036100128A6000FD7048E770C0
207C0001FFFC224A4280010A00045440
0C400FFF624442824283050A00003082
3418D642E35B588A534066EE050A0000
D6424A4366244A44672A0801000E6724

```



```

48E70F3E4EB9000200004CDF7CF04A40
671211C001B5705D6002705C11C001B4
003C00014CDF030E4E750C39000100FC
C1916756207C00FCC191227C00FCC1FF
42105488B3C866F8700113C000FCC191
13FC003C00FCC1C342B801BA427801BE
702C6100E6E2651E42817408610001E8
6514223C100000007408610001DA6506
70256100E6C2650001B061000EC80838
000202A26718103900FCC1C30C00003C
660470036002703C13C000FCC1C30C39
000100FCC199672203801BCE9984840
13C000FCC19B423900FCC1C5423900FC
C1C113FC000100FCC1990C39000200FC
C1C566000082423900FCC1C547FA1AEB
610013F47C0C47FAEE7E21CB0008207C
00FCC001267C00FCDD814A3802AF6704
785060124281782D6100FB4865166100
014E6510782D4281123C0008E8996100
013E642047FA1ACA610013AC7C0C0C00
00276700010A523900FCC19F64065239
00FCC19D610001AA117C0088000210BC
00876100FA9047FA1A56610013841039
00FCC1C36100F2AA7C0C523900FCC197
6406523900FCC19561000130203801BC
E9984840B03900FCC19B6712523900FC
C1C5523900FCC1C113C000FCC19B6100
E714103900FCC1C1123900FCC1C3B001
6C16103C000F6100F4766100FC146604
6100FD486000026C423900FCC1992038
01BC227C00FCC1A101C90000207C00FC
C00110BC00896100F9FC6552702D6100
E54665384281123900FCC1C3703CC2C0
700CE1B9227C00FCC1B103C900007405
61346518103C000F6100F41470236100
E51665084E7160FC703D6002703E45FA
14FB610010866100F1CA6000014008C7
00116000EF36E99910010200000F0000
00106100E4E26504534266EA4E75117C
0088000210BC00866100F97A653608AB
00040004243C00C00000063C800044228
000210BC00816100F996651610280010
6100F9C0650C4A00660A5241534466DA
4E757027003C00014E7547FA19146100
123052466100EDCA203801BC227C00FC
C1A101C90000E99872016100F18C5246
E19872026100F1825246E19872026100

```

```

F1785246E19872026100F1647C0C4E75
48E7C01047FA190D610011E6267C00FC
C19D010B000072046100F1447C0C4CDF
08034E756100DB044287423802A208F8
000102A248E780306100F7FC6100E55C
6100E57261000A764CDF0C013E7C0480
48E780306100E32061000B6C61000BFA
4CDF0401220A670461000FB24A406712
220A660A7A7E7C126100F0A660046100
F092265F200B670461001140007C0700
6100E10A4278053A0238000F02A208F8
000602A20838000102A2663020380180
0280001E3FFA661E4A7801886618327C
2956103C00F147FA18A0347C2CE84281
61000C98600608F8000102A208380000
02A26616327C1876103C00F447FA185A
347C1C08428161000C72327C3A36303C
00F247FA1884347C3DC872FF61000C5C
610009EC08F8000502A26100061A6500
00B6610009B60C0000F2660861000A9C
6000F2B60838000002A266120C0000F4
660C42874A3900FCE0106000DB2E0838
000102A2666C0C0000F1666661000A6C
02877000000020380180080000006600
E1000280008FFFFF6700F05C2F006100
0AF0103C00706100E2BE201F08000002
670C327C1DF661000ECC6000E6B0EE88
4A006600EC3AE4884A00670C327C1E12
61000EB26000EA464A3900FCE01E6000
E9300C0000F667000064610000306100
092E6000FF4645FA126397CB6000FE6E
428020780110323C1FFD20C051C9FFFC
4E75610006344A006AF84E757020323C
00FA74046100E3E04E7548E7408041FA
119C32000241007F044100206A047002
6004103010004CDF01024E756100099C
610000D26100005A610008C4610004F8
6594610008960C0000F4670000DA0C00
00F16700013E0C0000F26700019C0C00
00F3670001C00C0000E4670002780C00
00E1670002E80C0000E2661008B80000
02A2428095CA97CB6000F262600002D6
4278053A0238000F02A270127207C2FC
000B47FA170D61187807327C05A2347C
065847FA170549FA177161000B9E4E75
48E7C00008F8000702A2610009385441
327C05A261000A00327C011161000C18

```

```

428161000E4097CA240B5442020200FE
3002720E74FF92FC005B610009224CDF
00034E75327C0EDA7042223C00000140
47FAD830610009A031FC003E030031FC
001803024E756100036C650002384A43
670002400882000024426100034A6506
0C000020670847FA1774610002367204
610003506500020E610002E64A436702
60027410200A72086100EDFE58467808
301A72046100EDF25246534466F25182
51826F06610002BA60DA610002B46000
01E261000300650001CC4A43670001D4
2442610002E265060C000020670847FA
1705610001CE7208610002E8650001A6
4A43670001AE0C0300026E0414C26014
200A0880000024400C0300046E0434C2
600224C2610002A0650A0C0000206600
017460C26000017C6100029A65000166
4A436700016E0882000021C201F84DF8
01C04CDE3FFF2C7801F84E966100D70C
600001506100FEDE6100021647FA1602
781861000DBC47FA168C610001466500
01244A4366086100FEBC600001265343
6600011272016100024A0C02000C6200
010453426B0000FE2F026100075E6100
0800241F0C0200046C06327C1DF6601C
0C02000A6C06327C1E1260100C02000B
6C06327C1E046004327C1E2061000BC6
D44208C7001F41FA0024D0F020004ED0
4A3900FCE0106000D7E808F9006000FC
C18D70026100EE786000E434D7B8FFE4
E1C6E380DDA4DED4DF0EE62CE730E8B0
FFFFE91A70FF6100FCFA42806128701B
721C740C6120D04151CAFFFA42806130
702C722D740F6128D04151CAFFFA6100
FCE26000FD1848E7C080725AC0C12078
0110D1C04218534166FA4CDF01034E75
48E7F080720880C14282340020780110
D1C2484092405341745A363C016C0390
D1C2534366F84CDF010F4E756100066C
6000F7D26100FC9647FA15616100018C
6004610001A46100FD186100005826000
FCBC48E706206100071A3A3C00183C3C
001861000974544231C2052E61000C52
610000084CDF04604E75207C000002C0
224842836100FC3C6100FC504A0067F4
0C00000866124A4367EA5343A216100

```



```

008C6100009660DC0C00000D67180C03
00306D066100FC1660CA524361000088
61000C3860BE4E7548E7E0807A487C18
61000C08204ED0FC0384343C001B323C
000A303C00423CD855406EFA52457C18
61000BE8204ED0FC0384534166E45342
66DC3A3C014C3C3C00184CDF01074E75
3A3803007C180645000A0C45014C6F02
61A631C5030031C603024E755346BC78
052E6C043C38052E4E75303C00206100
0BBA53464E752F0A347C0300B5C96702
12C0245F4E75B3C8670410186004003C
00014E7547FA14156100FEE872086102
4E754283428261DE652E0C0000206604
4A2060240C0000306D240C000039630C
0C0000416D180C0000466E126116E98A
84005243B20366CE023C00FE6004003C
00014E750C0000406E06040000306008
040000410600000A4E7548E786006100
05A23A3C00183C3C001861000AFE6100
0AE04CDF00614E75705A3C7C05FA4281
2A4E223C00000708068100000168DBC1
610004AC4E750838000302A2666C6100
00E80C000006660808B8000402A260EE
0C000086660808F8000402A260144A00
661A61000376610003960838000402A2
67CC610001C267C6600000820C0000FF
660808F8000302A260200C00007F6608
08B8000302A260A60838000502A2669E
4A006A92610001EC4E756100007C0C00
007F660808B8000302A260820C000006
660808B8000402A260E00C0000866608
08F8000402A260144A006618610002FC
6100031C0838000402A267BE61000148
67B860086AB46100019A4E7532006100
00280C000006660A08B8000402A23001
4E754A0066E8610002C2610002E26100
011666D86000FF10610000844A006708
0C00008067164E75617411C0048A616E
11C0048B610001D442404E7561600C00
00DF63180C0000EF63180C0000FB6500
001867160C0000FD630E601011C001B2
60B611C00480601660AE604C0C0000FE
6604703460027035003C00014E7548E7
60E043F8048145F8048672056100DCD0
6504534166F64CDF07066000FF7C2F08
207C00FCDD811028001A0800000167F6

```

```

10280002205F4E75080700116622207C
00FCC0016144422800046100F06A117C
008000046100F06010BC00896100F006
49FA00066000D9FA203C0003D0906100
DCD270216100DB406100DCC26100F8EE
45FA0B2970346000E798117C00880002
10BC00866100EFCE267C00FCDD8108AB
000400044E7548E77F803C3804963E38
049841F8053A301867424C98003EBC42
6D0EBC446E0ABE436D06BE456E02600E
4A416A046100004C534066DE601E4A41
6B186100003E534067104A586B045048
60F44C98003C6100002A30014CDF01FE
4E7548E77E8041F8053A3C184C98003E
B2006706534666F4600261064CDF017E
4E7548E7F8406100012208680007FFF6
3004904280FC000832059243524193C9
E64AD2C2785AC6C4D2C30838000702A2
6706D2FC005A534174FF48E7E0406100
022E4CDF02070838000602A267066100
03E8600C0838000702A2670461000424
610000EC4CDF021F4E7548E77C003038
0486323804881438048A48821638048B
488338026C024443A036C02445D845
9878048E6E16D0423403D643D643D642
54436D025643E643D2436010D2433602
D443D4436D025242E242D0424A406C02
42400C4002D06F04303C02D04A416C02
42410C41016C6F04323C016C31C00486
31C1048831C0049631C104984CDF003E
4E7531FC0168048631FC00B6048831FC
0008048E707C6100D98E4E7542780490
4278049231FC0010049431FC01680496
31FC00B6049861CA4E7548E7C0C041F8
04A22278052830380526323C005A6004
2298D2C151C8FFFA4CDF03034E7548E7
FCF041F804A22278011045FA090047FA
08FC3038049032380492383804949078
0496444034000242000F444206420010
42834643E5AB0240FFF06C0A4240E18B
E18B064200100C4002B06F140C4002D0
66027600303C020B0E08BE08B06420010
9278049844416C0CD841D24194C196C1
424160103A3C016C9A44B2456F0C383C
016C98414A446C02424431C0052231C1
052431C40526E648D2C07A5AC2C5D3C1
21C905286016301AE5B8C083321BE5B9

```

```

C283468120D1C391B191D2C551CCFFE8
4CDF0F3F4E754282612E705A74FF327C
05A0720161423C7C05FA3A7C7FF8243C
AAAA555548427201224E6100002CDCC0
BCCD6DF04E7574FF323C016B705A93C9
61164E7548E7E040705A7210740093C9
61064CDF02074E7549FA0020600449FA
001ED3F801102F0A4283360044830683
0000005A2449D5C04ED432C26002B559
B5C9670260F2D3C3D4FC005A534166E8
245F4E757A597C0C48E7C040704E223C
000000A4327C1140616C4CDF02034E75
48E7C04070467254327C114461587A40
7C0E47FA0CA672FF6100049A610003EA
610003FA610003EC610003FC4CDF0203
4E7548E7C04070427214327C071E6126
4CDF02034E7548E7C040611A2F09D2FC
05A0720174FF6100FF50225F610001DA
4CDF02034E7548E7F87848E7C0404282
6100FF3693C393C0720174FF6100FF38
5449720155406100FF2E4CDF020348E7
404072016100FF124CDF041074015544
5540D3C0220442806130303C00B67407
22046136303C0110740622045341612A
224AD3F80110705AD3C0343C80002204
428061064CDF1E1F4E75338200005341
67060640005A60F24E7505F100005341
67060640005A60F24E7548E7402045F8
053A3412C4FC0005D442525A35802000
6100F48838006100016ECCFC00083586
200235852004700A721C614C2F093278
052C61000152CCFC0008358620063585
2008225F48E7F870E2886100013ADC80
E2895981DA81534642406100045E2004
610004584CDF0E1F4CDF0402224A6100
0116610003404E7548E7F87074FF7201
6100FE2693C3222F0004428074076100
FF5A226F0014D3F80110222F00044280
74076100FF46D3C02017720174FF6100
FE0693C393F8011031C9052C4CDF0E1F
4E7548E7C040D3F80110740742804EBA
FF1A2017222F0004D3C0534942804282
4EBAFF084CDF02034E7548E7F80C0FC
000841F8053A32184282263C000003DE
5241141C30C2617ECCFC000830C630C5
DC4030C60645000B30C52F09224A6166
6100035E225FD3C3D5C3534466D231C1

```



```

053A4CDF010F4E7548E7FEFE615AE288
E28A908261405845DC8048E706006100
03304CDF0060594555460246FFFE9DCE
6100033C224E266F00286100002C5242
0242FFFE20025840720F74FF6100FD30
4CDF7F7F4E752F02240984FC005A3A02
48423C02241F4E75244B141A66FC95CB
240A4E757005721F61024E7548E73802
765A42843400600252441D9A400051CA
FFF8DDC351C9FFEC52464CDF401C4E75
610000DA3A7C287E610ADB801106100
008C4E752F00224D619C47FA05A8B7CA
6608064500165646603847FA0552B7CA
660806450010584660284A3802AF6646
47FA088CB7CA66080645001258466012
47FA067CB7CA66065C45564660045B45
5846610002665341660645FA0480600E
5341660645FA047B600445FA047A7000
72046100FF58201F4E756100FC8C7A73
7C10610002362A4E610000F845FA079A
610845FA07E261024E7548E7F0062C4D
5C4E48E70006610000DA4CDF60007654
741F72023016815D425E51C9FFF8DBC3
DDC351CAFFEE4CDF600F4E7548E7C020
6100FC367A737C10610001E0220A6A0C
0881001F24416100FEDC600461000094
4CDF04034E756100FC107A737C106100
01BA2A4E617C45FA06FD2C4DDCFC0006
61704E7548E7E0007006722074FF6100
FBAE4CDF00074E7545FA04013C7C1DF6
601C45FA043A3C7C1E04601245FA03AD
3C7C1E12600845FA046C3C7C1E20DDF8
0110612E4E7561D03A7C1DF6601661D2
3A7C1E04600E61D43A7C1E12600661D6
3A7C1E2045FA06B2DBF801106100FF1C
4E7548E7E0A0204E72177405303C0100
101AE24867146404421E60021CDA51CA
FFF2DCFC0054740560E851C9FFE02C48
244ED4FC005A323C001E201EB19A301E
B15AD4FC0054DCFC005451C9FFEE4CDF
05074E7548E7A00045FA0887B7CA6770
244B103801B267680200003F14000202
00F0675C0C02003066080C00003D6726
604E0C020020661E0C00002F67420C00
002E67320C000026672C0C00002D672A
0C00002767240838000702A266120445
000A61240645000A611E0645000A6010

```

```

6130612E600A4A1B66FC4A1B66FC244B
61064CDF00054E7548E7060061422F0B
4A41670647FA07F76136265F4CDF0060
4E75612C702F61624E7548E70E003A3C
007E7C18380561184CDF00704E75CAFC
000A610C4E7561080645000A7C014E75
2F000C45014C6F046100F3FE4280101B
6704612660EC201F4E752C7801102F00
42803005C0FC005A0640005AD08E2C40
42803006DDC0201F4E7548E7C02261DA
0240007F4A0067380C000020672C0C00
000D67320C00003F67340400002D6D14
0C00000C6F2E5F000C00000D6D060C00
00266F2045FA013A602645FA003E6020
45FA0134601A0645000A3C04602645FA
011A600C3200E7489041904145FB0022
428072054216DDFC0000005A6100FC7E
422E02764CDF44034E75000000000000
0000007C000000000000003004081020
40803844444444380838080808083844
0810207C384418044438081828487C08
7C40780444383840784444387C081010
20203844384444383844443C04383048
84FC8484F884F88484F8788480808478
F884848484F8FC80F88080FCFC80F880
80807884809C847C8484FC8484843810
101010381C08080888708890A0D08884
8080808080FC84CCB484848484C4A494
8C84788484848478F88484F880807884
84849478F88484F88884788460188478
FE101010101084848484847844442828
1010828292AA44444428102844828244
28101010FC08102040FC384408100010
C7BBF7EFFFFEF0877FEFE7F3E1B2D112
37383914343536132E32330300000000
00000000000000000000002D3D0000
500800000D3000002F31000039305549
4A4B5B5D4D4C3B27202C2E4F45363738
35525459004647485643424E41323334
31515357005A5844000000008000C000
E000F000F800FC00FE00FF00F800F800
CC0080C000600060003000300E0606060
F0E03060C0F0E0306030E0FFFFFF3F03
FF00FFFFFFE001FFFFFFE4FFC0702FF9
50669EC00180F70379560674FE0FFFFF
FFFFFFFFF3F01AA60AAFFFC00307C4
FFE0555550F1FFFFFF8FFFFFFFFF3F03

```

```

FF00FFFFFFE001FFFFFFC4FFC079F240
F938E0BFEOEFF07138018079FC03C0FF
FFFFFFFFFFFF3F01FEC0AAAABFC003FC
841FE001555540E103FFFFE0FFFFFFFF
FFFFFFFFFC31FFFFFFC0800FFFFFF8007FF
E0E303FFF0FFFFFFF71045E88021C8E50
038C70C7010C207B0E040C5E8803FFD5
5700FF8007FF8003FFC0F32AA8FC7FFC
FFFFFFFFFFFFFFFFFFFFE101FFFF80FC
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFD03
63EAD80013CC0061550CF3FF8FFFFFFF
611FFFFFF80F18FFFFFF00FFFFFF86F011
FFFF886D05A00218800140014FF2CE80
A815F3542A3C2A541428CF0810FF3C08
101428CF2814F3500A1CA00501466286
800288114061051008A01118E007880F
FFFF86F00FFFFFF0E11FFFFFF8FFFFFFF
FFFFFFFFFFFFFFFFFFFF001FFFFFFF83F
FFF0FFFFFFFCFFCC037F0180CF037FFC
0180003FFFFFFFFFFFFC1CFF60FFF3FF38
030CFDC0FFFFFFFFFFFFFFFFFFFFFFFF
FFFFFF2F081FFF08FFFC101FFFFFFC02
201FFFFFFFC47F8201FFFFFFC801FFF08
FFFC801FFFFFFFC8047F81FFFFFFC20E0
1FFFFFFFC10FB08FFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFE023803FF0407FF22
883FFFF811FEE87FFFC208B207FFFC
A011FE3FFFF808E307FF84F803FF02FF
FFFFFFFFFFFFFFFF7907FE0FC6FF7FFFF0
71FFFFFF80A18AAAAA8FAAAAF6A060E3
03C007380420060810FFF308103C0420
03C0FFFFFFFF611FFFFFF80F38FFFFFF0
0180CF0240F304203C08101008CF1C38
FFFFFC03C0CF01C0FF3F0E38CF0808F3
04103C02200140DF80F301C03F014020
1B6DB6DB6016DB086DB6DF3B6DB6DB80
678006DB6DB6D8FF0F4FFFFFFFFF00FFF0
FFFFFFFFFFFF8001B6DB6D836C01B6D00
B6D836C01B6DB6D80036C01B6DB6D836
C0001B6DB6D836C01B6D00B6D836C01B
FFFFD80036C01BFFFFD836C000FFFFFFF
FFFF87FFFF0FFFFFFF0FFFEDE0FFF307
1C3C04040208CF0110F7A03D4001E0FF
F301207F3F9CFF80FFFEFFFF3C7E7CFF
FECF1FF0F30FE03C0FE01FF0FFFFFFF
FFFFCFFFFFFF00000000000000000000
F30380FE0CCF0380F7E0BF03EF03FFFF

```




```

03FFFFFFFFFFFFCF0780F30E80BE1B
36EF6C7BD8019EB00360E706C0790D80
1BDF36F76C3DD801B0CF0360F306C0BC
0D801BEF36FB6C9ED801B0E7036038F8
06C0DC0DCE80361BF31B363C0DEC06D8
CF0330F301E03C03C002E0CF01B0F7D8
7D6C36DF1BE70D807906C0039E6001B0
EFD8FB6CBE361BCF0D80F306C03C0360
01B0DFD8F76C7D361B9F0D80E706C079
036001BEB0D8EF6CFB363E1B0E80FFFF
FFFF1F03FFFF86F807FFFF4EF0AFB05
BC028001FF7D7884CF0102FF7C010284
DF78FFFFFF7F0710FFFFFF8003FFFFFFE
FFFFFFFFF0F03FFFFFF00FFC007FFFF
FFFFE0100CFFFFFF03001FF70FFF80780
019EFF01FFCF0780FF3F0780E701FF09
01FF01FFFFF84007800CFFFFFF0300007
FFFFFFFFE003FF00FFFFFFC003FFFFFFF
00FFE007FFFFFFFFF061AAAAA95FF18
FFFF9F5FFFD861F0AAAA8153F07FF00
FFFFFFFF003FFFFFFFFCFE0FFFFFFF504F
574552204359434C494E472041542000
54494D45204953200044524956452054
455354004C4F4F5020434F554E542049
532000504D20425553204552524F5200
464C4F5050592054455354204641494C
454400464C4F505059204552524F5220
434F554E54204953200054455354494E
47005445535400455320574952442047
45544553544554005245535441525400
5245434F4D4D454E434552004E455520
5354415254454E00434F4E54494E5545
00434F4E54494E554552005745495445
524D414348454E005354415254555020
46524F4D0044454D4152524552204445
005354415254454E20564F4E002E2E2E
004F5054494F4E5300444953504C4159
204D454D2020203100534554204D454D
4F525920202020320043414C4C205052
4F4752414D202033004C4F4F50204F4E
20544553542020340041444A55535420
564944454F20203500504F5745522043
59434C45202020360051554954202020
202020202020203700F4F1F2F3E4E1E2
31202D20524F4D0D32202D204D4D550D
33202D20564944454F0D34202D205041
524954590D35202D2050415241205649

```

```

410D36202D204B594244205649410D37
202D20434F50530D38202D205343430D
39202D204449534B0D41202D20434C4F
434B0D42202D204D454D4F52590D4320
2D20494F20534C4F5453004144445245
5353203F0044415441203F00434F554E
54203F0054455354203F005748415420
3F000000000000000000000000000000
000000004338344150504C4502483F7B
;
;THE END

```

###

