Chapter XXX

Lisa Boot ROM Information

NOTE

THIS IS A PRELIMINARY CHAPTER FROM MY LISA LEGACY PAPER REVISION. THIS CHAPTER _WILL_ CHANGE IN THE FUTURE WHEN I COMPLETE THIS PAPER.

Author

David T. Craig
71533.606@compuserve.com

X.1 INTRODUCTION

The Lisa featured a 16K byte ROM containing 68000 code that tested the Lisa's hardware, booted the Lisa operating system from the startup disk drive or a slot-based boot card, and provided a special service facility called the ROM Monitor which normally only Apple's manufacturing and service people used.

The Lisa source code, version 2 revision H dated 24 February 1984, provided a concise description of the Boot ROM's purpose and user controls. This summary follows:

```
<  1> ; Function:  Initializes LISA system for use and performs preliminary
<  2> ;            diagnostic checks.  If all tests pass, the system then
<  3> ;            does a keyboard scan to check for user input.  If any key
<  4> ;            is hit other than caps lock or the mouse button,
<  5> ;            a menu is displayed on the screen showing the available
<  6> ;            boot devices.  If a valid COMMAND key sequence is detected,
<  7> ;            a boot from an alternate device is attempted (see below).
<  8> ;            If no keyboard input is detected, the system first checks
<  9> ;            parameter memory for a valid boot device and, if none, defaults
< 10> ;            to booting from a Profile attached to the builtin parallel port
< 11> ;            for Lisa 1 systems.
< 12> ;
< 13> ;            For Lisa 2 systems, a check is first made to verify a disk
< 14> ;            (internal or external) is connected before defaulting to the
< 15> ;            hard disk boot.  If no disk is detected, the system defaults
< 16> ;            to booting from the floppy drive.
< 17> ;
< 18> ;
< 19> ;
< 20> ; Inputs:    Checks for keyboard input from the user.  Currently, the following
< 21> ;            key sequences are honored if input after the system "clicks" the
< 22> ;            speaker (CMD refers to the Apple key on the keyboard):
< 23> ;
< 24> ;               CMD/1 - boot from Twiggy drive #1 or integral hard disk
< 25> ;               CMD/2 - boot from Twiggy drive #2 or SONY drive
< 26> ;               CMD/3 - boot from Profile attached to parallel port or integral hard disk
< 27> ;               CMD/4 - boot from I/O slot #1, port 1
< 28> ;               CMD/5 - boot from I/O slot #1, port 2
< 29> ;               CMD/6 - boot from I/O slot #2, port 1
< 30> ;               CMD/7 - boot from I/O slot #2, port 2
< 31> ;               CMD/8 - boot from I/O slot #3, port 1
< 32> ;               CMD/9 - boot from I/O slot #3, port 2
< 33> ;               CMD/ENTER (on key pad) - abort boot, branch to ROM monitor
< 34> ;               CMD/SHIFT/P - abort boot, do power cycling
```

For detailed information about the Boot ROM's operation see the following Apple publications:

Lisa 2 Owner's Guide	**(release 2, dated 1983)**
Lisa 2 Owner's Guide	**(release 3, dated 1984)**
Lisa Office System	**(release 3, dated 1984, p. 162)**
Lisa Level 1 Technical Procedures	**(June 1986)**

The Lisa Level 1 Technical Procedures manual had the following to say about the Boot ROM:

Interpreting the Boot ROM

On power-up, the Lisa will automatically perform a self-test using the program in its boot ROM. As the CRT warms up (approximately 30 seconds after power-up), the monitor will display 4 icons representing the CPU, I/O, memory, and expansion cards as they

are being tested. Each icon will be hilighted in sequence as the boot ROM checks them.

The boot ROM diagnostics are an overview of the system. If one of the cards is faulty, its icon will be presented on the screen with an "X" through it. The testing will stop. Upon failure you should reseat the board, power-up, and if you get that error message again, swap out the board. If the system passes, and ROM code numbers are correct, you can use the Lisa/Macintosh XL Test diagnostic for further testing.

The Boot ROM was written in around 12,000 lines of 68000 assembly language. The source code existed as a set of 6 files with the following file names and purposes as found in the source code:

```
<  1> ; Filename:  RMXXX.Y.TEXT, XXX = ROM VERSION # (e.g., 200 for 2.00)
<  2> ;                           Y = E (equates)
<  3> ;                             = K (kernel tests)
<  4> ;                             = S (secondary tests)
<  5> ;                             = B (bootstrap code)
<  6> ;                             = M (monitor code)
<  7> ;                             = G (graphics, icon and message display)
```

The source file names for version 2.48 (48 is the ASCII value for "H") were:

RM248.E.TEXT
RM248.K.TEXT
RM248.S.TEXT
RM248.B.TEXT
RM248.M.TEXT
RM248.G.TEXT

X.2 BOOT ROM HISTORY AND PROGRAMMERS

The source code for the Boot ROM provided an extensive modification history. This history listed the main programmer, Rich Castro, and within the source code the names of other ROM programmers appeared:

Rich Castro	**(main programmer)**
Ken Schmal	**(video PROM serial number reading)**
Ron Hochsprung	**(video PROM serial number reading)**
Rick Meyers	**(mouse & cursor I/O)**
Mike Urquhart	**(mini-graphics package)**

The modification history follows:

```
<  1> ; Originator:  Rich Castro  7/30/81 - Version 0.0 released to manufacturing
<  2> ;
<  3> ; Modified by: Rich Castro  7/30 - 11/3/81 - Made the following changes:
<  4> ;                                    1) Twiggy bootstrap capability
<  5> ;                                    2) Initial COPS test and keyboard scan
<  6> ;                                    3) Moved parallel card to slot 2
<  7> ;                                    4) Changed ROM interrupt/exception vectors,
<  8> ;                                    5) Created jump table for ROM routines
<  9> ;
< 10> ;                           11/3/81  - Version 0.7 released to the world
< 11> ;
< 12> ;                           11/4/81 - 1/15/82 - Made the following changes:
< 13> ;                                    1) Added support for new memory cards
< 14> ;                                    2) Added warm-start capability and jump
< 15> ;                                       table for ROM subroutine usage
< 16> ;                                    3) Modified MMU reset routine to support
< 17> ;                                       single step board usage
```

```
< 18> ;                                  4) Added full memory initialization
< 19> ;                                  5) Added 256K memory parity test
< 20> ;                                  6) Modified COPS initialization so that
< 21> ;                                     keyboard commands can be sensed more
< 22> ;                                     reliably
< 23> ;                                  7) Added error code display routines and
< 24> ;                                     display of CPU and IO ROM versions
< 25> ;                                  8) Added preliminary disk controller test
< 26> ;                                  9) Updated warm-start check
< 27> ;                                 10) Modified disk interface test
< 28> ;                                 11) Changed low memory assignments
< 29> ;                                 12) Made corrections for no I/O board, disk
< 30> ;                                     interface error and contrast setting
< 31> ;                                 13) Modified memory sizing routine to
< 32> ;                                     catch memory errors
< 33> ;                                 14) Modify MMU test to avoid context 0
< 34> ;                                     destruction, add contrast setting for
< 35> ;                                     new machines, correct disk error and
< 36> ;                                     CPU ROM messages
< 37> ;                                 15) Move stack so old memory test still runs
< 38> ;
< 39> ;                           1/15/82 - Release version 0.16
< 40> ;                           1/18/82 - Fix stack problem and release vrsn 0.17
< 41> ;                           1/19/82 - Change stack for call routine and version
< 42> ;                                     to 0.18
< 43> ;                           1/27/82 - Change MMU error routine to do address
< 44> ;                                     and data line toggling
< 45> ;                           1/28/82 - Add video circuitry test
< 46> ;                           1/30/82 - Add write wrong parity test
< 47> ;                           1/31/82 - Move run time stack to $180
< 48> ;                           2/6/82  - Add Profile bootstrap with upgrade for
< 49> ;                                     OS use (add jump table entries also)
< 50> ;                           2/15/82 - Update Twiggy bootstrap and add entry
< 51> ;                                     for OS use; also add MMU test to
< 52> ;                                     conditional assembly and add context
< 53> ;                                     saving to MMUTST2 routine
< 54> ;                           2/17/82 - Add correction to memory test for
< 55> ;                                     reboot problem and leave parity on
< 56> ;                           2/24/82 - Add code for clock test and special
< 57> ;                                     burn-in cycling
< 58> ;                           2/25/82 - Add code to simulate soft on switch
< 59> ;                                     pressed for COPS problem
< 60> ;                           3/1/82  - Removed all changes since ROM 0.18
< 61> ;                                     release except for parity enabling,
< 62> ;                                     no reset feature, memory sizing change
< 63> ;                                     and Profile booting
< 64> ;                           3/1/82  - Restore default stack ptr loc to $300
< 65> ;                           3/1/82  - Move default stack to $0400, restore
< 66> ;                                     everything except MMU testing
< 67> ;                           3/4/82  - Add MMU initialization and modify
< 68> ;                                     Twiggy, Profile boot routines for new
< 69> ;                                     load point
< 70> ;                           3/10/82 - Add change for new I/O addresses and
< 71> ;                                     fix for Twiggy routine
< 72> ;                           3/10/82 - Change contrast value for new I/O's
< 73> ;                           3/15/82 - Add correction for Profile and COPS
< 74> ;                                     routines and display msg when booting
< 75> ;                           3/17/82 - Restore version # at end of file
< 76> ;                           3/18/82 - Release version 0.22
< 77> ;
< 78> ;                           4/5/82  - Make initial 2732 version (1.00); add
< 79> ;                                     following changes:
< 80> ;                                       1) correct MMU error routine bug
< 81> ;                                       2) change stack for CALL to $0400
< 82> ;                                       3) add parity disable to WWP routine
```

```
< 83> ;                                   4) change MMU I/O space code to '9'
< 84> ;                                   5) add invalid boot code message
< 85> ;                          4/6/82  - Add speaker click after COPS check
< 86> ;                          4/7/82  - Add jump table entry for speaker
< 87> ;                                    routine, 1 second delay before "click"
< 88> ;                                    and alpha lock key check
< 89> ;
< 90> ;                          4/8/82  - Release version 1.00
< 91> ;
< 92> ;                          5/5/82  - Add I/O slot configuration check and
< 93> ;                                    I/O slot booting.  Also add change to
< 94> ;                                    Profile read routine for PCR setting.
< 95> ;                          5/12/82 - Add burnin power-cycling routine as
< 96> ;                                    boot option invoked by CMD/P.
< 97> ;                          5/13/82 - Add changes for COPS command timing,
< 98> ;                                    Twiggy timeout, Twiggy booting, and
< 99> ;                                    add power-cycling routine.
<100> ;                          5/14/82 - Add fixes for booting via parameter
<101> ;                                    memory and COPS timing experiment.
<102> ;                          5/17/82 - Add display of loop count and run time,
<103> ;                                    and alter parameter memory useage for
<104> ;                                    power-cycling option.
<105> ;                          5/18/82 - Add display of Twiggy errors, change
<106> ;                                    COPS routine for precheck code.
<107> ;                          5/20/82 - Add contrast reset for "warm start",
<108> ;                                    add cycle value display, restore COPS
<109> ;                                    timeout code.
<110> ;
<111> ;                          5/21/82 - Release version 1.02
<112> ;
<113> ;                          5/26/82 - Begin addition of ROM monitor; set
<114> ;                                    default to Apple if PM = 00.
<115> ;                          6/1/82  - Make following changes:
<116> ;                                     1) Memory sizing retry count to 64
<117> ;                                     2) Save results on memory sizing errors
<118> ;                                     3) Update NMI routine to check for parity
<119> ;                                        errors.
<120> ;                                     4) Restore default NMI vector after
<121> ;                                        memory test.
<122> ;                                     5) Create read clock subroutine and call
<123> ;                                        when doing clock display.
<124> ;                                     6) Add boot fix to save device id.
<125> ;                          6/1/82  - Change to new sizing algorithm and retry
<126> ;                                    count back to 32.
<127> ;                          6/3/82  - Convert to version 1.03
<128> ;                          6/3/82  - Made following changes:
<129> ;                                     1)Localize message display to TSTCHK
<130> ;                                     2)Do clear screen only in INITVCT and
<131> ;                                       in TSTCHK and second monitor level.
<132> ;                                     3)Change default video page to last.
<133> ;                                     4)Complete first edition of monitor.
<134> ;                          6/7/82  - Modify monitor level2 user interface.
<135> ;                          6/10/82 - Made following changes:
<136> ;                                     1)Add boot from Apple as CMD/A.
<137> ;                                     2)Clear screen and display only in
<138> ;                                       routine TSTCHK.
<139> ;                                     3)Add ROM checksum error bit.
<140> ;                                     4)Add exception error check to TSTCHK.
<141> ;                                     5)Add speaker click just before
<142> ;                                       keyboard scan.
<143> ;                                     6)Reset to first video page for boot
<144> ;                                       from Apple.
<145> ;                                     7)Merge in changes from 1.03 file.
<146> ;                                     8)Add parity error check to TSTCHK
<147> ;                                     9)Change power-cycling so that double
```

```
<148> ;                                   bus fault used to restart diags
<149> ;                          6/11/82 - Made following changes:
<150> ;                                    1)Increase Twiggy timeout to 2 mins.
<151> ;                                    2)Add 5 sec delay in power-cycle mode
<152> ;                                      before shutting down.
<153> ;
<154> ;                          6/14/82 - Release version 1.04
<155> ;
<156> ;                          6/22/82 - Add loop after COPS test if error
<157> ;                                    since keyboard not accessible.  Also add
<158> ;                                    fix for NMI restore after memory test.
<159> ;                          6/30/82 - Made following changes:
<160> ;                                    1)Add parameter memory and I/O boot
<161> ;                                      checksum routines.
<162> ;                                    2)Remove boot id save to parameter
<163> ;                                      memory, except for power-cycle.
<164> ;                                    3)Change to new boot device id's.
<165> ;                          7/1/82  -  1)Add changes for new Twiggy firmware.
<166> ;                                    2)Add fixes for bugs in 1.04:
<167> ;                                     a)Add row setting before error display
<168> ;                                       to avoid writing over menu line.
<169> ;                                     b)Set device codes for Profile and
<170> ;                                       I/O slots to allow display if error.
<171> ;                                     c)Enable setting of timeout for Twiggy
<172> ;                                       reads.
<173> ;                                     d)Save error codes for I/O boot in
<174> ;                                       memory.
<175> ;                                     e)Add option of clearing memory in
<176> ;                                       INITMON routine.
<177> ;                          7/7/82  - Made following changes:
<178> ;                                    1)Modify checksum routines
<179> ;                                    2)Add keyboard/mouse check/reset code
<180> ;                          7/13/82 - Add speed parameter for new Twiggy code
<181> ;                          7/14/82 - Add check for DSKDIAG in disk test,
<182> ;                                    change to new Twiggy error codes
<183> ;                          7/15/82 - Made following changes:
<184> ;                                    1)Add Profile routine updates.
<185> ;                                    2)Restore old boot id codes - new ones
<186> ;                                      used only when new Twiggy code
<187> ;                                      released.
<188> ;                                    3)Upgrade burnin code for new parameter
<189> ;                                      memory usage.
<190> ;                                    4)Attempt to enable keyboard after MMU
<191> ;                                      errors.
<192> ;                                    5)Remove I/O boot checksum code until
<193> ;                                      conversion to new Twiggy code.
<194> ;                                    6)Add video pattern display code..
<195> ;                                    7)Remove characters from table and
<196> ;                                      make other changes to save bytes.
<197> ;                                    8)Upgrade service mode display option
<198> ;                                      to handle count up to $FFFF.
<199> ;
<200> ;                          7/16/82 - Create version 1.05
<201> ;                          7/19/82 - Add bug fixes for MMU testing, power-
<202> ;                                    cycle memory testing, Profile boot
<203> ;                                    and service mode display option.
<204> ;
<205> ;                          7/19/82 - Create version 1.06
<206> ;                          7/20/82 - Add fix for MMU testing to properly
<207> ;                                    record context in error
<208> ;
<209> ;                          7/20/82 - Release version 1.07
<210> ;
<211> ;                          7/21/82 - Make keyboard/mouse reset code changes
<212> ;                                    and move check to before first "click"
```

```
<213> ;                         7/23/82 - Add extended memory tests
<214> ;                         7/27/82 - Add screen memory test and VIA tests.
<215> ;                                   Change default boot for new Twiggy code
<216> ;                                   to upper Twiggy.  Add conditionals for
<217> ;                                   Apple code.
<218> ;                         7/29/82 - Add SCC test, optimize code.
<219> ;                         7/30/82 - Add RAM address uniqueness test.
<220> ;                         8/4/82  - Added the following:
<221> ;                                    1)Twiggy mods for interleave
<222> ;                                    2)Monitor options CONTINUE and LOOP
<223> ;                                    3)Exception routine for line 1111 and
<224> ;                                      line 1010 errors.
<225> ;                         8/9/82  - Add Twiggy mod for disk clamp, add mods
<226> ;                                   for kernel test failures such as screen
<227> ;                                   flash on MMU error.
<228> ;                         8/11/82 - Add memory sizing fix, increase delay
<229> ;                                   for COPS and change default boot to
<230> ;                                   TWIGGY!!
<231> ;                         8/12/82 - Begin code changes for new user interface
<232> ;                                   and add hooks for icon display.
<233> ;                         8/14/82 - Add mods for Twiggy changes to monitor
<234> ;                                   DSKDIAG line and add initial timeout.
<235> ;                                   Continue user interface changes.
<236> ;                         8/18/82 - Add mouse, cursor code and changes for
<237> ;                                   Customer mode to use mouse.
<238> ;                         8/23/82 - Add controls for 2716 version of ROM.
<239> ;                                   Add changes for Service mode to use
<240> ;                                   pull down menu, eliminate keyboard
<241> ;                                   queuing while awaiting input.
<242> ;                         8/24/82 - Add dialog box, and window to service
<243> ;                                   mode with modified scroll and character
<244> ;                                   output routines.
<245> ;                         8/25/82 - Add icons along with routines to display
<246> ;                                   during test and for errors.
<247> ;                         8/27/82 - Add routines for displaying and using
<248> ;                                   boot icon menu.
<249> ;                         8/30/82 - Add auto boot from Applenet.
<250> ;                         8/31/82 - Add minor additions to Service mode
<251> ;                                   for Set and Loop options.
<252> ;
<253> ;                         8/31/82 - Create and do internal release of
<254> ;                                   2716 (0.24), 2732 (1.15) and 2764 (2.00)
<255> ;                                   ROM versions.
<256> ;
<257> ;                         9/8/82  - Add fixes for I/O slot icon display
<258> ;                                   and Profile icon display.
<259> ;                         9/9/82  - Add fix for reboot problem in 2716 ROM.
<260> ;                                   Add serial # read routine and test for
<261> ;                                   2732 and 2764 ROM versions.  Expand
<262> ;                                   stack for serial read routine.
<263> ;                         9/10/82 - Add fix for device code display for ROM
<264> ;                                   versions 0.24 and 1.15.
<265> ;
<266> ;                         9/10/82 - Create and do internal release of new
<267> ;                                   ROM versions 0.25, 1.16 and 2.01.
<268> ;
<269> ;                         9/13/82 - Add fixes for memory sizing and I/O
<270> ;                                   slot booting.
<271> ;
<272> ;                         9/14/82 - Create and release ROM versions 0.26,
<273> ;                                   1.17 and 2.02.
<274> ;
<275> ;                         9/22/82 - Add fixes and code for:
<276> ;                                    1)Default video latch setting
<277> ;                                    2)Mask for I/O and exception errors
```

```
<278> ;                                          3)Message display on external calls
<279> ;                                            to ROM monitor
<280> ;                                          4)Contrast setting before screen test
<281> ;                                          5)Disable of NMI key on power-up
<282> ;                                          6)Boot failure after first load
<283> ;                                          7)Error tones for failures
<284> ;                                          8)Loop mode setting of NMI key
<285> ;                                  9/23/82 - Add
<286> ;                                           1)Power cycling
<287> ;                                           2)Full service mode menu
<288> ;                                           3)Loop mode test choice display
<289> ;                                  9/24/82 - Add dump memory option to service mode
<290> ;                                  9/25/82 - Modify display memory option to allow
<291> ;                                           count and address data on same line
<292> ;                                  9/29/82 - Add jump table entry for READMMU
<293> ;                                  9/30/82 - Add:
<294> ;                                           1)"No reset" feature
<295> ;                                           2)Verify Disk option for service mode
<296> ;                                           3)Optimize cursor routines and
<297> ;                                             remove unused CursorShield routine.
<298> ;                                           4)Invert rectangles when selected from
<299> ;                                             keyboard.
<300> ;                                           5)Display boot menu only if down keycode
<301> ;                                             detected.
<302> ;                                  10/5/82 - Add:
<303> ;                                           1)Memory error decoding to board level
<304> ;                                           2)New size and position for alert box
<305> ;                                           3)New test icon display
<306> ;                                           4)Diskette # for Twiggy errors
<307> ;                                  10/6/82 - Add:
<308> ;                                           1)Continue keyboard scan after COPS
<309> ;                                             errors
<310> ;                                           2)Set extended memory test bit for
<311> ;                                             loop on memory test option
<312> ;                                           3)Display I/O slot card # on errors
<313> ;                                           4)Change boot menu to "pull-down" format
<314> ;                                           5)Change to new icons
<315> ;                                  10/7/82 - Add:
<316> ;                                           1)SCC test
<317> ;                                           2)Error if no serial # (allow continue)
<318> ;                                           3)Two passes of memory tests for extended
<319> ;                                             mode, one for regular mode
<320> ;                                  10/9/82 - Create version 2.03
<321> ;                                  10/10/82 - Add bug fixes and I/O slot ROM check in
<322> ;                                           config scan.
<323> ;
<324> ;                                  10/12/82 - Create and release version 2.04.
<325> ;
<326> ;                                  10/13/82 - Make following changes:
<327> ;                                            1)Add keyboard reset code
<328> ;                                            2)Remove SCC test
<329> ;                                            3)Add bug fixes for making alert box
<330> ;                                              and displaying bad keyboard
<331> ;                                  10/14/82 - Add display of check marks for test icons
<332> ;                                  10/18/82 - Add fixes for Monitor entry, Profile boot,
<333> ;                                           looping on diag tests
<334> ;                                  10/20/82 - Add message translations
<335> ;                                  10/21/82 - 1)Adjust alert box and button dimensions
<336> ;                                           2)Add boot from all ports on I/O slots
<337> ;                                           3)Add fix for CMD key detection in monitor
<338> ;                                           4)Change powercycle window to alert box
<339> ;                                           5)Extend verify timeout to 4 minutes
<340> ;                                  10/22/82 - 1)Add keyboard reset on external entry to ROM
<341> ;                                             monitor
<342> ;                                           2)Make Dump Memory routine conditional on
```

```
<343> ;                                  final LISA ROM
<344> ;                          10/25/82 - 1)Change wait for disk error to branch to
<345> ;                                     monitor - CONTINUE option then continues
<346> ;                                     with the same boot device
<347> ;                                   2)Change RETRY phrase to RESTART
<348> ;                          10/27/82 - Made following changes:
<349> ;                                    1)RESET instruction on startup
<350> ;                                    2)Jump table entries for access to memory
<351> ;                                      test and display decimal routines
<352> ;                                    3)Optimize warm start reset check and
<353> ;                                      MMU error loop routines
<354> ;                                    4)Change default video page to $2F for
<355> ;                                      no memory found.
<356> ;                                    5)Rewrite screen memory test. Change main
<357> ;                                      memory test to go from low memory
<358> ;                                      to base of screen memory.
<359> ;                                    6)Move inverse video check to after screen
<360> ;                                      test, doing rewrite only of screen page.
<361> ;                                    7)Add new boot failure code, with hooks to
<362> ;                                      catch booting errors after ROM has
<363> ;                                      released control to boot loader for Twiggy
<364> ;                                      and Profile booting.
<365> ;                          10/29/82 - Add display for uncompressed slot card icons.
<366> ;                                    Modify TONE routine to init PCR reg.
<367> ;                          11/1/82  - Change external entry to monitor interface
<368> ;                                    so that error code displayed on same line as
<369> ;                                    message if no icon displayed
<370> ;                          11/3/82  - Made following changes:
<371> ;                                      1)Move creation of test icon display till
<372> ;                                        after keyboard reset so translation can
<373> ;                                        be done if necessary
<374> ;                                      2)Do cursor, mouse init only once so
<375> ;                                        cursor posn not reset
<376> ;                                      3)Optimize mouse, cursor routines
<377> ;                                      4)Correct COPSCMD routine
<378> ;                                      5)Upgrade check for Profile routine and
<379> ;                                        optimize Profile read code
<380> ;                          11/8/82  - Conditionally add check for keyboard connected
<381> ;                                    routine.
<382> ;                          11/9/82  - Create version 2.07
<383> ;                          11/11/82 - Modify ROM checksum algorithm
<384> ;                          11/12/82 - Add diskette eject on power-off
<385> ;                          11/13/82 - 1)Remove Dump Memory/Verify Disk from Service
<386> ;                                      mode menu
<387> ;                                    2)Add speaker beep and specific read/write
<388> ;                                      loop for memory sizing and lo mem errors
<389> ;                          11/15/82 - 1)Add keyboard/mouse disconnect check
<390> ;                                    2)Remove memory "clear" from sizing test - now
<391> ;                                      done after memory testing
<392> ;                          11/16/82 - 1)Change power-cycle invoking to CMD/SHIFT/P
<393> ;                                      key sequence.
<394> ;                                    2)Change customer monitor mode invoking to
<395> ;                                      CMD/ENTER (on key pad) key sequence.
<396> ;                                    3)Add wait for profile loop in boot menu
<397> ;                                      display routine
<398> ;                                    4)Add timeout to general wait for clock
<399> ;                                      input routine
<400> ;                                    5)Increase delay for poweroff wait loop
<401> ;                                    6)Optimize character display routine
<402> ;                          11/18/82 - 1)Add save of error code to special parameter
<403> ;                                      memory area for use during burnin.
<404> ;                                    2)Add context check for MMU testing
<405> ;                                    3)Create version 2.08 for internal release
<406> ;                          11/19/82 - 1)Change initial position of cursor to center
<407> ;                                      of screen.
```

```
<408> ;
<409> ;                                 11/19/82 -  Release versions 2.08 (internal) and
<410> ;                                             2.09 (for manufacturing)
<411> ;
<412> ;                                 12/15/82 -  Add:
<413> ;                                             1)Setting of VIA PCR reg for later use
<414> ;                                             2)Reset of keyboard before boot
<415> ;                                             3)Fix for slot 3 card check for boot menu
<416> ;                                 12/16/82 -  Add:
<417> ;                                             1)Move Profile cmd buffer to location $304
<418> ;                                             2)Change default boot device to Profile
<419> ;                                             3)Remove support for third boot port on
<420> ;                                               each slot
<421> ;                                             4)Expand id range for test card search
<422> ;                                             5)Don't display Restart button after boot
<423> ;                                               error
<424> ;                                             6)New icons
<425> ;                                 12/18/82 -  Fix memory test bug
<426> ;
<427> ;                                  1/3/83  -  Fix bug in reporting parity circuitry
<428> ;                                             failure.  Change version to 2.10.
<429> ;                                  1/7/83  -  Make following changes:
<430> ;                                             1)Change keyboard sequences for I/O slot
<431> ;                                               booting
<432> ;                                             2)Extend timeout for inital Profile check
<433> ;                                  1/11/83 -  Change SCC test to use max baud rate for
<434> ;                                             loopback test
<435> ;                                  1/12/83 -  Add running of expansion card status routines
<436> ;                                             when configuration check is done
<437> ;                                  1/18/83 -  Add fixes for:
<438> ;                                             1)Continuing after memory error
<439> ;                                             2)Checking for no reset function
<440> ;                                             3)Memory sizing - search entire possible 2 meg
<441> ;                                             4)Read of I/O slot ROM for icon data -
<442> ;                                               ensure odd address for icon count
<443> ;                                             5)Default boot setting when loop on memory
<444> ;                                               test selected
<445> ;                                  1/21/83 -  Add save of disk controller self-test status
<446> ;
<447> ;                                  1/28/83 -  Create and release ROM version 2.11
<448> ;
<449> ;                                  3/15/83 -  Extend Profile timeout for case where drive
<450> ;                                             may be parking head. (bug RM016)
<451> ;                                  4/20/83 -  Add fixes for:
<452> ;                                              1)Memory sizing (bug RM015).
<453> ;                                              2)Garbage sent out serial port (RM014).
<454> ;                                              3)Removed 6504 (bug RM013).
<455> ;                                              4)Never ready Profile (bug RM011).
<456> ;                                             Also do some code optimization in icon
<457> ;                                             routines to make room for fixes. (RM000)
<458> ;                                  4/22/83 -  Do code optimization for setting bus error
<459> ;                                             vector (labeled as RM000).
<460> ;                                             Add changes for following requests:
<461> ;                                              1)Display ROM id's on bootup  (CHG001)
<462> ;                                              2)Loop on address 1Meg-2 if sizing error  (CHG002)
<463> ;                                              3)Turn off contrast before doing poweroff (CHG003)
<464> ;                                              4)Change copyright notice.  (CHG005)
<465> ;                                                Also modify alert msg display routine (CHG005).
<466> ;                                  4/26/83 -  Add loop on CPU diags if no memory or I/O
<467> ;                                             board installed.  Also toggle LED. (CHG004)
<468> ;                                  4/27/83 -  Do only basic memory test on warm-start. (CHG006)
<469> ;                                             Add fix for NMI bug (RM010).
<470> ;                                  5/9/83  -  Made following changes:
<471> ;                                              1)Change ROM id display to rev # (D) (CHG001)
<472> ;                                              2)Change ROM test failure to loop at fixed address
```

```
<473> ;                                          $00FE00C8  (end of jump table)  (CHG007)
<474> ;                                        3)Make correction for screen not cleared when
<475> ;                                          continuing from I/O slot error to boot menu.
<476> ;                                          (CHG008)
<477> ;                                 5/10/83 - Add change to enable display of uncompressed icons
<478> ;                                           upon external entry to ROM Monitor (CHG008).
<479> ;
<480> ;                                 5/12/83 - Create and release rev D of boot ROM.
<481> ;
<482> ;                                 8/8/83  - Add changes for Pepsi system:   (CHG009)
<483> ;                                           1) New icons.
<484> ;                                           2) Display of icons with id #'s.
<485> ;                                 8/9/83  - Add save of disk ROM id in low memory. (CHG010)
<486> ;                                           Add fixes for:
<487> ;                                           1) SCC init for Applebus.  (CHG011)
<488> ;                                           2) Test card boot search.  (CHG012)
<489> ;                                 8/10/83 - Delete inverse video check. (CHG013)
<490> ;                                           Add fix to beep routine.    (CHG014)
<491> ;                                 8/16/83 - Delete memory address and ping pong routines,
<492> ;                                           add routines to decode parity error to
<493> ;                                           chip.       (CHG015)
<494> ;                                 9/1/83  - Add retry for hard disk booting.  (CHG016)
<495> ;                                           Add jump table entry for write to
<496> ;                                           parameter memory routine.   (CHG017)
<497> ;                                 9/2/83  - Add new font, modify display routines. (CHG018)
<498> ;                                           Add wait for hard disk ready when
<499> ;                                           power-cycling.  (CHG019)
<500> ;                                 9/6/83  - Add setting of video latch whenever boot
<501> ;                                           error causes jump to ROM low memory default
<502> ;                                           vectors. (CHG020)
<503> ;                                           Add fix for memory test/initialization
<504> ;                                           bug. (CHG021)
<505> ;                                 9/7/83  - Add read of disk controller ROM self-test
<506> ;                                           results.  (CHG022)
<507> ;                                           Add skip of disk eject on power-off if any
<508> ;                                           disk controller errors occurred.  (CHG023)
<509> ;
<510> ;                                 9/8/83  - Release for testing (rev 3B) with Pepsi systems.
<511> ;
<512> ;                                 10/10/83 - 1)Make Pepsi icon changes.  (CHG024)
<513> ;                                           2)Add fix for proper setting of carry bit
<514> ;                                             on floppy or hard disk boots. (CHG025)
<515> ;                                           3)Add fix for video reset on boot from not ready
<516> ;                                             Profile. (CHG026)
<517> ;                                 10/12/83 - Add change to reset SCC for Applebus before
<518> ;                                           doing memory test.  (CHG027)
<519> ;                                 10/20/83 - Add fix for service mode bus error problem. (CHG028)
<520> ;
<521> ;
<522> ;                                 10/20/83 - Release as rev E for Lisa and Pepsi systems.
<523> ;
<524> ;                                 12/15/83 - 1)Add new code to determine system type. (CHG029)
<525> ;                                           2)Change default boot device for Lisa 2
<526> ;                                              system if no hard disk connected. (CHG030)
<527> ;                                           3)Extend timeout for hard disk ready. (CHG031)
<528> ;                                           4)Add bug fix for wrong icon display on Lisa 2.
<529> ;                                              (CHG032)
<530> ;                                           5)Add bug fix for menu display when mouse or
<531> ;                                             keyboard not connected. (CHG033)
<532> ;                                           6)Remove save of error code in parameter memory.
<533> ;                                             (CHG034)
<534> ;                                 12/16/83 - Release as rev 'X' for testing
<535> ;
<536> ;                                 12/21/83 - Release as official rev 'F' for all systems
<537> ;
```

```
<538> ;                                 1/25/84  - 1)Add code to properly initialize Profile-reset
<539> ;                                            and parity-reset lines for Profile booting
(CHG036)
<540> ;
<541> ;                                 2/7/84   - 1)Extend hard disk default read timeout to 16
<542> ;                                            seconds for Widget systems. (CHG037)
<543> ;                                           2)Add delay after hard disk reset for Widget
<544> ;                                            systems. (CHG038)
<545> ;                                 2/8/84   - Release as rev G for testing
<546> ;
<547> ;                                 2/24/84  - Release as official rev H
<548> ;
```

X.3 BOOT ROM SYSTEM DIAGNOSTIC TESTS

The Boot ROM provided an extensive collection of diagnostic tests whose purpose was to verify that the Lisa's hardware was working correctly. The Lisa 2 Owner's Guide (release 2 dated 1983) had the following to say about these startup tests in Section C: Troubleshooting:

Procedure N: Startup Symptoms and Error Messages

Every time you turn on the Lisa, the system automatically runs a series of internal tests. These tests fall into two categories:

o The KERNEL TESTS, which are designed to catch problems serious enough to interfere with the rest of the sequence. After the kernel tests, the Lisa emits one click.

o The MODULE TESTS, which may result in specific error messages. After the module tests, the Lisa emits a double click.

Errors detected during the tests can result in screen messages, error tones, or both.

The Lisa 2 Owner's Guide (release 3 dated 1984) provided a list of the startup tests. A summary of this list follows:

ROM Checksum
MMU Register Test
Memory Sizing
Preliminary Memory Test
VIA Test
Screen Memory Test
CPU Board Test
I/O Board Test
Memory Test

When the startup tests completed, the Lisa would either boot the Lisa operating system if there were no test problems, or if testing indicated a problem an error message would appear. Error messages consisted of an icon showing the part which failed and a failure number (or the Lisa emitted Hi and Lo frequency tones if the failed test was a critical Lisa component which caused the screen to not display anything). For example, if the Lisa encountered an I/O board error the Lisa would display the I/O board icon and a number in the range 50 to 60 (number 50 signified that the keyboard VIA chip was not functioning correctly).

The startup tests took around one minute to execute. Booting the operating system took around 5 minutes.

The Lisa also displayed the Boot ROM and the floppy disk controller ROM versions in the upper right corner of the screen. For my Lisa 2/10 model these appear as "H/88" where

"H" is the Boot ROM version and "88" is the floppy controller ROM version.

After the diagnostic tests ran the Boot ROM stored a large amount of information into an area of the Lisa's memory. This information was used by the Lisa Operating System and most likely the Lisa Office System. Users with knowledge of the Boot ROM's Monitor could also access this information via the Monitor's memory display facility. Details of this information from the Boot ROM source code follow:

```
<  1> ;  OUTPUTS:  Saves various results and contents of system registers in memory
<  2> ;            for examination by system programs or with the ROM monitor.
<  3> ;
<  4> ;               $180-183 : Power-up status (x0000000 = ok)
<  5> ;               $184-185 : Memory sizing error results
<  6> ;               $186-1A5 : Results of memory read/write tests
<  7> ;               $1A6-1A9 : Parity error memory address (if error during mem test)
<  8> ;               $1AA-1AB : Memory error address latch
<  9> ;               $1AC-1AF : D7 save on exception errors
< 10> ;               $1B0-1B1 : Results of MMU tests (context/data bits)
< 11> ;               $1B2     : Keyboard ID (00 = no ID received)
< 12> ;               $1B3     : Boot device ID
< 13> ;               $1B4-1B9 : Boot failure data
< 14> ;               $1BA-1BF : Clock setting (Ey,dd,dh,hm,ms,st)
< 15> ;               $1C0-1DF : Data reg save area (D0 - D7)
< 16> ;               $1E0-1FF : Address reg save area (A0 - A7, A7 = USP)
< 17> ;               $240-260 : System serial #
< 18> ;               $260-267 : Scratch area
< 19> ;               $268-26B : Suspected (logical) memory error address for parity error
< 20> ;               $26C-26F : Save of data written to suspected error address
< 21> ;               $270-273 : Actual (logical) error address found during search
< 22> ;               $274-277 : Save of data read during parity error search
< 23> ;               $278-27B : (Physical) error address read from parity error address latch
< 24> ;               $27C     : Error row for parity chip failure (0 = first row, 7 = last row)
< 25> ;               $27D     : Error column for parity chip failure (9 or 14)
< 26> ;               $27E-280 : Reserved
< 27> ;               $280-293 : Exception data save area
< 28> ;                           (FC/EXCADR/IR/SR/PC/EXCTYPE/SSP)
< 29> ;                           44 = NMI or other interrupt
< 30> ;                           45 = bus error
< 31> ;                           46 = address error
< 32> ;                           47 = other exception/interrupt
< 33> ;                           48 = illegal instruction error
< 34> ;                           49 = line 1010 or 1111 trap
< 35> ;                           50 = bus error accessing keyboard VIA
< 36> ;                           51 = bus error accessing parallel port VIA
< 37> ;                           57 = bus error accessing disk controller
< 38> ;               $294-297 : Maximum physical memory address + 1
< 39> ;               $298-299 : I/O slot 1 card id (0 = no card present)
< 40> ;               $29A-29B : I/O slot 2 card id
< 41> ;               $29C-29D : I/O slot 3 card id
< 42> ;               $29E     : Reserved
< 43> ;               $29F     : Reserved
< 44> ;               $2A0     : Reserved
< 45> ;               $2A1     : Disk ROM id
< 46> ;               $2A2-2A3 : Reserved
< 47> ;               $2A4-2A7 : Minimum physical address
< 48> ;               $2A8-2AB : Total memory (Max-Min)
< 49> ;               $2AC     : SCC test results
< 50> ;               $2AD     : Slot # of memory board if memory error
< 51> ;               $2AE     : Result of disk controller self-test
< 52> ;               $2AF     : System type (0 = Lisa 1, 1 = Lisa 2, 2 = Lisa 2 with external hard
disk,
< 53> ;                                      3 = Lisa 2 with internal hard disk)
< 54> ;               $2B0-2BF : Keyboard queue (16 bytes)
< 55> ;               $2C0-480 : ROM scratchpad/stack area
```

```
< 56> ;              $480-800 : Reserved for ROM local variable usage
< 57> ;
< 58> ;           Also saves data in special parameter memory area reserved for boot ROM use if error
< 59> ;           encountered.  Usage is as follows:
< 60> ;
< 61> ;              $FCC161     : Error code
< 62> ;              $FCC163-165 : Contents of memory error address latch if parity error
< 63> ;              $FCC167     : Memory board slot # if memory error
< 64> ;              $FCC169-173 : Last value read from clock
< 65> ;              $FCC175-17B : Reserved
< 66> ;              $FCC17D-17F : Checksum
```

The Boot ROM also determined the type of Lisa that the ROM was running on (refer to memory location $2AF in the above listing). The ROM could detect the following Lisa systems:

- Lisa 1
- Lisa 2
- Lisa 2 with an external hard disk (either Apple's 5Mbyte ProFile or Priam's 80Mbyte Data Tower)
- Lisa 2 with an internal hard disk (Apple's 10Mbyte Widget)

The system type was determined by the following ROM routine (note the use of the floppy disk ROM id to determine if the Lisa is a Lisa 1 by checking if a Twiggy drive is present):

```
<  1> ;---------------------------------------------------------------------------
<  2> ;  Subroutine for determining system type
<  3> ;  Returns type value in D0 and sets SYSTYPE location in memory
<  4> ;    D0 = 0 - Lisa 1
<  5> ;         1 - Lisa 2/external disk with slow timers
<  6> ;         2 - Lisa 2/external disk with fast timers
<  7> ;         3 - Lisa 2/internal disk (Pepsi) with fast timers
<  8> ;---------------------------------------------------------------------------
<  9>
< 10> SETTYPE CLR.L   D0              ;clear for type usage                  CHG029
< 11>         MOVE.B  DISKROM,D1      ;read disk id                          CHG029
< 12>         TST.B   D1              ;check for Lisa 1                      CHG029
< 13>         BPL.S   @9              ;skip if yes                           CHG029
< 14>         BTST    #SLOTMR,D1      ;Lisa 2 with slow timers?              CHG029
< 15>         BEQ.S   @1              ;skip if not                           CHG029
< 16>         MOVEQ   #1,D0           ;else set type                         CHG029
< 17>         BRA.S   @9              ;                                      CHG029
< 18> @1      BTST    #FASTMR,D1      ;Lisa 2 with fast timers?              CHG029
< 19>         BEQ.S   @2              ;skip if not                           CHG029
< 20>         MOVEQ   #2,D0           ;else set type                         CHG029
< 21>         BRA.S   @9              ;                                      CHG029
< 22> @2      MOVEQ   #3,D0           ;else must be Pepsi with fast timers   CHG029
< 23> @9      MOVE.B  D0,SYSTYPE      ;save system type                      CHG029
< 24>         RTS                     ;                                      CHG029
```

Note the reference to "Pepsi" which was the Lisa 2's code name by Apple since John Sculley, formerly of Pepsi, was head of Apple at this time.

X.4 BOOT ROM OPERATING SYSTEM BOOTING

XXXX

X.5 BOOT ROM SERVICE MODE

The Lisa's ROM Service Mode is a very esoteric feature of the ROM which seems to be a tool for use by trained Lisa service or manufacturing people. Documentation for this

feature does not seem to exist. Even Apple's Lisa Level 1 Technical Procedures (revision June 1986) does not mention this feature.

An excellent discussion of the Service Mode may be found in Larry Pina's book Macintosh Repair & Upgrade Secrets (1st edition, pages 235-280). Chapter 13, Lisa/Macintosh XL Repair Secrets, has the following to say about the Service Mode (p. 254):

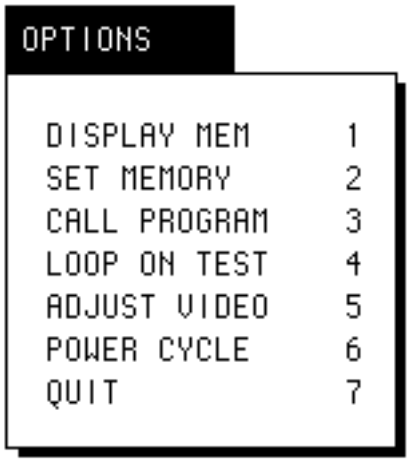
In addition to the automatic startup tests, the Lisa has a built-in service mode. The built-in service mode is top secret. Very few people know it exists. No one I've spoken to, not even the most knowledgable Lisa owners and technical support people, have ever seen the documentation. Whether the documentation was lost, or whether it was ever written, remains a mystery. Still, some tests are easy to figure out. Adjust Video, for example, puts up a 1/2-inch reverse video crosshatch.

Here's how to enter the built-in service mode:

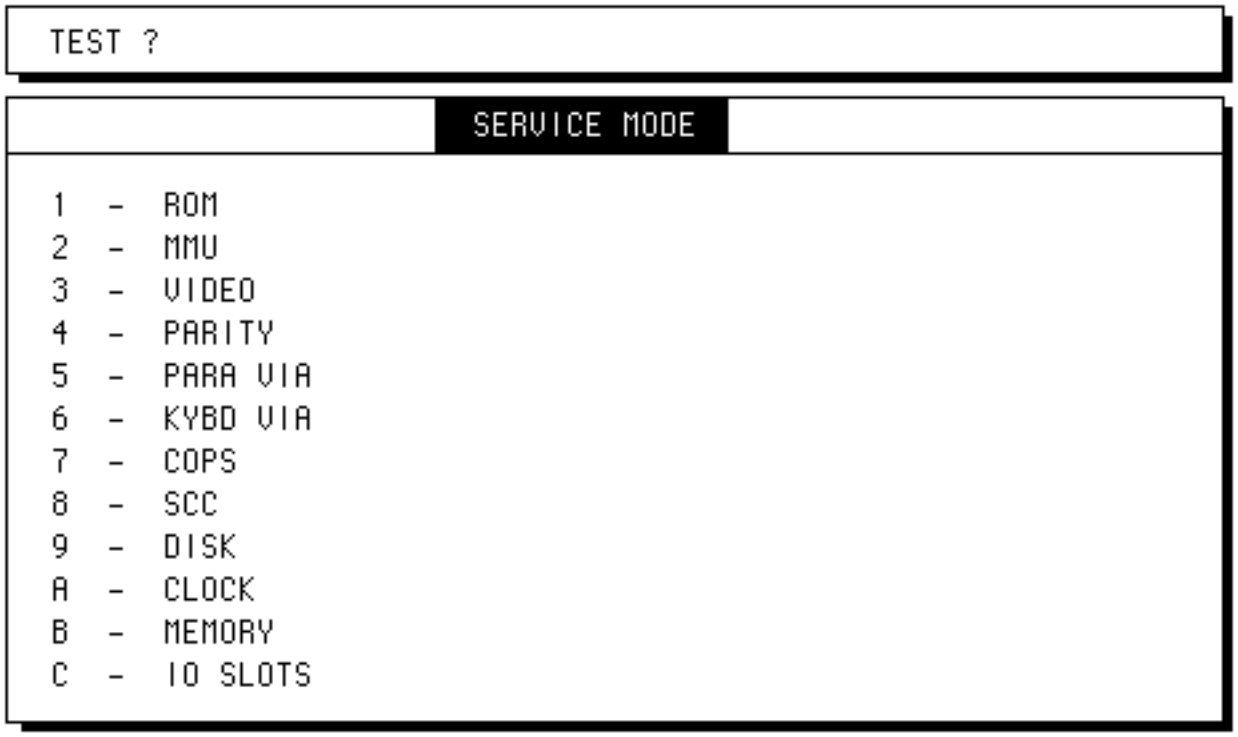
1. If the computer is on, turn it off. Wait a few seconds, then without inserting a startup floppy disk, turn the computer on again.
2. At the end of the kernal test (when you hear the first click), hit any key except Caps Lock. Hitting a key interrupts the normal startup procedure and turns on the Startup From mode.
3. At the end of the module test (when you hear the second click), hold down the Apple key and press the 2 key.
4. Since there's no floppy disk in the disk drive, the Lisa beeps three times and presents you with an error box. Ignore it!
5. Hold down the Apple key and press the "S" key, "S" presumably stands for "Service". Case is irrelevant. Pressing S, or s, will engage the service mode.

Without more information, I can only speculate as to the purpose of these tests. My best guess is that they were used on the assembly line for quality control.

If you follow the steps that Mr. Pina discusses you should see the following pull-down menu in the upper left corner of the screen:



You can select a specific test using either the mouse or by typing a numeric key (e.g. typing "2" will activate the SET MEMORY menu command). You will also see two windows on the right side of the screen:



The above SERVICE MODE window shows an additional set of commands that are activated when you select the LOOP ON TEST command from the OPTIONS menu. These tests allow you to select and run specified tests for selected durations. The window with TEST ? inside it contains other requests depending upon the selected menu command. For example, if select the DISPLAY MEM menu command the top window asks for the starting address of memory to display and the number of bytes to display.

From the Boot ROM source code's perspective, the Service Mode was called Level 2 of the Lisa ROM Monitor. Level 1 was called the customer level.

Additional ROM Monitor support existed in the area of communications with an external computer. There are several comments and routines in the source code which allowed an Apple 2 computer user to boot the Lisa and for sending data between the Lisa and the Apple 2. I assume this feature allowed an Apple 2 program to control the Lisa, at least at the Lisa ROM level.

X.6 BOOT ROM SOURCE CODE METRICS

As part of my interest in software metrics I wrote several Macintosh MPW shell tools which produced a fair amount of metric data concerning the Boot ROM source code. These metrics were aimed at the number of source file lines and the ratio of comments to source lines, and at the source code's opcode usage.

In my opinion the Boot ROM source code was very well written in terms of readability. I found the general comments at the start of the source code to be very helpful in understanding how the ROM worked and how it was organized. The history modification listing was excellent. The routines within the source code were also documented well with each having a heading commentary. The analysis of the source code for the comment ratio showed that on a line basis the source contained around 20% comments.

As an example of the routine header comments, here's a sample from the mouse handling portion of the Boot ROM:

```
<  1> ;--------------------------------------------------------------------------
<  2> ;
<  3> ;   Hardware Interface for the Mouse
<  4> ;
<  5> ;   Written by Rick Meyers
<  6> ;   (c) Apple Computer Incorporated, 1983
<  7> ;
<  8> ;   The routines below provide an assembly language interface to the mouse.
<  9> ;   Input parameters are passed in registers, output parameters are returned
< 10> ;   in registers.  Unless otherwise noted, all registers are preserved.
< 11> ;
< 12> ;   The Mouse
< 13> ;
< 14> ;   The mouse is a pointing device used to indicate screen locations.  Mouse
< 15> ;   coordinates are located between pixels on the screen.  Therefore, the
< 16> ;   X-coordinate can range from 0 to 720, and the Y-coordinate from 0 to 364.
< 17> ;   The initial mouse location is 0,0.
< 18> ;
< 19> ;   Mouse Scaling
< 20> ;
< 21> ;   The relationship between physical mouse movements and logical mouse
< 22> ;   movements is not necessary a fixed linear mapping.  Three alternatives
< 23> ;   are available: 1) unscaled, 2) scaled for fine movement and 3) scaled
< 24> ;   for coarse movement.
< 25> ;
< 26> ;   When mouse movement is unscaled, a horizontal mouse movement of x units
< 27> ;   yields a change in the mouse X-coordinate of x pixels.  Similiarly, a
< 28> ;   vertical movement of y units yields a change is the mouse Y-coordinate
< 29> ;   of y pixels.  These rules apply independent of the speed of the mouse
< 30> ;   movement.
< 31> ;
< 32> ;   When mouse movement is scaled, horizontal movements are magnified by 3/2
< 33> ;   relative to vertical movements.  This is intended to compensate for the
< 34> ;   2/3 aspect ratio of pixels on the screen.  When scaling is in effect, a
< 35> ;   distinction is made between fine (small) movements and coarse (large)
< 36> ;   movements.  Fine movements are slightly reduced, while coarse movements
< 37> ;   are magnified. For scaled fine movements, a horizontal mouse movement of
< 38> ;   x units yields a change in the X-coordinate of x pixels, but a vertical
< 39> ;   movement of y units yields a change of (2/3)*y pixels.  For scaled coarse
< 40> ;   movements, a horizontal movement a x units yields a change of (3/2)*x
< 41> ;   pixels, while a vertical movements of y units yields a change of y pixels.
< 42> ;
< 43> ;   The distinction between fine movements and coarse movements is determined
< 44> ;   by the sum of the x and y movements each time the mouse location is
< 45> ;   updated.  If this sum is at or below the 'threshold', the movement is
< 46> ;   considered to be a fine movement.  Values of the threshold range from 0
< 47> ;   (which yields all coarse movements) to 256 (which yields all fine
< 48> ;   movements).  Given the default mouse updating frequency, a threshold of
< 49> ;   about 8 (threshold's initial setting) gives a comfortable transition between
< 50> ;   fine and coarse movements.
< 51> ;--------------------------------------------------------------------------------
```

The source code for my metric analyzer tools appears at the end of this section.

Boot ROM Lines and Comments Metrics:

```
<  1> ################################################################################
<  2> #                                                                              #
<  3> #  SUMMARY LINE AND COMMENT METRICS FOR APPLE LISA BOOT ROM 2.48 SOURCE CODE  #
<  4> #                                                                              #
<  5> ################################################################################
<  6>
<  7>                       DAVID T CRAIG  -  12 JUNE 1993
```

```
<  8>
<  9> These metrics describe the following source line and character information
< 10> for a single source file:
< 11>
< 12>  Total Lines                = total number of source lines
< 13>  Total Comment Lines        = total number of full comment lines (*)
< 14>  Comment / Line Count %     = ratio of comment lines to source lines
< 15>
< 16>  Total Characters           = total number of source line characters
< 17>  Total Comment Characters   = total number of comment characters (**)
< 18>  Comment / Line Character % = ratio of comment characters to line characters
< 19>
< 20> Notes: ( *) includes only comment lines that exist soley by themselves,
< 21>             not source lines with ending comments.
< 22>        (**) comment character count includes comments that exist at the end
< 23>             of a source line also.
< 24>
< 25> SUMMARY: These metrics show that the amount of comments in the source code
< 26>          files are around 20% on a line basis and around 50% on a character
< 27>          basis.
< 28>
< 29> These metrics were obtained from the Macintosh utility tool DTCGetAsmComments
< 30> as written by David T. Craig in June 1993.
< 31>
< 32> Let the metrics begin ...
< 33>
< 34> FILE: Lisa Boot ROM RM248.S
< 35>
< 36>    Total Lines                =     2032
< 37>    Total Comment Lines        =      426
< 38>    Comment / Line Count %     =       20 %
< 39>
< 40>    Total Characters           =    72687
< 41>    Total Comment Characters   =    39675
< 42>    Comment / Line Character % =       54 %
< 43>
< 44> FILE: Lisa Boot ROM RM248.M
< 45>
< 46>    Total Lines                =     2452
< 47>    Total Comment Lines        =      518
< 48>    Comment / Line Count %     =       21 %
< 49>
< 50>    Total Characters           =    88459
< 51>    Total Comment Characters   =    46911
< 52>    Comment / Line Character % =       53 %
< 53>
< 54> FILE: Lisa Boot ROM RM248.K
< 55>
< 56>    Total Lines                =     1833
< 57>    Total Comment Lines        =      344
< 58>    Comment / Line Count %     =       18 %
< 59>
< 60>    Total Characters           =    66653
< 61>    Total Comment Characters   =    36724
< 62>    Comment / Line Character % =       55 %
< 63>
< 64> FILE: Lisa Boot ROM RM248.G
< 65>
< 66>    Total Lines                =     2165
< 67>    Total Comment Lines        =      625
< 68>    Comment / Line Count %     =       28 %
< 69>
< 70>    Total Characters           =    68960
< 71>    Total Comment Characters   =    33921
< 72>    Comment / Line Character % =       49 %
```

```
< 73>
< 74> FILE: Lisa Boot ROM RM248.E
< 75>
< 76>    Total Lines                =     1838
< 77>    Total Comment Lines        =      742
< 78>    Comment / Line Count %     =       40 %
< 79>
< 80>    Total Characters           =    83586
< 81>    Total Comment Characters   =    51852
< 82>    Comment / Line Character % =       62 %
< 83>
< 84> FILE: Lisa Boot ROM RM248.B
< 85>
< 86>    Total Lines                =     2203
< 87>    Total Comment Lines        =      401
< 88>    Comment / Line Count %     =       18 %
< 89>
< 90>    Total Characters           =    81080
< 91>    Total Comment Characters   =    42170
< 92>    Comment / Line Character % =       52 %
< 93>
< 94> ###############################################################################
< 95> #                                                                             #
< 96> #                    F        I        N        I        S                    #
< 97> #                                                                             #
< 98> ###############################################################################

Boot ROM Opcode Usage Metrics:

<  1> ###############################################################################
<  2> #                                                                             #
<  3> #     APPLE LISA BOOT ROM 2.48 SOURCE CODE OPCODE STATISTICAL INFORMATION     #
<  4> #                                                                             #
<  5> ###############################################################################
<  6>
<  7> Assembly Source File Opcode Information Gatherer
<  8> Version: 1.0  [7/11/93]
<  9>
< 10>  File #   1: Lisa Boot ROM RM248.B.TEXT
< 11>  File #   2: Lisa Boot ROM RM248.E.TEXT
< 12>  File #   3: Lisa Boot ROM RM248.G.TEXT
< 13>  File #   4: Lisa Boot ROM RM248.K.TEXT
< 14>  File #   5: Lisa Boot ROM RM248.M.TEXT
< 15>  File #   6: Lisa Boot ROM RM248.S.TEXT
< 16>
< 17> Opcode List unsorted:
< 18>
< 19>   Minimum opcode frequency = 1
< 20>   Maximum opcode frequency = 804
< 21>
< 22>  #  Opcode          Frequency Histogram
< 23> --- --------------- --------- --------------------------------------------------
< 24>   1           .PAGE       115 ********
< 25>   2           .LIST        10 **
< 26>   3             BSR       479 *******************************
< 27>   4           CLR.L        95 *******
< 28>   5            BTST        86 *******
< 29>   6           BEQ.S       195 *************
< 30>   7          MOVE.B       212 **************
< 31>   8             .IF       253 *****************
< 32>   9           CMP.B        96 *******
< 33>  10           BNE.S       210 **************
< 34>  11           .ENDC       253 *****************
< 35>  12           BRA.S       251 *****************
```

```
< 36>  13            BNE       19 ***
< 37>  14            BCS       14 **
< 38>  15           .ELSE      78 ******
< 39>  16           BCC.S      20 ***
< 40>  17           TST.B      88 *******
< 41>  18           MOVEQ     376 ************************
< 42>  19           LSR.B       5 **
< 43>  20            MOVE     228 ***************
< 44>  21           BPL.S      22 ***
< 45>  22            MULU      21 ***
< 46>  23             BRA      94 *******
< 47>  24           BGT.S      20 ***
< 48>  25          MOVE.L     383 *************************
< 49>  26             BEQ      25 ***
< 50>  27            BCLR      20 ***
< 51>  28            BSET      64 *****
< 52>  29             LEA     179 ************
< 53>  30           BCS.S      59 *****
< 54>  31          CMPI.B      40 ****
< 55>  32           CLR.B      29 ***
< 56>  33           MOVEP      11 **
< 57>  34         MOVEM.L     152 ***********
< 58>  35         MOVEA.L      37 ****
< 59>  36           BSR.S     145 **********
< 60>  37             RTS     176 ************
< 61>  38           LSL.B       1 *
< 62>  39          ANDI.B      26 ***
< 63>  40            OR.B       4 **
< 64>  41             NOT      11 **
< 65>  42            ADDQ      56 *****
< 66>  43             TST      47 ****
< 67>  44          ADDQ.L      14 **
< 68>  45             ADD      39 ****
< 69>  46             ROL       5 **
< 70>  47             DBF      23 ***
< 71>  48           ORI.B      32 ***
< 72>  49          CMPA.L      31 ***
< 73>  50             AND       3 **
< 74>  51             CMP      16 **
< 75>  52             CLR      29 ***
< 76>  53            ANDI      17 **
< 77>  54           BLE.S      12 **
< 78>  55          SUBA.L      19 ***
< 79>  56           ADD.L      26 ***
< 80>  57            SUBQ      74 ******
< 81>  58          ANDI.L      18 ***
< 82>  59           ROR.L       4 **
< 83>  60            OR.L       3 **
< 84>  61             ORI       7 **
< 85>  62             JMP      23 ***
< 86>  63           BSRS4      20 ***
< 87>  64            RTS4      17 **
< 88>  65           MOVEA      55 *****
< 89>  66         DISABLE       4 **
< 90>  67         MOVEP.L      12 **
< 91>  68          SUBQ.B       2 **
< 92>  69          MOVE.W      69 ******
< 93>  70           ADD.W      26 ***
< 94>  71          ENABLE       8 **
< 95>  72          SUBQ.L      15 **
< 96>  73         .NOLIST       8 **
< 97>  74             BSR4      41 ****
< 98>  75           TST.W       7 **
< 99>  76           TST.L      33 ***
<100>  77            CMPI       9 **
```

```
<101>  78          BHI.S        3 **
<102>  79            JSR        4 **
<103>  80          ROL.L       17 **
<104>  81           SWAP       13 **
<105>  82         ADDQ.B        7 **
<106>  83          BGE.S       15 **
<107>  84            NOP       11 **
<108>  85         .MACRO       11 **
<109>  86          .ENDM       11 **
<110>  87           .EQU      804 **************************************************
<111>  88       .INCLUDE        5 **
<112>  89          CMP.W        7 **
<113>  90          BLT.S       13 **
<114>  91          NEG.L        1 *
<115>  92          ADDI.L       1 *
<116>  93          EOR.W        2 **
<117>  94          CMP.L        7 **
<118>  95           ADDA       14 **
<119>  96         SUBQ.W        3 **
<120>  97          SUB.L       11 **
<121>  98         ADDI.W        2 **
<122>  99          LSR.L       19 ***
<123> 100         ANDI.W        2 **
<124> 101           DIVU        4 **
<125> 102             OR       13 **
<126> 103         ADDA.L       29 ***
<127> 104          LSR.W        3 **
<128> 105          EOR.L        4 **
<129> 106            SUB        6 **
<130> 107          SUB.B        4 **
<131> 108          BMI.S       24 ***
<132> 109            LSL        6 **
<133> 110          .BYTE      306 ********************
<134> 111         .ALIGN        1 *
<135> 112          .WORD       62 *****
<136> 113         .ASCII       76 ******
<137> 114           .ORG        6 **
<138> 115           .END        1 *
<139> 116      .ABSOLUTE        1 *
<140> 117          .PROC        1 *
<141> 118            RTE        2 **
<142> 119          RESET        3 **
<143> 120           BSR6       11 **
<144> 121           ROXL        2 **
<145> 122           RTS6        5 **
<146> 123            EOR        6 **
<147> 124          BSRS6        5 **
<148> 125            ROR        2 **
<149> 126           BSR2        9 **
<150> 127           RTS2        1 *
<151> 128            BGT        1 *
<152> 129          ADD.B        1 *
<153> 130           SUBI        1 *
<154> 131          SUB.W        9 **
<155> 132            BHI        1 *
<156> 133            BMI        1 *
<157> 134          NOT.B        8 **
<158> 135          BLS.S        4 **
<159> 136          LSL.L        7 **
<160> 137         SUBI.B        3 **
<161> 138         ADDI.B        1 *
<162> 139           BLO.S       1 *
<163> 140          MOVEM        7 **
<164> 141           BCHG        1 *
<165> 142            LSR        4 **
```

```
<166> 143           EXT.W         2 **
<167> 144           NEG.W         5 **
<168> 145           ASR.W         2 **
<169> 146          ADDQ.W         2 **
<170> 147           AND.W         2 **
<171> 148           AND.L         6 **
<172> 149           NOT.L         6 **
<173> 150           ORI.W         1 *
<174> 151            LINK         1 *
<175> 152            DBLE         2 **
<176> 153            DBRA         5 **
<177> 154           CLR.W         6 **
<178> 155          CMPI.W         1 *
<179> 156          SUBI.W         1 *
<180> 157            UNLK         1 *
<181> 158           LSL.W         3 **
<182> 159          ROXL.B         2 **
<183> 160          ROXL.L         2 **
<184> 161           EOR.B         1 *
<185> 162          CMPI.L         2 **
<186> --- --------------- --------- --------------------------------------------------
<187>
<188> Opcode List sorted by NAME:
<189>
<190>   Minimum opcode frequency = 1
<191>   Maximum opcode frequency = 804
<192>
<193>  #  Opcode          Frequency Histogram
<194> --- --------------- --------- --------------------------------------------------
<195>   1       .ABSOLUTE         1 *
<196>   2          .ALIGN         1 *
<197>   3          .ASCII        76 ******
<198>   4           .BYTE       306 ********************
<199>   5           .ELSE        78 ******
<200>   6            .END         1 *
<201>   7           .ENDC       253 *****************
<202>   8           .ENDM        11 **
<203>   9            .EQU       804 **************************************************
<204>  10             .IF       253 *****************
<205>  11        .INCLUDE         5 **
<206>  12           .LIST        10 **
<207>  13          .MACRO        11 **
<208>  14         .NOLIST         8 **
<209>  15            .ORG         6 **
<210>  16           .PAGE       115 ********
<211>  17           .PROC         1 *
<212>  18           .WORD        62 *****
<213>  19             ADD        39 ****
<214>  20           ADD.B         1 *
<215>  21           ADD.L        26 ***
<216>  22           ADD.W        26 ***
<217>  23            ADDA        14 **
<218>  24          ADDA.L        29 ***
<219>  25          ADDI.B         1 *
<220>  26          ADDI.L         1 *
<221>  27          ADDI.W         2 **
<222>  28            ADDQ        56 *****
<223>  29          ADDQ.B         7 **
<224>  30          ADDQ.L        14 **
<225>  31          ADDQ.W         2 **
<226>  32             AND         3 **
<227>  33           AND.L         6 **
<228>  34           AND.W         2 **
<229>  35            ANDI        17 **
<230>  36          ANDI.B        26 ***
```

```
<231>  37          ANDI.L       18 ***
<232>  38          ANDI.W        2 **
<233>  39           ASR.W        2 **
<234>  40           BCC.S       20 ***
<235>  41            BCHG        1 *
<236>  42            BCLR       20 ***
<237>  43             BCS       14 **
<238>  44           BCS.S       59 *****
<239>  45             BEQ       25 ***
<240>  46           BEQ.S      195 *************
<241>  47           BGE.S       15 **
<242>  48             BGT        1 *
<243>  49           BGT.S       20 ***
<244>  50             BHI        1 *
<245>  51           BHI.S        3 **
<246>  52           BLE.S       12 **
<247>  53           BLO.S        1 *
<248>  54           BLS.S        4 **
<249>  55           BLT.S       13 **
<250>  56             BMI        1 *
<251>  57           BMI.S       24 ***
<252>  58             BNE       19 ***
<253>  59           BNE.S      210 **************
<254>  60           BPL.S       22 ***
<255>  61             BRA       94 *******
<256>  62           BRA.S      251 ******************
<257>  63            BSET       64 *****
<258>  64             BSR      479 ********************************
<259>  65           BSR.S      145 **********
<260>  66            BSR2        9 **
<261>  67            BSR4       41 ****
<262>  68            BSR6       11 **
<263>  69           BSRS4       20 ***
<264>  70           BSRS6        5 **
<265>  71            BTST       86 *******
<266>  72             CLR       29 ***
<267>  73           CLR.B       29 ***
<268>  74           CLR.L       95 *******
<269>  75           CLR.W        6 **
<270>  76             CMP       16 **
<271>  77           CMP.B       96 *******
<272>  78           CMP.L        7 **
<273>  79           CMP.W        7 **
<274>  80          CMPA.L       31 ***
<275>  81            CMPI        9 **
<276>  82          CMPI.B       40 ****
<277>  83          CMPI.L        2 **
<278>  84          CMPI.W        1 *
<279>  85             DBF       23 ***
<280>  86            DBLE        2 **
<281>  87            DBRA        5 **
<282>  88         DISABLE        4 **
<283>  89            DIVU        4 **
<284>  90          ENABLE        8 **
<285>  91             EOR        6 **
<286>  92           EOR.B        1 *
<287>  93           EOR.L        4 **
<288>  94           EOR.W        2 **
<289>  95           EXT.W        2 **
<290>  96             JMP       23 ***
<291>  97             JSR        4 **
<292>  98             LEA      179 ************
<293>  99            LINK        1 *
<294> 100             LSL        6 **
<295> 101           LSL.B        1 *
```

```
<296> 102           LSL.L         7 **
<297> 103           LSL.W         3 **
<298> 104             LSR         4 **
<299> 105           LSR.B         5 **
<300> 106           LSR.L        19 ***
<301> 107           LSR.W         3 **
<302> 108            MOVE       228 ***************
<303> 109          MOVE.B       212 **************
<304> 110          MOVE.L       383 *************************
<305> 111          MOVE.W        69 ******
<306> 112           MOVEA        55 *****
<307> 113         MOVEA.L        37 ****
<308> 114           MOVEM         7 **
<309> 115         MOVEM.L       152 ***********
<310> 116           MOVEP        11 **
<311> 117         MOVEP.L        12 **
<312> 118           MOVEQ       376 ************************
<313> 119            MULU        21 ***
<314> 120           NEG.L         1 *
<315> 121           NEG.W         5 **
<316> 122             NOP        11 **
<317> 123             NOT        11 **
<318> 124           NOT.B         8 **
<319> 125           NOT.L         6 **
<320> 126              OR        13 **
<321> 127            OR.B         4 **
<322> 128            OR.L         3 **
<323> 129             ORI         7 **
<324> 130           ORI.B        32 ***
<325> 131           ORI.W         1 *
<326> 132           RESET         3 **
<327> 133             ROL         5 **
<328> 134           ROL.L        17 **
<329> 135             ROR         2 **
<330> 136           ROR.L         4 **
<331> 137            ROXL         2 **
<332> 138          ROXL.B         2 **
<333> 139          ROXL.L         2 **
<334> 140             RTE         2 **
<335> 141             RTS       176 ************
<336> 142            RTS2         1 *
<337> 143            RTS4        17 **
<338> 144            RTS6         5 **
<339> 145             SUB         6 **
<340> 146           SUB.B         4 **
<341> 147           SUB.L        11 **
<342> 148           SUB.W         9 **
<343> 149          SUBA.L        19 ***
<344> 150            SUBI         1 *
<345> 151          SUBI.B         3 **
<346> 152          SUBI.W         1 *
<347> 153            SUBQ        74 ******
<348> 154          SUBQ.B         2 **
<349> 155          SUBQ.L        15 **
<350> 156          SUBQ.W         3 **
<351> 157            SWAP        13 **
<352> 158             TST        47 ****
<353> 159           TST.B        88 *******
<354> 160           TST.L        33 ***
<355> 161           TST.W         7 **
<356> 162            UNLK         1 *
<357> --- --------------- --------- --------------------------------------------------
<358>
<359> Opcode List sorted by FREQUENCY:
<360>
```

```
<361>   Minimum opcode frequency = 1
<362>   Maximum opcode frequency = 804
<363>
<364>  #  Opcode          Frequency Histogram
<365> --- --------------- --------- --------------------------------------------------
<366>   1             .EQU       804 **************************************************
<367>   2              BSR       479 ******************************
<368>   3          MOVE.L       383 ************************
<369>   4           MOVEQ       376 ************************
<370>   5           .BYTE       306 ********************
<371>   6           .ENDC       253 ****************
<372>   7             .IF       253 ****************
<373>   8           BRA.S       251 ****************
<374>   9            MOVE       228 **************
<375>  10          MOVE.B       212 *************
<376>  11           BNE.S       210 *************
<377>  12           BEQ.S       195 ************
<378>  13             LEA       179 ***********
<379>  14             RTS       176 ***********
<380>  15         MOVEM.L       152 **********
<381>  16           BSR.S       145 *********
<382>  17           .PAGE       115 ********
<383>  18           CMP.B        96 *******
<384>  19           CLR.L        95 *******
<385>  20             BRA        94 *******
<386>  21           TST.B        88 *******
<387>  22            BTST        86 *******
<388>  23           .ELSE        78 ******
<389>  24          .ASCII        76 ******
<390>  25             SUBQ       74 ******
<391>  26          MOVE.W        69 ******
<392>  27            BSET        64 *****
<393>  28           .WORD        62 *****
<394>  29           BCS.S        59 *****
<395>  30            ADDQ        56 *****
<396>  31           MOVEA        55 *****
<397>  32             TST        47 ****
<398>  33            BSR4        41 ****
<399>  34          CMPI.B        40 ****
<400>  35             ADD        39 ****
<401>  36         MOVEA.L        37 ****
<402>  37           TST.L        33 ***
<403>  38           ORI.B        32 ***
<404>  39          CMPA.L        31 ***
<405>  40          ADDA.L        29 ***
<406>  41             CLR        29 ***
<407>  42           CLR.B        29 ***
<408>  43           ADD.L        26 ***
<409>  44           ADD.W        26 ***
<410>  45          ANDI.B        26 ***
<411>  46             BEQ        25 ***
<412>  47           BMI.S        24 ***
<413>  48             DBF        23 ***
<414>  49             JMP        23 ***
<415>  50           BPL.S        22 ***
<416>  51            MULU        21 ***
<417>  52           BCC.S        20 ***
<418>  53            BCLR        20 ***
<419>  54           BGT.S        20 ***
<420>  55           BSRS4        20 ***
<421>  56             BNE        19 ***
<422>  57           LSR.L        19 ***
<423>  58          SUBA.L        19 ***
<424>  59          ANDI.L        18 ***
<425>  60            ANDI        17 **
```

```
<426>  61          ROL.L       17 **
<427>  62          RTS4        17 **
<428>  63           CMP        16 **
<429>  64          BGE.S       15 **
<430>  65          SUBQ.L      15 **
<431>  66          ADDA        14 **
<432>  67          ADDQ.L      14 **
<433>  68           BCS        14 **
<434>  69          BLT.S       13 **
<435>  70            OR        13 **
<436>  71          SWAP        13 **
<437>  72          BLE.S       12 **
<438>  73         MOVEP.L      12 **
<439>  74          .ENDM       11 **
<440>  75         .MACRO       11 **
<441>  76          BSR6        11 **
<442>  77         MOVEP        11 **
<443>  78           NOP        11 **
<444>  79           NOT        11 **
<445>  80          SUB.L       11 **
<446>  81          .LIST       10 **
<447>  82          BSR2         9 **
<448>  83          CMPI         9 **
<449>  84          SUB.W        9 **
<450>  85        .NOLIST        8 **
<451>  86         ENABLE        8 **
<452>  87          NOT.B        8 **
<453>  88         ADDQ.B        7 **
<454>  89          CMP.L        7 **
<455>  90          CMP.W        7 **
<456>  91          LSL.L        7 **
<457>  92          MOVEM        7 **
<458>  93            ORI        7 **
<459>  94          TST.W        7 **
<460>  95           .ORG        6 **
<461>  96          AND.L        6 **
<462>  97          CLR.W        6 **
<463>  98           EOR         6 **
<464>  99           LSL         6 **
<465> 100          NOT.L        6 **
<466> 101           SUB         6 **
<467> 102       .INCLUDE        5 **
<468> 103          BSRS6        5 **
<469> 104           DBRA        5 **
<470> 105          LSR.B        5 **
<471> 106          NEG.W        5 **
<472> 107           ROL         5 **
<473> 108          RTS6         5 **
<474> 109          BLS.S        4 **
<475> 110        DISABLE        4 **
<476> 111           DIVU        4 **
<477> 112          EOR.L        4 **
<478> 113           JSR         4 **
<479> 114           LSR         4 **
<480> 115          OR.B         4 **
<481> 116          ROR.L        4 **
<482> 117          SUB.B        4 **
<483> 118           AND         3 **
<484> 119          BHI.S        3 **
<485> 120          LSL.W        3 **
<486> 121          LSR.W        3 **
<487> 122          OR.L         3 **
<488> 123         RESET         3 **
<489> 124         SUBI.B        3 **
<490> 125         SUBQ.W        3 **
```

```
<491> 126          ADDI.W        2 **
<492> 127          ADDQ.W        2 **
<493> 128           AND.W        2 **
<494> 129          ANDI.W        2 **
<495> 130           ASR.W        2 **
<496> 131          CMPI.L        2 **
<497> 132            DBLE        2 **
<498> 133           EOR.W        2 **
<499> 134           EXT.W        2 **
<500> 135             ROR        2 **
<501> 136            ROXL        2 **
<502> 137          ROXL.B        2 **
<503> 138          ROXL.L        2 **
<504> 139             RTE        2 **
<505> 140          SUBQ.B        2 **
<506> 141       .ABSOLUTE        1 *
<507> 142          .ALIGN        1 *
<508> 143            .END        1 *
<509> 144           .PROC        1 *
<510> 145           ADD.B        1 *
<511> 146          ADDI.B        1 *
<512> 147          ADDI.L        1 *
<513> 148            BCHG        1 *
<514> 149             BGT        1 *
<515> 150             BHI        1 *
<516> 151           BLO.S        1 *
<517> 152             BMI        1 *
<518> 153          CMPI.W        1 *
<519> 154           EOR.B        1 *
<520> 155            LINK        1 *
<521> 156           LSL.B        1 *
<522> 157           NEG.L        1 *
<523> 158           ORI.W        1 *
<524> 159            RTS2        1 *
<525> 160            SUBI        1 *
<526> 161          SUBI.W        1 *
<527> 162            UNLK        1 *
<528> --- --------------- --------- --------------------------------------------------
<529>
<530> FINIS
```

Boot ROM Source Line Label Metrics:

```
<  1>
<  2> Assembly Source File Label Information Gatherer     Version: 1.0
<  3> Written by David T. Craig  [9/5/94 6:06:51 PM]
<  4> 736 Edgewater, Wichita, Kansas 67230
<  5> Copyright (c) 1993 by David T. Craig
<  6>
<  7> Current Date and Time:  Monday, September 5, 1994  6:09:28 PM
<  8>
<  9>
< 10> ********************************************************************************
< 11> * LABEL INFO FOR FILE : RM248.E.TEXT
< 12> ********************************************************************************
< 13>    750 DIAGS                 751 NEWLISA               752 BURNIN
< 14>    753 NORESET               754 EXTERNAL              756 ROM16K
< 15>    758 NEWTWIG               760 FINLISA               761 FINKBD
< 16>    762 AAPL                  763 USERINT               764 DEBUG
< 17>    765 ROM4K                 766 ROM8K                 767 BMENU
< 18>    769 FULLSCC               770 INVERTCK              786 ROMBASE
< 19>    787 ROMSLCT               788 IOSPACE               789 VIDLTCH
< 20>    790 DEFVID                792 DEFVID2               795 SCRNBASE
< 21>    797 SCRNBASE              801 RBYTES                803 RBYTES
```

```
< 22>   805 TOPOFFSET      806 RLONGS        807 R0
< 23>   808 R1             809 R2            810 R3
< 24>   811 R4             812 R5            813 R6
< 25>   814 R7             815 BUSVCTR       816 ADRVCTR
< 26>   817 ILLVCTR        818 L10VCTR       819 L11VCTR
< 27>   820 NMIVCT         821 TRPVCT0       822 MAXADR
< 28>   823 ONEMEG         824 HALFMEG       825 QTRMEG
< 29>   826 ROW2ADR        827 STKBASE       828 CALLBASE
< 30>   829 SETUP          830 SETUPON       831 PATRN
< 31>   832 PATRN2         833 PARON         834 PAROFF
< 32>   835 MEALTCH        836 STATREG       837 SFER
< 33>   838 PBIT           839 VRBIT         840 VIDBIT
< 34>   841 CSBIT          842 INVIDBIT      843 RETRYCNT
< 35>   844 VTIRDIS        845 VTIRENB       846 HEX512K
< 36>   847 HEX128K        848 HEX96K        849 HEX32K
< 37>   850 HEX8K          851 HEX2K         852 LOMEM
< 38>   853 DG2ON          854 DG2OFF        855 ONESEC
< 39>   856 TWOSEC         857 FIVESEC       858 QTRSEC
< 40>   859 TNTHSEC        860 KBDDLY        861 HALFSEC
< 41>   865 MSRCHSZ        866 VSRCHSZ       867 VMSK
< 42>   868 ADRMSK         869 PHYTOLOG      877 VIA1BASE
< 43>   878 ORB1           879 ORA1          880 DDRB1
< 44>   881 DDRA1          882 T1LL1         883 T1LH1
< 45>   884 T2CL1          885 T2CH1         886 SHR1
< 46>   887 ACR1           888 PCR1          889 IFR1
< 47>   890 IER1           891 PORTA1        893 FDIR
< 48>   895 VIA2BASE       896 ORB2          897 IRB2
< 49>   898 ORA2           899 IRA2          900 DDRB2
< 50>   901 DDRA2          902 T1LL2         903 T1LH2
< 51>   904 T2CL2          905 T2CH2         906 PCR2
< 52>   907 PORTA2         909 DSKDIAG       911 CSTRB
< 53>   915 PIABASE        916 INDATA        917 OUTDATA
< 54>   918 INCSR          919 OUTCSR        923 SCCBCTL
< 55>   924 ACTL           925 SCCDATA       926 RXBF
< 56>   927 TXBE           933 CPUSEL        934 MMU
< 57>   935 VID            936 PAR           937 CPUINTR
< 58>   938 BUSEXCP        939 ADREXCP       940 MISEXCP
< 59>   941 ILLEXCP        942 TRPEXCP       944 VIA1
< 60>   945 VIA2           946 IOCOPS        947 KBDCOPS
< 61>   948 CLK            949 RS232A        950 RS232B
< 62>   951 DISK           952 IOEXCP        953 IOCOPS2
< 63>   955 MEM            956 MPAR          957 IOKBD
< 64>   959 KBDOUT         960 MOUSOUT       961 IO1ERR
< 65>   962 IO2ERR         963 IO3ERR        964 ALTBOOT
< 66>   965 WRMSTRT        967 LOOP          969 ERRMSK
< 67>   970 CPUMSK         971 EXMSK         972 MEMMSK
< 68>   973 IOMSK          974 OTHRMSK       975 CONTMSK
< 69>   977 MMU            978 CPUSEL        979 VID
< 70>   980 PAR            981 CPUINTR       982 BUSEXCP
< 71>   983 ADREXCP        984 MISEXCP       985 ILLEXCP
< 72>   986 TRPEXCP        988 VIA1          989 VIA2
< 73>   990 IOCOPS         991 KBDCOPS       992 CLK
< 74>   993 RS232A         994 RS232B        995 DISK
< 75>   996 IOEXCP         997 IOCOPS2       998 IOKBD
< 76>  1000 MEM           1001 MPAR         1003 KBDOUT
< 77>  1004 MOUSOUT       1005 IO1ERR       1006 IO2ERR
< 78>  1007 IO3ERR        1009 ALTBOOT      1010 BTMENU
< 79>  1011 WRMSTRT       1013 LOOP         1015 ERRMSK
< 80>  1016 CPUMSK        1017 EXMSK        1018 IOMSK
< 81>  1019 MEMMSK        1020 OTHRMSK      1021 IOSMSK
< 82>  1022 CONTMSK       1025 SCANMSK      1026 ALTBMSK
< 83>  1027 BOOTMSK       1028 CPIOMSK      1034 ECPUSEL
< 84>  1035 EMMU          1036 EVID         1037 ECPAR
< 85>  1038 ECPUINTR      1039 EBUSEXCP     1040 EADREXCP
< 86>  1041 EMISEXCP      1042 EILLEXCP     1043 ETRPEXCP
```

```
< 87>  1045 EVIA1          1046 EVIA2          1047 EIOCOP
< 88>  1048 EKBDCOP        1049 ECLK           1050 ERS232A
< 89>  1051 ERS232B        1052 EDISK          1053 EIOEXCP
< 90>  1054 EIOCOP2        1056 EMEM           1057 EPAR
< 91>  1058 EIOKBD         1062 SERR1          1063 SERR2
< 92>  1066 EMMU           1067 ECPUSEL        1068 EVID
< 93>  1069 ECPAR          1070 ECPUINTR       1071 EBUSEXCP
< 94>  1072 EADREXCP       1073 EMISEXCP       1074 EILLEXCP
< 95>  1075 ETRPEXCP       1077 EVIA1          1078 EVIA2
< 96>  1079 EIOCOP         1080 EKBDCOP        1081 ECLK
< 97>  1082 ERS232A        1083 ERS232B        1084 EDISK
< 98>  1085 EIOEXCP        1086 EIOCOP2        1087 EIOKBD
< 99>  1089 EMEM           1090 EPAR           1092 EBOOT
<100>  1096 SERR1          1097 SERR2          1103 NORSTRT
<101>  1104 NOCONT         1105 MSBUTN         1106 CMDFLG
<102>  1107 MOUSE          1108 CHKCMD         1110 BTN
<103>  1111 MENU           1115 MMUSADRL       1116 MMUSADRB
<104>  1117 MMUEADRL       1118 MMUEADRB       1119 ADR128K
<105>  1120 PAG128K        1121 MEMLMT         1122 NMEMLMT
<106>  1125 IOLMT          1126 NIOLMT         1127 IOLMT2
<107>  1128 RSTLMT         1130 IOLMT          1131 NIOLMT
<108>  1133 SPLMT          1134 NSPLMT         1135 INVPAG
<109>  1136 MMU0B          1137 MMU0L          1138 MMU126B
<110>  1139 MMU126L        1140 MMU127B        1141 MMU127L
<111>  1142 SEG1ON         1143 SEG1OFF        1144 SEG2ON
<112>  1145 SEG2OFF        1149 Dlycnst        1150 TKiller
<113>  1151 BytesPerRead   1152 WordsPerRead   1153 HalfSize
<114>  1154 ScrachSize     1156 Snum           1158 dLcnt
<115>  1159 dSavArry       1160 dScrach        1162 dStack
<116>  1166 MOUSDWN        1167 CMDKEY         1168 ALPHKEY
<117>  1171 AKEY           1172 MKEY           1173 FKEY
<118>  1174 GKEY           1175 HKEY           1176 PKEY
<119>  1177 KEY1           1178 KEY2           1179 KEY3
<120>  1181 KEY1           1182 KEY2           1183 KEY3
<121>  1184 AKEY           1185 BKEY           1186 CKEY
<122>  1190 ENTRKEY        1191 SHFTKEY        1192 PKEY
<123>  1195 RSTCODE        1196 KUNPLG         1197 ICERR
<124>  1198 KCERR          1199 MSUNPLG        1200 MSPLG
<125>  1210 TWIG1          1211 TWIG2          1212 PROFILE
<126>  1213 IO1PORT1       1214 IO2PORT1       1215 IO3PORT1
<127>  1216 PC             1217 MON            1218 APPLE
<128>  1220 TWIG1          1221 TWIG2          1222 PROFILE
<129>  1223 IO1PORT1       1224 IO1PORT2       1225 IO2PORT1
<130>  1226 IO2PORT2       1227 IO3PORT1       1228 IO3PORT2
<131>  1229 PC             1230 MON            1235 PC
<132>  1236 APPLE          1242 IOS1           1243 IOS2
<133>  1244 IOS3           1245 TWG1           1246 TWG2
<134>  1247 PRO            1249 TWG1           1250 TWG2
<135>  1251 PRO            1252 IOS1           1253 IOS2
<136>  1254 IOS3           1261 TWIGGY         1262 DISKMEM
<137>  1263 CMD            1264 DRV            1265 SIDE
<138>  1266 SCTR           1267 TRAK           1270 STAT
<139>  1271 CHKCNT         1272 HDR            1273 ROMV
<140>  1274 INTSTAT        1275 DSKSTAT        1276 DISKROM
<141>  1277 HDRSTRT        1278 DSKBFR         1280 SEEK
<142>  1281 READS          1282 CLAMP          1283 STATS
<143>  1284 UNCLAMP        1285 FORMAT         1286 VERIFY
<144>  1289 SPEED          1290 CNFRM          1291 STAT
<145>  1292 INTLV          1293 TYPE           1294 STST
<146>  1295 ROMV           1296 RTRYCNT        1297 INTSTAT
<147>  1298 CHKCNT         1299 CHKCNT2        1300 DSKBUFF
<148>  1301 DSKDATA        1302 DISKROM        1303 SLOTMR
<149>  1304 FASTMR         1306 READS          1307 WRT
<150>  1308 UNCLAMP        1309 FMT            1310 VFY
<151>  1311 CLAMP          1312 OK             1314 SEEK
```

```
<152>  1317 EXRW        1318 CLRSTAT    1319 ENBLINT
<153>  1320 DSABLINT    1321 SLEEP      1322 DIE
<154>  1324 DRV1        1325 DRV2       1326 TRK1
<155>  1327 TOPSIDE     1328 BOTSIDE    1331 HDRLEN
<156>  1332 SECLEN      1333 TWGHDR     1334 TWGDATA
<157>  1337 HDRLEN      1338 SECLEN     1339 TWGHDR
<158>  1340 TWGDATA     1341 LASTBLK    1342 DSKSIZE
<159>  1347 STATSTRT    1348 STATSAV    1349 STATSUM
<160>  1350 STATWRDS    1352 PMSTRT     1353 DVCCODE
<161>  1354 MEMCODE     1355 MOUSEON    1356 EXMEM
<162>  1357 PMCHKSM     1358 PMWRDS     1363 TIMOUT
<163>  1364 WRPERR      1365 DRVERR     1366 FMTERR
<164>  1367 RDWRERR     1369 CMDTIME    1370 FDIRTIME
<165>  1373 DRVERR      1374 NODISK     1375 WRPERR
<166>  1376 CLMPERR     1377 RDWRERR    1378 UCLMPERR
<167>  1379 BADTHDR     1380 TIMOUT     1382 CMDTIME
<168>  1383 FDIRTIME    1384 VFYTIME    1385 EJCTTIME
<169>  1386 DSKTMOUT    1387 INSRTTIM   1388 FMTTIME
<170>  1393 DSK1IN      1394 BUTN1      1395 RWF1
<171>  1396 DSK2IN      1397 BUTN2      1398 RWF2
<172>  1402 DSKIN       1403 BUTN       1405 DRVTYPE
<173>  1413 PROFLE      1414 OCD        1415 BSY
<174>  1416 CMDBUFR     1417 STATBFR    1418 STAT1
<175>  1419 STAT2       1420 STAT3      1421 STAT4
<176>  1422 STATMSK     1423 PCMDSZ     1424 PCMD
<177>  1425 BLKH        1426 BLKM       1427 BLKL
<178>  1428 RETRY       1429 THRESH     1430 HDRBUFR
<179>  1431 FILEID      1432 BOOTPAT    1433 DATABFR
<180>  1434 HDRSIZE     1435 BLKSIZE    1436 STRTIME
<181>  1437 RSTRTIME    1438 RDTIME     1439 BSYTIME
<182>  1440 RSPTIME     1441 RCNT       1442 TCNT
<183>  1447 TMOUT       1448 NODSK      1449 DSKBSY
<184>  1450 BADRSP      1451 STATNZ     1452 BADHDR
<185>  1454 NODSK       1455 DSKBSY     1456 BADRSP
<186>  1457 STATNZ      1458 BADHDR     1459 TMOUT
<187>  1467 SLOT1L      1468 SLOT2L     1469 SLOT3L
<188>  1470 STBIT       1471 ICBIT      1472 TSTBIT
<189>  1473 STENTRY     1474 BTENTRY    1475 ICONPTR
<190>  1476 APPLENET    1477 APPLQUAL   1478 TSTCRD
<191>  1479 TSTQUAL     1484 NOC        1485 INV
<192>  1486 BADSM       1487 BADST      1489 NOC
<193>  1490 INV         1491 BADSM      1492 BADST
<194>  1502 INITFLG     1503 HOURSAV    1504 LCNTHI
<195>  1505 LCNTLO      1506 TIMFLG     1507 MINSAV
<196>  1508 DSKCNTH     1509 DSKCNTL    1510 CLKSAVE
<197>  1511 ALRMSAV     1512 CYCLCNT    1513 CYCLVAL
<198>  1514 MINCNT      1515 ENDPM      1516 SET1
<199>  1517 SET2        1518 HOUR       1519 MINUTE
<200>  1520 ONEHOUR     1521 ONEMIN     1522 TENSECS
<201>  1523 DLYTIME     1534 QUESTN     1535 RET
<202>  1536 BS          1540 KEY4       1541 KEY5
<203>  1542 KEY6        1543 KEY7       1544 KEY8
<204>  1545 KEY9        1546 SKEY       1547 CmdDwn
<205>  1548 CmdUp       1549 MousUp     1553 KBDBFR
<206>  1554 KBDEND      1557 CRTROW     1558 CRTCOL
<207>  1560 CRTROW      1561 CRTCOL     1564 MAXTEST
<208>  1567 FIRSTROW    1568 FIRSTCOL   1569 LASTROW
<209>  1570 LASTCOL     1575 ROWBYTES   1576 MaxX
<210>  1577 MaxY        1578 MENULINE   1579 DESKLINE
<211>  1580 DESKLMT     1581 DESKPATRN  1583 WROW
<212>  1584 WCOL        1585 WINDWIDTH  1586 WINDHIGH
<213>  1587 WMIDROW     1588 WMIDCOL    1589 W14COL
<214>  1590 W34COL      1591 WINDSTRT   1593 ALBOXROW
<215>  1594 ALBOXCOL    1595 ALRTWIDTH  1596 ALRTHIGH
<216>  1597 ALRTSTRT    1598 MIDALROW   1599 MIDALCOL
```

```
<217>   1601 BTNWIDTH            1602 BTNHIGH            1603 BTNSPC
<218>   1604 BTNMSPC            1605 BTNROW             1606 BTNCOL
<219>   1607 BTN1STRT           1608 BTN2STRT           1609 BTN3STRT
<220>   1610 BTN1MSG            1611 BTN2MSG            1612 BTN3MSG
<221>   1614 MENUSTRT           1615 MENULEN            1616 MENUSPC
<222>   1617 MENUWIDTH          1618 MENULOC            1619 MENU1MSG
<223>   1620 MBARLEN            1622 MITEMS             1623 MENUEND
<224>   1626 BMENUWIDTH         1627 BMENULEN           1628 BMENUSPC
<225>   1631 DBOXWIDTH          1632 DBOXHIGH           1633 DBOXTOP
<226>   1634 DBOXLEFT           1635 DBOXSTRT           1636 DBOXROW
<227>   1637 DBOXCOL            1639 SVCTOP             1641 SVCLEFT
<228>   1642 SVCSTRT            1643 SVCWIDTH           1644 SVCHIGH
<229>   1646 FIRSTROW           1647 FIRSTCOL           1648 ROWSLEFT
<230>   1649 CHARROWS           1650 LASTROW            1651 LASTCOL
<231>   1652 ROWLINES           1653 ROWLEN             1654 NROWS
<232>   1655 CHRHIGH            1656 CHRWIDTH           1657 CHRSPC
<233>   1659 ICONWIDTH          1660 ICONHIGH           1662 TSTROW
<234>   1663 TSTCOL             1664 TSTWSTRT           1665 TSTWWIDTH
<235>   1666 TSTWHIGH           1667 TSTMROW            1668 TSTMCOL
<236>   1669 MIDTSTROW          1670 CHKROW             1671 TSTIROW
<237>   1672 TSTICOL            1673 TSTISPC            1675 CPUSTRT
<238>   1676 MEMSTRT            1677 IOSTRT             1678 XCRDSTRT
<239>   1680 ERRROW             1681 ERRCOL             1682 ERRSTRT
<240>   1683 ALRTROW            1684 ALRTCOL            1685 CODEROW
<241>   1686 CODECOL            1687 MSGROW             1688 MSGCOL
<242>   1689 MEMROW             1690 MEMCOL             1691 DISKROW
<243>   1692 DISKCOL            1693 SLOTROW            1694 SLOTCOL
<244>   1695 DRVROW             1696 DRVCOL             1697 INSRTROW
<245>   1698 INSRTCOL           1701 DEFROW             1702 DEFCOL
<246>   1703 DEFSTRT            1704 ALTCOL             1705 COL1STRT
<247>   1706 COL2STRT           1707 COL2MID            1708 COL3STRT
<248>   1709 ICONCSPC           1710 ICONMSPC           1711 ICONRSPC
<249>   1713 ALTKYADDR          1715 PCWIDTH            1716 PCHIGH
<250>   1717 PCSTRT             1718 PCROW              1719 PCCOL
<251>   1721 ROMIDROW           1722 ROMIDCOL           1725 GLOBALS
<252>   1727 GLOBALS            1730 ClockBytes         1731 MousX
<253>   1732 MousY              1733 MousDx             1734 MousDy
<254>   1735 MousScaling        1736 MousThresh         1738 CrsrHotx
<255>   1739 CrsrHoty           1740 CrsrHeight         1741 CrsrX
<256>   1742 CrsrY              1743 CrsrTracking       1744 CrsrBusy
<257>   1745 CrsrVisible        1746 CrsrHidden         1747 CrsrObscured
<258>   1749 SavedData          1750 SavedX             1751 SavedY
<259>   1752 SavedRows          1753 SavedAddr          1755 LwrRight
<260>   1756 MsgLen             1759 IconBase           1760 IconAddr
<261>   1762 MenuBase           1763 IconAddr           1766 IconCnt
<262>   1768 DRIVE              1769 BLKNUM             1770 CONTXT
<263>   1772 RectCnt            1773 RectTable          1781 STATUS
<264>   1782 SIZRSLT            1783 MEMRSLT            1784 BOOTMEM
<265>   1785 PEADDR             1786 ADRLTCH            1787 D7SAV
<266>   1788 MMURSLT            1789 KEYID              1790 BOOTDVCE
<267>   1791 BOOTDATA           1792 CLKDATA            1793 DATARGS
<268>   1794 ADRREGS            1795 A6SAV              1796 USPSAV
<269>   1798 SERNUM             1799 KBDQPTR            1801 XPCTADDR
<270>   1802 XPCTDATA           1803 ACTADDR            1804 ACTDATA
<271>   1805 PEADR2             1806 PCHPROW            1807 PCHIP
<272>   1809 EXCFC              1810 EXCADR             1811 EXCIR
<273>   1812 EXCSR              1813 EXCPC              1814 EXCTYPE
<274>   1815 SUPSTK             1816 MAXMEM             1817 IO1ID
<275>   1818 IO2ID              1819 IO3ID              1820 IO1STAT
<276>   1821 IO2STAT            1822 IO3STAT            1823 IOROM
<277>   1824 STATFLGS           1825 MINMEM             1826 TOTLMEM
<278>   1827 SCCRSLT            1828 MEMSLOT            1829 DSKRSLT
<279>   1830 SYSTYPE            1831 KBDQ               1832 QEND
<280> ********************************************************************************
<281> * FILE LABEL STATISTICS
```

```
<282> ********************************************************************************
<283> File Lines           =     1838
<284> Label Count          =      801
<285> Local Label Count    =        6
<286> Min     Label Length =        2 R0
<287> Max     Label Length =       12 BytesPerRead
<288> Avg     Label Length =        6
<289> Std Dev Label Length =     1.61
<290>
<291> Label Length Frequency Counts:
<292>   No. Labels with Length   2:       13
<293>   No. Labels with Length   3:       29
<294>   No. Labels with Length   4:      108
<295>   No. Labels with Length   5:      113
<296>   No. Labels with Length   6:      174
<297>   No. Labels with Length   7:      231
<298>   No. Labels with Length   8:      107
<299>   No. Labels with Length   9:       14
<300>   No. Labels with Length  10:        6
<301>   No. Labels with Length  11:        2
<302>   No. Labels with Length  12:        4
<303> ********************************************************************************
<304> * END OF FILE: RM248.E.TEXT
<305> ********************************************************************************
<306>
<307>
<308> ********************************************************************************
<309> * LABEL INFO FOR FILE : RM248.K.TEXT
<310> ********************************************************************************
<311>     13 BASE                  22 BUSVCT                24 ADRVCT
<312>     26 ILLVCT                28 DIV0VCT               30 CHKVCT
<313>     32 TRAPVCT               34 PRIVCT                36 TRCVCT
<314>     38 L10VCT                40 L11VCT                48 EXCPERR
<315>     58 SAVEREGS              60 SAVEREG2              70 SVCMSG
<316>     77 SPURVCT               79 LVL1VCT               81 LVL2VCT
<317>     83 LVL3VCT               85 LVL4VCT               87 LVL5VCT
<318>     89 LVL6VCT               91 LVL7VCT              104 JMPTBL
<319>    155 SPIN                 167 NMIEXCP              181 BEGIN
<320>    216 BEGIN2               232 BEGIN3               246 ROMTST
<321>    252 DOSUM                276 MMUTST               295 MMUERR
<322>    302 MMULP                314 TSTLOOP              315 REGTST
<323>    331 MMUINIT              344 MMURW                348 RWCHK1
<324>    351 RWCHK2               353 RWCHK3               359 CHKBASE
<325>    374 MMUACHK              379 ACHK1                384 MMUSET
<326>    391 ACHK2                403 MADRERR              416 CHKRW
<327>    425 RWERR                436 SETMMU               445 LOADORG
<328>    465 LOADLMT              494 MMUTST2              548 MMUERR2
<329>    549 MMUERR3              554 MMULPCHK             565 CONCHK
<330>    579 CONOK                616 INITMMU              623 RWLOOP
<331>    629 CHKBASE              645 START                659 MEMSIZ
<332>    670 CHKLO                706 SAVELO               718 TSTHI
<333>    733 CHKHI                744 SAVEHI               748 WRAPXIT
<334>    750 SIZXIT               760 CHKMEM               803 LOTONE
<335>    827 RSTMMU               846 REMAP                856 MAPINV
<336>    882 WRTMMU               907 READMMU              947 MEMTST1
<337>    967 TONEDLY              980 CLRMEM               988 INITMEM
<338>   1013 INITVCT             1018 SETVCTRS            1047 SETBUSVCT
<339>   1056 MISC                1061 IERR                1066 NMI
<340>   1080 NOTPE               1083 TRPERR              1088 BERR
<341>   1093 AERR                1097 EXCP0               1102 EXCP1
<342>   1114 SCCSET              1125 VIA2TST             1127 VIA2CHK
<343>   1156 VIA2VCT             1169 VIATST              1182 VIARW
<344>   1203 VIAFAIL             1205 VIARWEND            1213 CONSET
<345>   1216 CONOFF              1218 CONSET2             1239 SCRNTST
<346>   1269 SCRNERR             1271 SCRNOK              1286 SCRNSAV
```

```
<347>   1305 SETVLTCH              1322 INVTST               1343 VIA1TST
<348>   1356 VIA1CHK              1375 DSPCPURM             1388 COPSENBL
<349>   1401 COPSBAD              1410 COPSVCT              1411 IOVCT
<350>   1418 CPSINIT              1426 TURNON               1452 COPSCMD
<351>   1539 RSTSCAN              1556 RST0                 1563 GET0
<352>   1569 RST2                 1579 RST1                 1604 GETJMP
<353>   1609 RSTXIT               1622 MSCHK                1638 SCANERR
<354>   1641 IOCERR               1651 SCANXIT              1673 GETDATA
<355>   1685 GETIT                1700 RSTKBD               1717 CLRRST
<356>   1727 DELAY_1              1730 DELAY5               1733 KBDDELAY
<357>   1737 DELAY                1746 BEEP                 1754 CLICK
<358>   1767 TONE                 1774 TONE2                1800 SILENCE
<359>   1810 NOIO                 1821 NOIO2                1828 NOIO3
<360> ********************************************************************************
<361> * FILE LABEL STATISTICS
<362> ********************************************************************************
<363> File Lines           =     1833
<364> Label Count          =      147
<365> Local Label Count    =       50
<366> Min     Label Length =        3 NMI
<367> Max     Label Length =        9 SETBUSVCT
<368> Avg     Label Length =        6
<369> Std Dev Label Length =     1.11
<370>
<371> Label Length Frequency Counts:
<372>   No. Labels with Length   3:        1
<373>   No. Labels with Length   4:       13
<374>   No. Labels with Length   5:       26
<375>   No. Labels with Length   6:       47
<376>   No. Labels with Length   7:       50
<377>   No. Labels with Length   8:        9
<378>   No. Labels with Length   9:        1
<379> ********************************************************************************
<380> * END OF FILE: RM248.K.TEXT
<381> ********************************************************************************
<382>
<383>
<384> ********************************************************************************
<385> * LABEL INFO FOR FILE : RM248.S.TEXT
<386> ********************************************************************************
<387>     15 VIDTST                 23 VIDCHK                 47 VIDERR
<388>     54 VIDXIT                 95 RDSERN                122 GetBits1:
<389>    152 GetBits2:             179 GetBytes:             213 CheckSum:
<390>    250 Exit:                 263 FindSync:             288 GetNibbles:
<391>    306 Tag                   324 PARTST                372 PARERR
<392>    381 PARXIT                388 WWPERR                395 VIA1VCT
<393>    414 MEMTST2               423 MEMLOOP               457 TSTDONE
<394>    472 RUNTESTS              476 BASICTST              478 CALL3
<395>    493 BASICTST              495 CALL3                 502 TSTDONE
<396>    517 TSTINIT               530 SAVRSLT               565 RAMTEST
<397>    571 RAMRW                 576 RAMCHK2               584 RAMNXT
<398>    592 ADRTST                598 ADRCHK                602 ADRCLR
<399>    614 RDERR                 633 PRTYINT1              671 TSTSTAT
<400>    685 PRTYINT2              718 PRIXIT                723 PCERR
<401>    740 GETPADDR              752 IOTST                 792 SCCTEST
<402>    799 VECTLOOP              822 b96data:              831 b96lth
<403>    833 SETSCC                840 LPTEST                842 SCCLOOP
<404>    848 SCCOUT                851 SCCLOOP2              857 SCCIN
<405>    865 SCCLXIT               867 SCCLERR               871 SCCEXIT
<406>    903 WRITESCC              914 INITBDATA             917 INITBLTH
<407>    919 INITB2                920 INITB2L               922 RSTSCC
<408>    939 SCCVCT                959 DSKTST               1018 INTERR
<409>   1023 DSKXIT               1033 DSKVCT               1053 SETTYPE
<410>   1074 COPSCHK              1089 SCANCPS              1095 KEYSCAN
<411>   1165 XLATE                1178 KEYTBL               1186 TBLEND
```

```
<412>   1193 CLKTST               1206 READCLK              1212 RDCLK0
<413>   1224 RDCLK1               1233 CLKERR               1246 CONFIG
<414>   1252 CONFIG2              1275 RDSLOTS              1291 NOCRD1
<415>   1293 SLOT2                1302 NOCRD2               1304 SLOT3
<416>   1313 NOCRD3               1317 CFGEXIT              1328 CHKID
<417>   1348 TSTCHK               1354 TST2                 1434 EXCHK
<418>   1486 IOCHK                1576 ERRDISP              1589 KBDCHK
<419>   1610 MEMCHK               1635 CHKMADR              1644 SCNRSLTS
<420>   1671 MERRCHK              1681 MEMERR               1697 IOSCHK
<421>   1731 TSTXIT               1739 TSTXIT2              1768 GOTOMON
<422>   1777 PMVCT                1801 OTHER                1844 DSPCODE
<423>   1867 DSPDEC               1869 GETDIG               1901 DSPCXIT
<424>   1908 OUTCHR               1926 OUTCH                1944 OUTNIB
<425>   1951 ALPHA                1954 DSPCH                1964 NOTIFY
<426>   1977 HIPTCH               1984 LOPTCH               1985 SETDUR
<427>   1997 SYSOK                2010 NOTOK                2022 DSPROMS
<428> ********************************************************************************
<429> * FILE LABEL STATISTICS
<430> ********************************************************************************
<431> File Lines           =     2032
<432> Label Count          =      123
<433> Local Label Count    =      100
<434> Min     Label Length =        3 Tag
<435> Max     Label Length =       11 GetNibbles:
<436> Avg     Label Length =        6
<437> Std Dev Label Length =     1.18
<438>
<439> Label Length Frequency Counts:
<440>   No. Labels with Length   3:        1
<441>   No. Labels with Length   4:        1
<442>   No. Labels with Length   5:       21
<443>   No. Labels with Length   6:       49
<444>   No. Labels with Length   7:       32
<445>   No. Labels with Length   8:       12
<446>   No. Labels with Length   9:        6
<447>   No. Labels with Length  10:        0
<448>   No. Labels with Length  11:        1
<449> ********************************************************************************
<450> * END OF FILE: RM248.S.TEXT
<451> ********************************************************************************
<452>
<453>
<454> ********************************************************************************
<455> * LABEL INFO FOR FILE : RM248.B.TEXT
<456> ********************************************************************************
<457>      7 DOBOOT                  9 BOOTCHK                80 DVCECHK
<458>    173 PMEXIT                185 PMERR                 193 LSTCHK
<459>    214 CHKPM                 227 SAV2PM                256 WRTSUM
<460>    278 VFYCHKSM              281 CKLOOP                293 CKXIT
<461>    300 EXPAND                321 EXPAND                345 SEARCH
<462>    364 BOOTMENU              426 ICONCHK               464 SCNSLTS
<463>    473 CHKS2                 480 CHKS3                 491 WT4BOOT
<464>    511 CHKPROFILE            562 DSPMNTRY              585 ICONMENU
<465>    668 CHKSLOT               686 CHKICONS              706 CHKSXIT
<466>    719 RDSLT                 749 TWGBOOT               784 CLRINT
<467>    792 DOREAD                807 RDRETRY               824 RDSCTR1
<468>    841 STRTBOOT              854 DSKTIMERR             857 DSKCHK
<469>    878 DSKBAD                885 DSKOUT                887 DSKDIS
<470>    889 DSKERR                922 TBOOTERR              933 DSKERR2
<471>    935 DSKERR3               938 SAVEXCP               944 BTERR
<472>    963 DSABLDSK              981 CHKDRIVE             1020 TWGRD
<473>   1023 TWGREAD              1050 XFRHDR               1057 XFRDATA
<474>   1065 XFRHDR               1074 XFRDATA              1090 TWGOUT
<475>   1092 TWGERR               1096 TWGOK                1099 TWGRXIT
<476>   1112 CMDCHK               1135 CHKFIN               1156 EJCTDSK
```

```
<477>   1174 CLRFDIR              1190 CHKFDIR              1202 CLAMPIT
<478>   1223 WAITALRT             1224 DSPWTICON            1246 VCTRINIT
<479>   1272 PROBOOT              1297 PBOOT                1306 HDSKERR
<480>   1337 BOOTFAIL             1340 BFAIL2               1349 HDERR2
<481>   1350 HDERR3               1374 PROREAD              1385 CHKBSY
<482>   1395 TRYRD                1422 RDDATA               1433 PROERR
<483>   1439 PROXIT               1442 PROXIT2              1457 PROINIT
<484>   1480 READIT               1501 STRTRD               1516 GETSTAT
<485>   1522 STRTXIT              1533 STAT01               1547 COPY6
<486>   1550 COPY6LP              1556 STATERR              1558 STATXIT
<487>   1571 FINDD2               1578 GETRSP               1585 RSPOK
<488>   1587 SNDR1                1594 FINDERR              1601 FNDXIT
<489>   1614 WFBSY                1616 WFBSY1               1629 SENDRSP
<490>   1644 WFNBSY               1647 WFNBSY2              1650 WFNBSY3
<491>   1652 WFNBSY1              1663 DOCRES               1690 IOSBOOT
<492>   1717 STATOK               1724 NOCRD                1729 INVID
<493>   1732 BADBRD               1734 SENDMSG              1778 RDIOSLT
<494>   1789 LOADPGM              1820 INVSUM               1822 SAVERR
<495>   1825 RDIOXIT              1841 CHKPASS              1843 CHKPAS2
<496>   1846 CLRPM                1877 CHKTIM               1905 TWGCHK
<497>   1939 TSTERR               1950 DISINT               1957 WRTMSG
<498>   1965 CNTINC               1969 DSPTIM               1984 NOCHG
<499>   2005 SHUTDOWN             2013 DSCONT               2046 SELF
<500>   2049 SETERR1              2052 SETERR2              2054 DSPERR
<501>   2068 CMDERR               2079 TODSET               2087 SETXIT
<502>   2100 TWGTST               2107 TWGLOOP              2122 TOOLONG
<503>   2123 TERR                 2131 DSPCLK               2170 TWGDSP
<504>   2194 DSPDVC
<505> ********************************************************************************
<506> * FILE LABEL STATISTICS
<507> ********************************************************************************
<508> File Lines           =     2203
<509> Label Count          =      142
<510> Local Label Count    =       86
<511> Min     Label Length =        4 SELF
<512> Max     Label Length =       10 CHKPROFILE
<513> Avg     Label Length =        6
<514> Std Dev Label Length =     0.98
<515>
<516> Label Length Frequency Counts:
<517>   No. Labels with Length   4:        2
<518>   No. Labels with Length   5:       19
<519>   No. Labels with Length   6:       58
<520>   No. Labels with Length   7:       47
<521>   No. Labels with Length   8:       13
<522>   No. Labels with Length   9:        2
<523>   No. Labels with Length  10:        1
<524> ********************************************************************************
<525> * END OF FILE: RM248.B.TEXT
<526> ********************************************************************************
<527>
<528>
<529> ********************************************************************************
<530> * LABEL INFO FOR FILE : RM248.M.TEXT
<531> ********************************************************************************
<532>      7 INITMON                14 INIT1                 22 INIT2
<533>     42 INIT3                  71 MONITOR               87 LEVEL1
<534>    110 OTHRBTNS              113 DOMENU               139 GETL1
<535>    180 DORESET               185 CONTCHK              244 GETL1XIT
<536>    249 LEV1LOOP              255 GETERR               268 CLRSCRN
<537>    269 WRTSCRN               282 CLRBOX               289 CLRIT
<538>    299 WRTMENU               314 DRWLINE              320 DRWIT
<539>    330 WRTBOX1               351 ReadKey              362 ReadKey
<540>    374 SQUAWK                386 KeyToAscii           405 LEVEL2
<541>    431 DSPMENU               446 GETLEV2              455 DSPMENU
```

```
<542>    500 WRTMENU               549 DSPMENUBOX            579 MAKESVCW
<543>    595 DSPMEM                620 RDCNT                 670 SETMEM
<544>    694 RDDTA                 740 CALLRTN               773 LOOPTST
<545>    864 MMUTSTE1              869 NOSCCTST              875 MEMTST3
<546>    882 LOOPTBL               907 VIDAJST               944 DRWHORZ
<547>    961 DRWVERT               988 PowerCycle            999 INVALID
<548>   1010 LEV2LOOP             1012 INVXIT               1022 NOTAVAIL
<549>   1039 SETCUR               1051 CLRCUR               1069 SCROLL
<550>   1087 PUTLF                1099 PUTBS                1109 PROMPT
<551>   1130 RDINPUT              1135 READIN               1175 PROMPT
<552>   1192 RDINPUT              1197 READIN               1234 SCROLL
<553>   1268 PUTLF                1281 PUTBS                1289 CLRIT
<554>   1300 ENQKBD               1313 GETCH                1325 GETA
<555>   1342 GETPARM              1345 READQ                1361 OKCH
<556>   1368 GETEXIT              1371 INVPARM              1373 GETXIT2
<557>   1380 CONVERT              1396 DBOXDSPLY            1410 CLRDBOX
<558>   1429 GETINPUT             1433 GET1                 1435 CHKIT
<559>   1479 GET2                 1481 WAIT2                1483 CHKIT2
<560>   1516 GET3                 1517 WAIT3                1542 WT4INPUT
<561>   1546 COPS0                1555 COPS1                1565 COPS2
<562>   1608 COPS4                1629 ReadCOPS             1644 PowerOff
<563>   1673 ENBLDRVS             1713 CHKPOSN              1719 GETNTRY
<564>   1763 CHKPXIT              1792 CHKINPUT             1796 RDENTRY
<565>   1823 INVERT               1940 MouseMovement        1948 Scale
<566>   1960 Fine                 1972 Coarse               1981 Bounds
<567>   2010 MousInit             2060 CursorInit           2091 CursorHide
<568>   2127 CursorDisplay        2232 NMISET               2238 RSTOR
<569>   2310 CursorInit           2331 NMISET               2337 CMDDONE
<570>   2341 DEPOSIT              2343 DEMLOOP              2357 READ
<571>   2367 WRITE                2379 RDMULTI              2390 WRMULTI
<572>   2400 CHKHDR               2435 CHKSUM               2444 ECHOSUM
<573> ********************************************************************************
<574> * FILE LABEL STATISTICS
<575> ********************************************************************************
<576> File Lines           =     2452
<577> Label Count          =      123
<578> Local Label Count    =      122
<579> Min     Label Length =        4 GETA
<580> Max     Label Length =       13 MouseMovement
<581> Avg     Label Length =        6
<582> Std Dev Label Length =     1.60
<583>
<584> Label Length Frequency Counts:
<585>   No. Labels with Length   4:        7
<586>   No. Labels with Length   5:       25
<587>   No. Labels with Length   6:       26
<588>   No. Labels with Length   7:       41
<589>   No. Labels with Length   8:       15
<590>   No. Labels with Length   9:        1
<591>   No. Labels with Length  10:        6
<592>   No. Labels with Length  11:        0
<593>   No. Labels with Length  12:        0
<594>   No. Labels with Length  13:        2
<595> ********************************************************************************
<596> * END OF FILE: RM248.M.TEXT
<597> ********************************************************************************
<598>
<599>
<600> ********************************************************************************
<601> * LABEL INFO FOR FILE : RM248.G.TEXT
<602> ********************************************************************************
<603>     33 DRAWDESK               40 CLRDESK                59 gray
<604>     61 gray1                  92 BLACKEN                95 whiten
<605>    112 CLRMENU               151 PAINT_BOX             153 paintb1
<606>    156 inverse               157 cont                  159 paintb2
```

```
<607>    183 startop              186 movinst              190 exclusive
<608>    193 compare              199 nextline             212 MAKEPCALRT
<609>    223 MAKEALERT            241 MAKETEST             270 MAKEDBOX
<610>    300 MAKEWINDOW           341 MAKEBOX              448 PAINT_V
<611>    450 paintv1              481 PAINTBIT             520 MAKEBUTN
<612>    604 DRAWBUTN             670 DRAWSIDES            711 MAKEMENU
<613>    773 WRITETITLE           827 GETROWCOL            857 GETLENGTH
<614>    877 DSPRGICON            900 OUTPUT               904 loop0
<615>    908 loop1                910 loop2                935 DSPNUMICON
<616>    957 DSPNUM              1027 DSPERRICON          1034 DSPBAD
<617>   1054 MRGICON             1087 DSPALRTICON         1115 DSPQICON
<618>   1139 INVICON             1158 DSPCPU              1163 DSPMBRD
<619>   1168 DSPIOB              1173 DSPXCRD             1177 DODSPLY
<620>   1194 CHKCPU              1199 CHKMBRD             1204 CHKIOBRD
<621>   1209 CHKXCRD             1213 DSPCHECK            1231 DSPICON
<622>   1238 DLOOP               1241 MLOOP               1246 BLACK
<623>   1248 CHECK               1253 DONE                1265 XLOOP
<624>   1291 DSPSTRING           1333 DSPALL              1346 DSPGERMN
<625>   1350 DSPFRNCH            1355 DSPOUT              1371 DSPIT
<626>   1394 DSPMSLSH            1408 DSPALRTMSG          1424 CONVRTD5
<627>   1436 DSPMSGR             1460 DSPMSG              1473 DSPDONE
<628>   1487 SETCRSR             1489 SETCRSR2            1532 DSPVAL
<629>   1581 OUTPUT              1616 out                 1624 DSPVXIT
<630>   1635 SPACE               1638 FONTTBL             1685 QUESTCH
<631>   1689 INVCHAR             1693 APPLICON            1702 AsciiTable
<632>   1722 CrsrData            1723 CrsrMask            1729 ONE
<633>   1730 TWO                 1731 THREE               1733 IObrd
<634>   1742 CPUbrd              1752 MEMbrd              1762 Xcard
<635>   1769 waiticon            1783 proicon             1792 upper
<636>   1802 driven              1810 insertd             1821 keybdout
<637>   1839 mouseout            1848 Question            1854 checkmrk
<638>   1865 badmrk              1875 diskette            1883 lisa
<639>   1910 INITMSG             1915 CROMMSG             1919 CROMMSG
<640>   1924 IORMMSG             1926 CPUMSG              1931 RAMMSG
<641>   1933 IOMSG               1935 IOSMSG              1937 DSKMSG
<642>   1939 BOOTERR             1941 DVCMSG              1944 ERRMSG
<643>   1946 EXCPMSG             1950 BOOTMSG             1953 BOOTMSG
<644>   1957 BADBOOT             1963 BRNMSG              1965 TIMMSG
<645>   1967 TWGMSG              1969 LOOPMSG             1971 PMMSG
<646>   1973 TWGFAIL             1975 TWGRSLT             1982 CONTMSG
<647>   1985 LEV1MSG             1988 LEV2MSG             1999 CHKMSG
<648>   2006 RTRYMSG             2013 CONTMSG             2020 STRTMSG
<649>   2027 PERIODS             2030 MENUHDG             2033 DISPMSG
<650>   2035 SETMSG              2037 CALLMSG             2041 LPMSG
<651>   2045 VIDMSG              2049 CYCLMSG             2053 QUITMSG
<652>   2056 MENUID              2074 NOTAMSG             2078 TSTMENU
<653>   2105 ADDRMSG             2107 DATAMSG             2109 CNTMSG
<654>   2111 DRVMSG              2113 DVCEMSG             2115 TSTMSG
<655>   2119 ADDRMSG             2121 DATAMSG             2123 CNTMSG
<656>   2125 TSTMSG              2130 WHATMSG             2137 VRSN
<657>   2143 VRSN                2146 VRSN                2154 HDGMSG
<658>   2157 VRSN                2158 REV                 2162 LAST
<659> ********************************************************************************
<660> * FILE LABEL STATISTICS
<661> ********************************************************************************
<662> File Lines           =     2165
<663> Label Count          =      168
<664> Local Label Count    =       37
<665> Min     Label Length =        3 out
<666> Max     Label Length =       11 DSPALRTICON
<667> Avg     Label Length =        6
<668> Std Dev Label Length =     1.50
<669>
<670> Label Length Frequency Counts:
<671>   No. Labels with Length   3:        4
```

```
<672>   No. Labels with Length   4:        9
<673>   No. Labels with Length   5:       18
<674>   No. Labels with Length   6:       35
<675>   No. Labels with Length   7:       61
<676>   No. Labels with Length   8:       25
<677>   No. Labels with Length   9:        8
<678>   No. Labels with Length  10:        7
<679>   No. Labels with Length  11:        1
<680> ********************************************************************************
<681> * END OF FILE: RM248.G.TEXT
<682> ********************************************************************************
<683>
<684> ********************************************************************************
<685> * SUMMARY FILE LABEL INFO:
<686> ********************************************************************************
<687> File Name List:
<688>   [  1]  Text File? Yes     Size:    85424 bytes - "RM248.E.TEXT"
<689>   [  2]  Text File? Yes     Size:    68486 bytes - "RM248.K.TEXT"
<690>   [  3]  Text File? Yes     Size:    74719 bytes - "RM248.S.TEXT"
<691>   [  4]  Text File? Yes     Size:    83283 bytes - "RM248.B.TEXT"
<692>   [  5]  Text File? Yes     Size:    90911 bytes - "RM248.M.TEXT"
<693>   [  6]  Text File? Yes     Size:    71125 bytes - "RM248.G.TEXT"
<694>                             --------------
<695>                                    473948
<696>
<697> All File Label Statistics:
<698>   File Lines           =    12523
<699>   Label Count          =     1504
<700>   Local Label Count    =      401
<701>   Min     Label Length =        2 R0
<702>   Max     Label Length =       13 MouseMovement
<703>   Avg     Label Length =        6
<704>   Std Dev Label Length =     1.49
<705>
<706> Label Length Frequency Counts:
<707>   No. Labels with Length   2:       13
<708>   No. Labels with Length   3:       35
<709>   No. Labels with Length   4:      140
<710>   No. Labels with Length   5:      222
<711>   No. Labels with Length   6:      389
<712>   No. Labels with Length   7:      462
<713>   No. Labels with Length   8:      181
<714>   No. Labels with Length   9:       32
<715>   No. Labels with Length  10:       20
<716>   No. Labels with Length  11:        4
<717>   No. Labels with Length  12:        4
<718>   No. Labels with Length  13:        2
<719> ********************************************************************************
<720> * END OF SUMMARY FILE LABEL INFO
<721> ********************************************************************************
<722>
<723> That's all Folks !
```

I wrote the following MPW tools that analyzed the Boot ROM source code:

DTCGetAsmComments
DTCAsmOpcodeInfo
DTCAsmLabelInfo

The source code for these tools follows:

MPW Tool Listing: DTCGetAsmComments

```
<  1> {
<  2>   | get assembler comments
<  3>   |
<  4>   | purpose: this program is an apple mpw shell tool that fetchs
<  5>   |          all comment lines from an assembler listing
<  6>   |
<  7>   | author     : david t craig, 736 edgewater, wichita, kansas 67230
<  8>   | date       : june 1993
<  9>   |
< 10>   | language   : apple mpw pascal 3.2
< 11>   | type       : apple mpw shell tool
< 12>   | environment: apple mpw shell
< 13>   |
< 14>   | input      : DTCGetAsmComments  comment-start-string  asm-text-file
< 15>   |
< 16>   | output     : list of all lines whose first character is in the
< 17>   |              comment-start-string parameter, output is to
< 18>   |              the mpw shell's standard output
< 19>   |
< 20>   |              each outputted line begins with the source file line
< 21>   |              number of the comment line
< 22>   |
< 23>   |              at the end of the output exists information on the
< 24>   |              total number of lines in the file, the total number of
< 25>   |              comment lines in the file, and the ratio of these
< 26>   |              two values which may be of some interest to those with
< 27>   |              an interest in software metrics (such as the good folks
< 28>   |              who write papers for the IEEE Computer magazine)
< 29>   |
< 30>   | example    : DTCGetAsmComments ";*" FooBar.a
< 31>   |
< 32>   |              this example searches for all lines in file FooBar.a which
< 33>   |              begin with either a ";" or "*" character and write those
< 34>   |              such lines to standard output
< 35> }
< 36>
< 37> program get_asm_comments;
< 38>
< 39> USES
< 40>   MemTypes,   { Macintosh common types }
< 41>   OSIntf,     { Macintosh Operating System interface }
< 42>   ToolIntf,   { Macintosh ToolBox interface }
< 43>   Packages,   { Macintosh Package interface }
< 44>   PasLibIntf, { Pascal runtime libary interface }
< 45>   IntEnv,     { MPW integrated environment interface }
< 46>   CursorCtl;  { MPW shell cursor unit }
< 47>
< 48> {$r+}
< 49>
< 50> type
< 51>   t_string = string[255];
< 52>
< 53> var
< 54>   arg_tool_name        : t_string;
< 55>   arg_comment_string   : t_string;
< 56>   arg_asm_file_name    : t_string;
< 57>   asm_file             : text;
< 58>   line_count           : longint;
< 59>   line_rem_count       : longint;  { remark/comment count }
< 60>   line_data            : t_string;
< 61>   line_rem_count_ratio : longint;  { remark / count }
< 62>   line_char_count      : longint;
< 63>   line_char_rem_count  : longint;
< 64>   error                : integer;
< 65>
```

```
< 66> function char_in_string (c : char; s : t_string) : boolean;
< 67>
< 68>   var
< 69>     in_string : boolean;
< 70>     i         : integer;
< 71>
< 72>   begin
< 73>     in_string := false;
< 74>
< 75>     for i := 1 to length(s) do
< 76>       if s[i] = c then
< 77>         in_string := true;
< 78>
< 79>     char_in_string := in_string;
< 80>   end;
< 81>
< 82> function rem_chars_in_line (s : t_string) : integer;
< 83>
< 84>   var
< 85>     rem_char_count : integer;
< 86>     i              : integer;
< 87>     done           : boolean;
< 88>
< 89>   begin
< 90>     rem_char_count := 0;
< 91>
< 92>     i    := length(s);
< 93>     done := false;
< 94>
< 95>     repeat
< 96>       if i <= 0 then
< 97>         done := true
< 98>       else
< 99>         begin
<100>           if s[i] = arg_comment_string[1] then
<101>             begin
<102>               done           := true;
<103>               rem_char_count := length(s) - i + 1;
<104>             end;
<105>         end;
<106>       i := i - 1;
<107>     until done;
<108>
<109>     rem_chars_in_line := rem_char_count;
<110>   end;
<111>
<112> begin
<113>   InitCursorCtl (NIL); { mpw beach ball cursor }
<114>
<115>   writeln('Assembly Source Code Comment Fetcher Utility');
<116>   writeln('Written by David T. Craig  [',compdate,' ',comptime,']');
<117>   writeln('736 Edgewater, Wichita, Kansas 67230   (316) 733-0914');
<118>   writeln;
<119>   writeln('This utility reads all comment lines in an assembly language');
<120>   writeln('source file and writes those comment lines to standard output.');
<121>   writeln;
<122>   writeln('At the end of the output listing exists some source file metrics.');
<123>   writeln('These metrics cover the number of source and comment lines, and the');
<124>   writeln('number of line and comment characters.  Note that the coment character');
<125>   writeln('count also takes into account comments existing at the end of a regular');
<126>   writeln('source line (eg: "  OPCODE OPERAND ; line end comment")');
<127>   writeln;
<128>
<129>   arg_tool_name := argv^[0]^;
<130>
```

```
<131>   if argc <> 3 then
<132>     begin
<133>       writeln('### ERROR : Wrong number of arguments for this tool');
<134>       writeln('### SYNTAX: ',arg_tool_name,' comment-start-string asm-text-file');
<135>       writeln('###          comment-start-string : comment line start (eg ";*:)');
<136>       writeln('###          asm-text-file        : file name (eg "FooBar.a")');
<137>     end
<138>   else
<139>     begin
<140>       arg_comment_string := argv^[1]^;
<141>       arg_asm_file_name  := argv^[2]^;
<142>
<143>       writeln('Assembly Source File Name: ',arg_asm_file_name);
<144>       writeln;
<145>
<146>       reset(asm_file,arg_asm_file_name); error := ioresult;
<147>
<148>       if error <> 0 then
<149>         writeln('### ERROR ',error:0,
<150>                 ' opening file "',arg_asm_file_name,'"')
<151>       else
<152>         begin
<153>           line_count          := 0;
<154>           line_rem_count      := 0;
<155>
<156>           line_char_count     := 0;
<157>           line_char_rem_count := 0;
<158>
<159>           writeln('LINE #  ASSEMBLY SOURCE FILE LINE');
<160>           writeln('======  ==============================================',
<161>                   '==============================================');
<162>
<163>           while not(eof(asm_file)) and (error = 0) do
<164>             begin
<165>               line_count := line_count + 1;
<166>               if (line_count mod 16) = 0 then
<167>                 SpinCursor(1);
<168>
<169>               readln(asm_file,line_data); error := ioresult;
<170>
<171>               if error <> 0 then
<172>                 writeln('### ERROR ',error:0,
<173>                         ' reading file "',arg_asm_file_name,'"')
<174>               else
<175>                 begin
<176>                   if length(line_data) > 0 then
<177>                     begin
<178>                       line_char_count := line_char_count + length(line_data);
<179>
<180>                       if char_in_string(line_data[1],arg_comment_string) then
<181>                         begin
<182>                           line_rem_count      := line_rem_count + 1;
<183>                           line_char_rem_count := line_char_rem_count +
<184>                                                  length(line_data);
<185>
<186>                           writeln(line_count:6,': ',line_data);
<187>                           error := ioresult;
<188>
<189>                           if error <> 0 then
<190>                             writeln('### ERROR ',error:0,
<191>                                     ' writing output file');
<192>                         end
<193>                       else
<194>                         begin
<195>                           { count comment chars at end of line too }
```

```
<196>
<197>                          line_char_rem_count := line_char_rem_count +
<198>                                                 rem_chars_in_line(line_data);
<199>                        end;
<200>                    end;
<201>                end;
<202>            end;
<203>
<204>          close(asm_file);
<205>
<206>          if error = 0 then
<207>            begin
<208>              if line_count > 0 then
<209>                line_rem_count_ratio := (line_rem_count * 100) div line_count
<210>              else
<211>                line_rem_count_ratio := 0;
<212>
<213>              writeln;
<214>              writeln('File Line and Comment Metrics:');
<215>              writeln;
<216>              writeln('  Total Lines               = ',line_count:8);
<217>              writeln('  Total Comment Lines       = ',line_rem_count:8);
<218>              writeln('  Comment / Line Count %    = ',line_rem_count_ratio:8,' %');
<219>
<220>              if line_char_count > 0 then
<221>                line_rem_count_ratio := (line_char_rem_count * 100) div
<222>                                        line_char_count
<223>              else
<224>                line_rem_count_ratio := 0;
<225>
<226>              writeln;
<227>              writeln('  Total Characters          = ',line_char_count:8);
<228>              writeln('  Total Comment Characters  = ',line_char_rem_count:8);
<229>              writeln('  Comment / Line Character % = ',line_rem_count_ratio:8,' %');
<230>            end;
<231>        end;
<232>    end;
<233>
<234>  writeln;
<235>  writeln('That''s all Folks !');
<236> end.
<237>
<238> { finis }
```

MPW Tool Listing: DTCAsmOpcodeInfo

```
<  1> { +--------------------------------------------------------------------------
<  2>   |
<  3>   |            FETCH ASSEMBLY LANGUAGE SOURCE FILE OPCODE STATISTICS
<  4>   |            ---------------------------------------------------
<  5>   |
<  6>   |                                Version 1.0
<  7>   |
<  8>   | purpose    : this program is an apple mpw shell tool that outputs
<  9>   |              statistical information about the opcodes existing in
< 10>   |              assembly language source code text files, this info
< 11>   |              consists of a list of all opcode names, their
< 12>   |              frequencies, and a histogram of these frequencies
< 13>   |
< 14>   | author     : david t craig
< 15>   | address    : 736 edgewater, wichita, kansas 67230
< 16>   | date       : july 1993
< 17>   |
< 18>   | language   : apple mpw pascal 3.2
```

```
< 19>   | type       : apple mpw shell tool
< 20>   | environment: apple mpw shell
< 21>   |
< 22>   | input      : DTCAsmOpcodeInfo  comment-char  asm-text-file-list
< 23>   |
< 24>   |                 where comment-char       is a list of characters
< 25>   |                                         denoting a comment line
< 26>   |
< 27>   |                       asm-text-file-list is a list of assembly language
< 28>   |                                         source code file names
< 29>   |
< 30>   | output     : Progress info goes to the screen and opcode info goes to
< 31>   |              standard output, the opcode info consists of a list of the
< 32>   |              opcodes and for each opcode appears its name, frequency, and
< 33>   |              a frequency histogram (3 lists are produced with 1st being
< 34>   |              an unsorted list, 2nd sorted by name, 3rd sorted by frequency)
< 35>   |
< 36>   | example    : DTCAsmOpcodeInfo  ";#"  FooBar.a  Frodor.a
< 37>   |
< 38>   | sample output:
< 39>   |
< 40>   |   the following command line was used to produce this section's
< 41>   |   sample output listing;
< 42>   |
< 43>   |      DTCAsmOpcodeInfo ";" FooBar.a  Frodor.a
< 44>   |
< 45>   |   the progress info looks like the following;
< 46>   |
< 47>   |      Assembly Source File Opcode Information Gatherer     Version: 1.0
< 48>   |      Written by David T. Craig [7/11/93 2:34:05 PM]
< 49>   |      736 Edgewater, Wichita, Kansas 67230
< 50>   |      Copyright (c) 1993 by David T. Craig
< 51>   |
< 52>   |      Current Date and Time:  Sunday, July 11, 1993  2:34:41 PM
< 53>   |
< 54>   |      Scanning file "FooBar.a" for opcodes ...
< 55>   |      Scanning file "Frodor.a" for opcodes ...
< 56>   |
< 57>   |      That's all, folks !
< 58>   |
< 59>   |   the opcode info looks like the following;
< 60>   |
< 61>   |      Assembly Source File Opcode Information Gatherer
< 62>   |      Version: 1.0  [7/11/93]
< 63>   |
< 64>   |       File #   1: FooBar.a
< 65>   |       File #   2: Frodor.a
< 66>   |
< 67>   |      Opcode List unsorted:
< 68>   |
< 69>   |        Minimum opcode frequency = 306
< 70>   |        Maximum opcode frequency = 804
< 71>   |
< 72>   |       #  Opcode          Frequency Histogram
< 73>   |      --- -------------- --------- ----------------------------------------
< 74>   |        1           .PAGE       115 ********
< 75>   |        2           .LIST        10 **
< 76>   |        3             BSR       479 ******************************
< 77>   |        4           CLR.L        95 *******
< 78>   |        5            BTST        86 *******
< 79>   |      --- -------------- --------- ----------------------------------------
< 80>   |
< 81>   |      Opcode List sorted by NAME:
< 82>   |
< 83>   |        Minimum opcode frequency = 306
```

```
< 84>   |        Maximum opcode frequency = 804
< 85>   |
< 86>   |       #  Opcode          Frequency Histogram
< 87>   |      --- --------------- --------- ----------------------------------------
< 88>   |        1       .ABSOLUTE         1 *
< 89>   |        2          .ALIGN         1 *
< 90>   |        3          .ASCII        76 ******
< 91>   |        4           .BYTE       306 ********************
< 92>   |        5           .ELSE        78 ******
< 93>   |      --- --------------- --------- ----------------------------------------
< 94>   |
< 95>   |      Opcode List sorted by FREQUENCY:
< 96>   |
< 97>   |        Minimum opcode frequency = 306
< 98>   |        Maximum opcode frequency = 804
< 99>   |
<100>   |       #  Opcode          Frequency Histogram
<101>   |      --- --------------- --------- ----------------------------------------
<102>   |        1            .EQU       804 ****************************************
<103>   |        2             BSR       479 ******************************
<104>   |        3          MOVE.L       383 ************************
<105>   |        4           MOVEQ       376 ************************
<106>   |        5           .BYTE       306 ********************
<107>   |      --- --------------- --------- ----------------------------------------
<108>   |
<109>   |      FINIS
<110>   |
<111>   | usage notes:
<112>   |
<113>   |   (0) a spinning beach ball cursor appears during file processing,
<114>   |       the spin direction alternates for every other file
<115>   |
<116>   |   (1) opcodes with up to 15 characters are handled (longer opcodes
<117>   |       are truncated), opcodes are read until either a space character
<118>   |       or control character is found, or until the end of the line
<119>   |       is reached
<120>   |
<121>   | theory of operation:
<122>   |
<123>   |   for each argument file all the lines are scanned with each line's
<124>   |   opcode name extracted (if existing) and name either added to
<125>   |   opcode list or if name already exists in the list then name's
<126>   |   frequency is incremented
<127>   |
<128>   | programming notes:
<129>   |
<130>   |   (0) opcode names are delimited by "white space" which consists of
<131>   |       all characters less than or equal to ascii space character (#32)
<132>   |
<133>   |   (1) all opcode names are stored in the opcode list with uppercase
<134>   |       characters
<135>   |
<136>   |   (2) opcode list contains opcode name and frequency, list is a simple
<137>   |       array without any specific ordering
<138>   |
<139>   |   (3) opcode list is finite in length
<140>   |
<141>   +-------------------------------------------------------------------------
<142> }
<143>
<144> program asm_opcode_info;
<145>
<146> USES
<147>   MemTypes,   { Macintosh common types }
<148>   OSIntf,     { Macintosh Operating System interface }
```

```
<149>   ToolIntf,   { Macintosh ToolBox interface }
<150>   Packages,   { Macintosh Package interface }
<151>   PasLibIntf, { Pascal runtime libary interface }
<152>   IntEnv,     { MPW integrated environment interface }
<153>   ErrMgr,     { MPW error manager interface }
<154>   CursorCtl;  { MPW shell cursor unit }
<155>
<156> {$r+}
<157>
<158> const
<159>   k_pgm_title           = 'Assembly Source File Opcode Information Gatherer';
<160>   k_pgm_version         = '1.0';
<161>   k_pgm_date            = compdate; { mpw pascal specific }
<162>   k_pgm_time            = comptime; { mpw pascal specific }
<163>   k_pgm_author          = 'David T. Craig';
<164>   k_pgm_address         = '736 Edgewater, Wichita, Kansas 67230';
<165>   k_pgm_copyright       = 'Copyright (c) 1993 by David T. Craig';
<166>
<167>   k_max_opcodes         = 1500;
<168>   k_max_opcode_name_len = 15;
<169>
<170>   k_err_alpha           = 32000;
<171>   k_err_list_full       = 32000;
<172>   k_err_omega           = 32000;
<173>
<174> type
<175>   t_opcode_name         = string[k_max_opcode_name_len];
<176>   t_opcode_freq         = integer;
<177>   t_opcode_info         = record
<178>                             oi_name : t_opcode_name;
<179>                             oi_freq : t_opcode_freq;
<180>                           end;
<181>
<182>   t_opcode_list         = array [1..k_max_opcodes] of t_opcode_info;
<183>   t_opcode_list_ptr     = ^t_opcode_list;
<184>
<185>   t_sort_order          = (sort_by_name , sort_by_freq);
<186>
<187>   t_string              = str255;
<188>
<189> var
<190>   g_mac_date_time       : t_string;          { current machine date/time }
<191>   g_arg_tool_name       : t_string;
<192>   g_arg_comment_chars   : t_string;
<193>   g_arg_index           : integer;
<194>   g_arg_file_name       : t_string;
<195>   g_process_error       : integer;
<196>   g_process_error_total : integer;
<197>   g_good_list           : boolean;
<198>   g_opcode_list         : t_opcode_list_ptr;
<199>   g_opcode_list_count   : integer;
<200>
<201> {$S SgAsmOpcodeInfo}
<202>
<203> { ------------------------------------------------------------------------ }
<204>
<205> procedure fetch_mac_error_message (    the_error     : integer;
<206>                                      var the_error_msg : t_string);
<207>
<208>   var
<209>     msg : string[99];
<210>
<211>   begin
<212>     if (the_error >= k_err_alpha) and (the_error <= k_err_omega) then
<213>       begin
```

```
<214>         case the_error of
<215>           k_err_list_full : msg := 'Opcode list is full.';
<216>           otherwise       msg := 'Unknown tool error (contact programmer).';
<217>         end; { case }
<218>
<219>         the_error_msg := msg;
<220>       end
<221>     else
<222>       begin
<223>         GetSysErrText (the_error,@the_error_msg);
<224>       end;
<225>   end;
<226>
<227> { ---------------------------------------------------------------------- }
<228>
<229> procedure show_error (the_error : integer; the_msg : t_string);
<230>
<231>   var
<232>     err_msg : t_string;
<233>
<234>   begin
<235>     fetch_mac_error_message(the_error,err_msg);
<236>
<237>     writeln(diagnostic,'### ERROR ',the_error:0,': ',the_msg);
<238>     writeln(diagnostic,'          ',err_msg);
<239>   end;
<240>
<241> { ---------------------------------------------------------------------- }
<242>
<243> procedure get_current_date_time (var s : t_string);
<244>
<245>   var
<246>     mac_date_time_info : longint;
<247>     date_time_string   : str255;
<248>
<249>   begin
<250>     getdatetime  (mac_date_time_info);
<251>     iudatestring (mac_date_time_info,longdate,date_time_string);
<252>
<253>     s := date_time_string;
<254>
<255>     iutimestring (mac_date_time_info,true,date_time_string);
<256>
<257>     s := concat(s,'  ',date_time_string);
<258>   end;
<259>
<260> { ---------------------------------------------------------------------- }
<261>
<262> function is_whitespace (c : char) : boolean;
<263>
<264>   const
<265>     k_max_whitespace = chr(32); { ascii SPACE }
<266>
<267>   begin
<268>     is_whitespace := (c <= k_max_whitespace);
<269>   end;
<270>
<271> { ---------------------------------------------------------------------- }
<272>
<273> procedure trim_leading_whitespace (var s : t_string; var c1 : char);
<274>
<275>   var
<276>     done : boolean;
<277>
<278>   begin
```

```
<279>     if length(s) > 0 then  { setup first string character for caller }
<280>       c1 := s[1]
<281>     else
<282>       c1 := chr(0);
<283>
<284>     done := false;
<285>
<286>     repeat
<287>       begin
<288>         if length(s) = 0 then
<289>           done := true
<290>         else
<291>           begin
<292>             if is_whitespace(s[1]) then
<293>               delete(s,1,1)
<294>             else
<295>               done := true;
<296>           end;
<297>       end;
<298>     until done;
<299>   end;
<300>
<301> { ------------------------------------------------------------------------ }
<302>
<303> procedure fetch_next_word (s : t_string; var next_word : t_string);
<304>
<305>   var
<306>     i    : integer;
<307>     done : boolean;
<308>
<309>   begin
<310>     next_word := '';
<311>
<312>     if length(s) > 0 then
<313>       begin
<314>         i    := 0;
<315>         done := false;
<316>
<317>         repeat
<318>           begin
<319>             i := i + 1;
<320>
<321>             if i > length(s) then
<322>               begin
<323>                 i := i - 1;
<324>                 if i > 0 then
<325>                   next_word := copy(s,1,i);
<326>
<327>                 done := true;
<328>               end
<329>             else
<330>               begin
<331>                 if is_whitespace(s[i]) then
<332>                   begin
<333>                     i := i - 1;
<334>                     if i > 0 then
<335>                       next_word := copy(s,1,i);
<336>
<337>                     done := true;
<338>                   end;
<339>               end;
<340>           end;
<341>         until done;
<342>       end;
<343>   end;
```

```
<344>
<345> { ------------------------------------------------------------------------ }
<346>
<347> procedure uppcase_opcode_name (var s : t_opcode_name);
<348>
<349>   var
<350>     i : integer;
<351>
<352>   begin
<353>     i := length(s);
<354>
<355>     while i > 0 do
<356>       begin
<357>         if s[i] in ['a'..'z'] then
<358>           s[i] := chr( ord(s[i]) - ord('a') + ord('A') );
<359>
<360>         i := i - 1;
<361>       end;
<362>   end;
<363>
<364> { ------------------------------------------------------------------------ }
<365>
<366> procedure initialize_opcode_list (var the_good_init : boolean);
<367>
<368>   var
<369>     list_index : integer;
<370>
<371>   begin
<372>     the_good_init := true;
<373>
<374>     g_opcode_list := t_opcode_list_ptr(newptr(sizeof(t_opcode_list)));
<375>
<376>     if g_opcode_list = nil then
<377>       the_good_init := false
<378>     else
<379>       begin
<380>         g_opcode_list_count := 0;
<381>
<382>         for list_index := 1 to k_max_opcodes do
<383>           begin
<384>             g_opcode_list^[list_index].oi_name := '';
<385>             g_opcode_list^[list_index].oi_freq := 0;
<386>           end;
<387>       end;
<388>   end;
<389>
<390> { ------------------------------------------------------------------------ }
<391>
<392> procedure terminate_opcode_list (var the_good_term : boolean);
<393>
<394>   begin
<395>     the_good_term := true;
<396>
<397>     if g_opcode_list = nil then
<398>       the_good_term := false
<399>     else
<400>       begin
<401>         disposptr(ptr(g_opcode_list));
<402>
<403>         g_opcode_list       := nil;
<404>         g_opcode_list_count := 0;
<405>       end;
<406>   end;
<407>
<408> { ------------------------------------------------------------------------ }
```

```
<409>
<410> procedure get_asm_line_opcode (    the_asm_line : t_string;
<411>                                var the_opcode   : t_opcode_name);
<412>
<413>   var
<414>     first_char  : char;
<415>     comment_set : set of char;
<416>     i           : integer;
<417>     next_word   : t_string;
<418>
<419>   begin
<420>     the_opcode := '';
<421>
<422>     if length(the_asm_line) > 0 then
<423>       begin
<424>         trim_leading_whitespace(the_asm_line,first_char);
<425>
<426>         if length(the_asm_line) > 0 then
<427>           begin
<428>             comment_set := [];
<429>             for i := 1 to length(g_arg_comment_chars) do
<430>               comment_set := comment_set + [g_arg_comment_chars[i]];
<431>
<432>             if not(the_asm_line[1] in comment_set) then
<433>               begin
<434>                 if is_whitespace(first_char) then
<435>                   begin
<436>                     { "  opcode ..." }
<437>
<438>                     fetch_next_word(the_asm_line,next_word);
<439>
<440>                     if length(next_word) >= (sizeof(the_opcode) - 1) then
<441>                       next_word := copy(next_word,1,sizeof(the_opcode)-1);
<442>
<443>                     the_opcode := next_word;
<444>                   end
<445>                 else
<446>                   begin
<447>                     { "label [opcode] ..." }
<448>
<449>                     fetch_next_word(the_asm_line,next_word);
<450>                     delete(the_asm_line,1,length(next_word));
<451>                     trim_leading_whitespace(the_asm_line,first_char);
<452>
<453>                     { "opcode" or "" }
<454>
<455>                     if length(the_asm_line) > 0 then
<456>                       begin
<457>                         if not(the_asm_line[1] in comment_set) then
<458>                           begin
<459>                             fetch_next_word(the_asm_line,next_word);
<460>
<461>                             if length(next_word) >= (sizeof(the_opcode) - 1) then
<462>                               next_word := copy(next_word,1,sizeof(the_opcode)-1);
<463>
<464>                             the_opcode := next_word;
<465>                           end;
<466>                       end;
<467>                   end;
<468>               end;
<469>           end;
<470>       end;
<471>   end;
<472>
<473> { ------------------------------------------------------------------------ }
```

```
<474>
<475> procedure find_opcode_in_list (    the_name       : t_opcode_name;
<476>                                var the_list_index : integer);
<477>
<478>   var
<479>     opcode_found : boolean;
<480>
<481>   begin
<482>     the_list_index := 0;
<483>
<484>     if g_opcode_list_count > 0 then
<485>       begin
<486>         opcode_found := false;
<487>
<488>         repeat
<489>           begin
<490>             the_list_index := the_list_index + 1;
<491>
<492>             if the_name = g_opcode_list^[the_list_index].oi_name then
<493>               opcode_found := true;
<494>           end;
<495>         until (the_list_index >= g_opcode_list_count) or opcode_found;
<496>
<497>         if not(opcode_found) then
<498>           the_list_index := 0;
<499>       end;
<500>   end;
<501>
<502> { ------------------------------------------------------------------------ }
<503>
<504> procedure add_opcode_to_list (    the_name  : t_opcode_name;
<505>                               var the_error : integer);
<506>
<507>   var
<508>     list_index : integer;
<509>
<510>   begin
<511>     the_error := noerr;
<512>
<513>     uppcase_opcode_name(the_name);
<514>
<515>     find_opcode_in_list (the_name,list_index);
<516>
<517>     if list_index = 0 then
<518>       begin
<519>         { ++++++++++++++++++++++++++++++++++++ name not in list, so add }
<520>
<521>         if g_opcode_list_count >= k_max_opcodes then
<522>           begin
<523>             the_error := k_err_list_full; { list full !!! }
<524>           end
<525>         else
<526>           begin
<527>             g_opcode_list_count := g_opcode_list_count + 1;
<528>
<529>             with g_opcode_list^[g_opcode_list_count] do
<530>               begin
<531>                 oi_name := the_name;
<532>                 oi_freq := 1;
<533>               end;
<534>           end;
<535>       end
<536>     else { list_index > 0 }
<537>       begin
<538>         { ++++++++++++++++++++++++++++++++++++ name in list, so inc freq }
```

```
<539>
<540>         with g_opcode_list^[list_index] do
<541>           oi_freq := oi_freq + 1;
<542>       end;
<543>   end;
<544>
<545> { ------------------------------------------------------------------------ }
<546>
<547> procedure sort_opcode_list (sort_order : t_sort_order);
<548>
<549>   var
<550>     list_sorted   : boolean;
<551>     list_index    : integer;
<552>     opcode_info_a : t_opcode_info;
<553>     opcode_info_b : t_opcode_info;
<554>     swap_em       : boolean;
<555>
<556>   begin
<557>     { simple bubble sort (use comb sort if bs is too slow) }
<558>
<559>     if g_opcode_list_count >= 2 then
<560>       begin
<561>         repeat
<562>           begin
<563>             list_sorted := true;
<564>
<565>             for list_index := 1 to (g_opcode_list_count - 1) do
<566>               begin
<567>                 opcode_info_a := g_opcode_list^[list_index    ];
<568>                 opcode_info_b := g_opcode_list^[list_index + 1];
<569>
<570>                 swap_em := false;
<571>
<572>                 case sort_order of
<573>                   sort_by_name :
<574>                     if opcode_info_a.oi_name > opcode_info_b.oi_name then
<575>                       swap_em := true;
<576>                   sort_by_freq :
<577>                     if opcode_info_a.oi_freq < opcode_info_b.oi_freq then
<578>                       swap_em := true;
<579>                 end; { case }
<580>
<581>                 if swap_em then
<582>                   begin
<583>                     list_sorted := false;
<584>
<585>                     g_opcode_list^[list_index    ] := opcode_info_b;
<586>                     g_opcode_list^[list_index + 1] := opcode_info_a;
<587>
<588>                     spincursor(1);
<589>                   end;
<590>               end; { for list_index }
<591>           end;
<592>         until list_sorted;
<593>       end;
<594>   end;
<595>
<596> { ------------------------------------------------------------------------ }
<597>
<598> procedure dump_opcode_list (the_title : t_string);
<599>
<600>   const
<601>     k_h_len    = 50;
<602>     k_h_symbol = '*';
<603>     k_divider1 = '--- --------------- --------- ';
```

```
<604>     k_divider2 = '------------------------------------------------';
<605>
<606>   var
<607>     list_index  : integer;
<608>     freq_min    : t_opcode_freq;
<609>     freq_max    : t_opcode_freq;
<610>     freq        : t_opcode_freq;
<611>     h_value     : integer;
<612>
<613>   begin
<614>     writeln;
<615>     writeln(the_title);
<616>     writeln;
<617>
<618>     if g_opcode_list_count > 0 then
<619>       begin
<620>         freq_min := g_opcode_list^[1].oi_freq;
<621>         freq_max := g_opcode_list^[1].oi_freq;
<622>
<623>         for list_index := 1 to g_opcode_list_count do
<624>           begin
<625>             freq := g_opcode_list^[list_index].oi_freq;
<626>
<627>             if freq < freq_min then freq_min := freq;
<628>             if freq > freq_max then freq_max := freq;
<629>           end;
<630>       end;
<631>
<632>     writeln('  Minimum opcode frequency = ',freq_min:0);
<633>     writeln('  Maximum opcode frequency = ',freq_max:0);
<634>     writeln;
<635>
<636>     writeln(' #  Opcode          Frequency Histogram');
<637>     writeln(k_divider1,k_divider2);
<638>
<639>     if g_opcode_list_count <= 0 then
<640>       writeln('(no opcodes exist)')
<641>     else
<642>       begin
<643>         for list_index := 1 to g_opcode_list_count do
<644>           begin
<645>             write(list_index:3,' ');
<646>             write(g_opcode_list^[list_index].oi_name:15,' ');
<647>             write(g_opcode_list^[list_index].oi_freq: 9,' ');
<648>
<649>             if freq_max <> freq_min then
<650>               begin
<651>                 with g_opcode_list^[list_index] do
<652>                   h_value := k_h_len - (ord4(k_h_len-1) * ord4(freq_max-oi_freq) div
<653>                                         ord4(freq_max-freq_min));
<654>               end
<655>             else { freq_max = freq_min }
<656>               begin
<657>                 h_value := k_h_len;
<658>               end;
<659>
<660>             while h_value > 0 do
<661>               begin
<662>                 write(k_h_symbol);
<663>                 h_value := h_value - 1;
<664>               end;
<665>
<666>             writeln;
<667>           end;
<668>       end;
```

```
<669>
<670>     writeln(k_divider1,k_divider2);
<671>
<672>   end;
<673>
<674> { ------------------------------------------------------------------------ }
<675>
<676> procedure process_asm_file (    the_file_name   : t_string;
<677>                                 the_file_number : integer;
<678>                             var the_error       : integer);
<679>
<680>   var
<681>     f           : text;
<682>     l           : string[255];
<683>     c           : longint;
<684>     opcode_name : t_opcode_name;
<685>
<686>   begin
<687>     the_error := noerr;
<688>
<689>     writeln(' File # ',the_file_number:3,': ',the_file_name);
<690>
<691>     reset(f,the_file_name); the_error := ioresult;
<692>
<693>     if the_error = noerr then
<694>       begin
<695>         while not(eof(f)) and (the_error = noerr) do
<696>           begin
<697>             readln(f,l); the_error := ioresult;
<698>
<699>             if the_error = noerr then
<700>               begin
<701>                 c := c + 1;
<702>                 if c mod 16 = 0 then
<703>                   begin
<704>                     if odd(the_file_number) then spincursor(+1)
<705>                                             else spincursor(-1);
<706>                   end;
<707>
<708>                 if length(l) > 0 then
<709>                   begin
<710>                     get_asm_line_opcode(l,opcode_name);
<711>
<712>                     if length(opcode_name) > 0 then
<713>                       begin
<714>                         add_opcode_to_list(opcode_name,the_error);
<715>                       end;
<716>                   end;
<717>               end;
<718>           end; { while not(eof) }
<719>
<720>         close(f);
<721>       end;
<722>   end;
<723>
<724> { ------------------------------------------------------------------------ }
<725> {                              M   A   I   N                             }
<726> { ------------------------------------------------------------------------ }
<727>
<728> begin { ------------ MAIN : asm_opcode_info ------------ }
<729>
<730>   PLSetVBuf     (output,nil,_iolbf,0); { mpw output buffer flushing control }
<731>   InitCursorCtl (nil);                 { mpw beach ball cursor }
<732>   InitErrMgr    ('','',false);         { mpw error message manager }
<733>
```

```
<734>   get_current_date_time(g_mac_date_time);
<735>
<736>   writeln(diagnostic);
<737>   writeln(diagnostic,k_pgm_title,'     Version: ',k_pgm_version);
<738>   writeln(diagnostic,'Written by ',k_pgm_author,' [',k_pgm_date,' ',k_pgm_time,']');
<739>   writeln(diagnostic,k_pgm_address);
<740>   writeln(diagnostic,k_pgm_copyright);
<741>   writeln(diagnostic);
<742>   writeln(diagnostic,'Current Date and Time:  ',g_mac_date_time);
<743>   writeln(diagnostic);
<744>
<745>   g_arg_tool_name := argv^[0]^;
<746>
<747>   if argc < 3 then
<748>     begin
<749>       writeln(diagnostic,'### ERROR : Tool argument list is invalid');
<750>       writeln(diagnostic,'### SYNTAX: ',g_arg_tool_name,' comment-char ',
<751>                                                          ' asm-text-file-list');
<752>     end
<753>   else
<754>     begin
<755>       g_arg_comment_chars := argv^[1]^;
<756>
<757>       if length(g_arg_comment_chars) = 0 then
<758>         begin
<759>           writeln(diagnostic,'### ERROR: Comment character string is empty');
<760>         end
<761>       else
<762>         begin
<763>           initialize_opcode_list(g_good_list);
<764>
<765>           if not(g_good_list) then
<766>             writeln(diagnostic,'### ERROR: Creating opcode list failed !')
<767>           else
<768>             begin
<769>               writeln(k_pgm_title);
<770>               writeln('Version: ',k_pgm_version,'  [',k_pgm_date,']');
<771>               writeln;
<772>
<773>               g_process_error_total := 0;
<774>               g_arg_index           := 1;
<775>
<776>               while g_arg_index < (argc - 1) do
<777>                 begin
<778>                   g_arg_index     := g_arg_index + 1;
<779>                   g_arg_file_name := argv^[g_arg_index]^;
<780>
<781>                   writeln(diagnostic,'Scanning file "',g_arg_file_name,'" ',
<782>                                      'for opcodes ...');
<783>
<784>                   process_asm_file(g_arg_file_name,g_arg_index-1,g_process_error);
<785>
<786>                   if g_process_error <> noerr then
<787>                     begin
<788>                       show_error(g_process_error,'Scanning file failed !');
<789>                       g_process_error_total:= g_process_error_total + 1;
<790>                     end;
<791>                 end; { while g_arg_index }
<792>
<793>               if g_process_error_total = 0 then
<794>                 begin
<795>                   dump_opcode_list('Opcode List unsorted:');
<796>
<797>                   sort_opcode_list(sort_by_name);
<798>                   dump_opcode_list('Opcode List sorted by NAME:');
```

```
<799>
<800>                   sort_opcode_list(sort_by_freq);
<801>                   dump_opcode_list('Opcode List sorted by FREQUENCY:');
<802>
<803>                   writeln;
<804>                   writeln('FINIS');
<805>                 end;
<806>
<807>               terminate_opcode_list(g_good_list);
<808>
<809>               if not(g_good_list) then
<810>                 writeln(diagnostic,'### ERROR: Deallocating opcode list failed !');
<811>             end;
<812>         end;
<813>     end;
<814>
<815>   writeln(diagnostic);
<816>   writeln(diagnostic,'That''s all, folks !');
<817>
<818> end.  { ------------ MAIN : asm_opcode_info ------------ }
<819>
<820> { finis }
```

MPW Tool Listing: DTCAsmLabelInfo

```
<  1> { +--------------------------------------------------------------------------
<  2>   |
<  3>   |            FETCH ASSEMBLY LANGUAGE SOURCE FILE LABEL STATISTICS
<  4>   |            ---------------------------------------------------
<  5>   |
<  6>   |                                Version 1.0
<  7>   |
<  8>   | purpose    : this program is an apple mpw shell tool that outputs
<  9>   |              statisitcal information about the labels existing in
< 10>   |              assembly language source code text files
< 11>   |
< 12>   | author     : david t craig
< 13>   | address    : 736 edgewater, wichita, kansas 67230
< 14>   | date       : june 1993
< 15>   |
< 16>   | language   : apple mpw pascal 3.2
< 17>   | type       : apple mpw shell tool
< 18>   | environment: apple mpw shell
< 19>   |
< 20>   | input      : DTCAsmLabelInfo  local-label-char  asm-text-file-list
< 21>   |
< 22>   |                 where local-label-char   is a single character which
< 23>   |                                          exists at the beginning of
< 24>   |                                          local variable names
< 25>   |
< 26>   |                       asm-text-file-list is a list of assembly language
< 27>   |                                          source code file names
< 28>   |
< 29>   |              note: if local-label-char contains the word "DEBUG" then
< 30>   |                    special internal debugging info is outputted (phrase
< 31>   |                    case in-sensitive, so "DEBUG" is the same as "debug")
< 32>   |
< 33>   | output     : for each file output consists of a set of lists as follows;
< 34>   |
< 35>   |                - list of all regular labels found in the file
< 36>   |                  (note: local labels are not listed)
< 37>   |                - list of file label statistics
< 38>   |
< 39>   |              at the end of the output exists summary information about
```

```
< 40>   |              all the processed files;
< 41>   |
< 42>   |                - list of all processed files, whether they are standard
< 43>   |                  Macintosh text files
< 44>   |                - statistics for all the labels in all the files
< 45>   |                - list of label length frequencies
< 46>   |
< 47>   | example    : DTCAsmLabelInfo  "@"  FooBar.a  Frodor.a
< 48>   |
< 49>   |              this example states that the character that begins a local
< 50>   |              label name is "@" (eg: "@9") and that there are two files
< 51>   |              to process, FooBar.a and Frodor.a
< 52>   |
< 53>   | sample output:
< 54>   |
< 55>   |   the following command line was used to produce this section's
< 56>   |   sample output listing;
< 57>   |
< 58>   |      DTCAsmLabelInfo  "@"  FooBar.a  Frodor.a
< 59>   |
< 60>   |   at the beginning of the output appears some information about the
< 61>   |   program, its author, and the current date and time;
< 62>   |
< 63>   |      Assembly Source File Label Information Gatherer     Version: 1.0
< 64>   |      Written by David T. Craig  [6/13/93 4:40:18 PM]
< 65>   |      736 Edgewater, Wichita, Kansas 67230
< 66>   |      Copyright (c) 1993 by David T. Craig
< 67>   |
< 68>   |      Current Date and Time:  Sunday, June 13, 1993  4:40:38 PM
< 69>   |
< 70>   |   for each file the following information appears;
< 71>   |
< 72>   |      ************************************************************
< 73>   |      * LABEL INFO FOR FILE : FooBar.a
< 74>   |      ************************************************************
< 75>   |          15 VIDTST          23 VIDCHK          47 VIDERR
< 76>   |          54 VIDXIT          95 RDSERN         122 GetBits1:
< 77>   |         152 GetBits2:      179 GetBytes:      213 CheckSum:
< 78>   |         250 Exit:          263 FindSync:      288 GetNibbles:
< 79>   |         ...
< 80>   |
< 81>   |      ************************************************************
< 82>   |      * FILE LABEL STATISTICS
< 83>   |      ************************************************************
< 84>   |      File Lines           =     2032
< 85>   |      Label Count          =      123
< 86>   |      Local Label Count    =      100
< 87>   |      Min     Label Length =        3 Tag
< 88>   |      Max     Label Length =       11 GetNibbles:
< 89>   |      Avg     Label Length =        6
< 90>   |      Std Dev Label Length =     1.18
< 91>   |
< 92>   |      Label Length Frequency Counts:
< 93>   |        No. Labels with Length   3:        1
< 94>   |        No. Labels with Length   4:        1
< 95>   |        No. Labels with Length   5:       21
< 96>   |        No. Labels with Length   6:       49
< 97>   |        No. Labels with Length   7:       32
< 98>   |        No. Labels with Length   8:       12
< 99>   |        No. Labels with Length   9:        6
<100>   |        No. Labels with Length  10:        0
<101>   |        No. Labels with Length  11:        1
<102>   |      ************************************************************
<103>   |      * END OF FILE: FooBar.a
<104>   |      ************************************************************
```

```
<105>   |
<106>   |   at the end of the output appears summary information;
<107>   |
<108>   |      ************************************************************
<109>   |      * SUMMARY FILE LABEL INFO:
<110>   |      ************************************************************
<111>   |      File Name List:
<112>   |        [  1]  Text File? Yes     Size:    83283 bytes - "FooBar.a"
<113>   |        [  2]  Text File? Yes     Size:    85424 bytes - "Frodor.a"
<114>   |                                  --------------
<115>   |                                         168707
<116>   |
<117>   |      All File Label Statistics:
<118>   |        File Lines           =    12523
<119>   |        Label Count          =     1504
<120>   |        Local Label Count    =      401
<121>   |        Min     Label Length =        2 R0
<122>   |        Max     Label Length =       13 MouseMovement
<123>   |        Avg     Label Length =        6
<124>   |        Std Dev Label Length =     1.49
<125>   |
<126>   |      Label Length Frequency Counts:
<127>   |        No. Labels with Length   2:       13
<128>   |        No. Labels with Length   3:       35
<129>   |        No. Labels with Length   4:      140
<130>   |        No. Labels with Length   5:      222
<131>   |        No. Labels with Length   6:      389
<132>   |        No. Labels with Length   7:      462
<133>   |        No. Labels with Length   8:      181
<134>   |        No. Labels with Length   9:       32
<135>   |        No. Labels with Length  10:       20
<136>   |        No. Labels with Length  11:        4
<137>   |        No. Labels with Length  12:        4
<138>   |        No. Labels with Length  13:        2
<139>   |      ************************************************************
<140>   |      * END OF SUMMARY FILE LABEL INFO
<141>   |      ************************************************************
<142>   |
<143>   | usage notes:
<144>   |
<145>   |   (0) a spinning beach ball cursor appears during file processing,
<146>   |       the spin direction alternates for every other file
<147>   |
<148>   |   (1) labels with up to 63 characters are handled (longer labels
<149>   |       are truncated), labels are read until either a space character
<150>   |       or control character is found, or until the end of the line
<151>   |       is reached
<152>   |
<153>   |   (2) only the first 20 characters of a label appear in the label
<154>   |       list (most labels should not be longer than this limit)
<155>   |
<156>   |   (3) local labels are labels which are used within an assembly
<157>   |       source file for localized branches and exist only within a
<158>   |       very narrow scope (for example, the mpw 68000 assembler
<159>   |       denotes local labels with "@" followed by a number [@9]),
<160>   |       this program supports any single character to desinate that
<161>   |       a label is a local label (actually, a multi-character string
<162>   |       may be entered for the local label character, but only the
<163>   |       first character will be used by this program)
<164>   |
<165>   |   (4) only macintosh text files are processed (ie files with macintosh
<166>   |       file type of "TEXT"), if a non-text file is entered the program
<167>   |       skip the file and not process it
<168>   |
<169>   |   (5) only files with data in them are processed (files with no bytes
```

```
<170>   |       are not processed since there is nothing to process)
<171>   |
<172>   |   (6) input file lines are read with up to 255 characters, longer
<173>   |       lines will most likely cause a read error from the pascal
<174>   |       file i/o macintosh library
<175>   |
<176>   | theory of operation:
<177>   |
<178>   |   each file is processed which results in
<179>   |     a) the output of all found label names (excluding local labels),
<180>   |     b) various statistics are gathered, outputted, and returned to
<181>   |        the main program file-fetch loop
<182>   |
<183>   |   file label statistics are accumulated by the main program loop,
<184>   |   processed, and outputted for a single set of statistical data
<185>   |   for all the inputted file names
<186>   |
<187>   |   special debugging data is also outputted if the local label argument
<188>   |   string contains a special debug-me flag, all debug data is
<189>   |   prefaced by a special phrase ("%%%") to ease human identification
<190>   |   of debugging data
<191>   |
<192>   | programming notes:
<193>   |
<194>   |   (0) standard deviation is calculated as follows:
<195>   |
<196>   |                 +-                    -+
<197>   |       std.dev = | sum(x^2) - sum(x)^2 | ^ (1/2)
<198>   |                 |            -------- |
<199>   |                 |                n    |
<200>   |                 | ------------------- |
<201>   |                 |        n - 1        |
<202>   |                 +-                   -+
<203>   |
<204>   |       reference: Personal Programming: TI-58C/59 Calculator Owner's Manual
<205>   |                  Texas Instruments, 1979, p. V-34
<206>   +-------------------------------------------------------------------------------
<207> }
<208>
<209> program get_asm_label_info;
<210>
<211> USES
<212>   MemTypes,   { Macintosh common types }
<213>   OSIntf,     { Macintosh Operating System interface }
<214>   ToolIntf,   { Macintosh ToolBox interface }
<215>   Packages,   { Macintosh Package interface }
<216>   PasLibIntf, { Pascal runtime libary interface }
<217>   IntEnv,     { MPW integrated environment interface }
<218>   ErrMgr,     { MPW error manager interface }
<219>   CursorCtl;  { MPW shell cursor unit }
<220>
<221> {$r+}
<222>
<223> const
<224>   k_pgm_title        = 'Assembly Source File Label Information Gatherer';
<225>   k_pgm_version      = '1.0';
<226>   k_pgm_date         = compdate; { mpw pascal specific }
<227>   k_pgm_time         = comptime; { mpw pascal specific }
<228>   k_pgm_author       = 'David T. Craig';
<229>   k_pgm_address      = '736 Edgewater, Wichita, Kansas 67230';
<230>   k_pgm_copyright    = 'Copyright (c) 1993 by David T. Craig';
<231>
<232>   k_max_label_length = 63; { max no. of characters supported per label name }
<233>   k_divider          = '****************************************';
<234>   k_debug_phrase     = 'DEBUG'; { for use by local-label-char argument }
```

```
<235>
<236>   { speical tool-specific error codes }
<237>
<238>   k_err_alpha         = 32700;
<239>   k_err_not_text_file = 32700; { file is not a text file }
<240>   k_err_omega         = 32720;
<241>
<242> type
<243>   t_string          = string[255];
<244>   t_counter         = longint;
<245>
<246>     { counts of label lengths, eg L[1] contains count for all labels }
<247>     { with 1 character, L[2] count for all labels with 2 characters  }
<248>
<249>   t_length_list     = array [1..k_max_label_length] of t_counter;
<250>
<251>   t_file_label_info = record
<252>                         i_file_lines   : t_counter; { lines in file }
<253>                         i_label_count  : t_counter; { regular labels }
<254>                         i_llabel_count : t_counter; { local labels }
<255>                         i_length_min   : t_counter; { min label length }
<256>                         i_length_min_n : t_string;
<257>                         i_length_max   : t_counter; { max label length }
<258>                         i_length_max_n : t_string;
<259>                         i_length_total : t_counter; { total number of label chars }
<260>                         i_length_avg   : t_counter; { avg label length }
<261>                         i_length_std   : real;      { standard deviation length }
<262>                         i_length_stdx2 : t_counter; { std dev: sum(x^2) }
<263>                         i_length_stdx  : t_counter; { std dev: sum(x)^2 }
<264>                         i_length_list  : t_length_list;
<265>                       end;
<266>
<267> var
<268>   g_debug_tool        : boolean;           { write special debugging info }
<269>   g_mac_date_time     : t_string;          { current machine date/time }
<270>   g_arg_tool_name     : t_string;
<271>   g_arg_local_label   : t_string;
<272>   g_arg_index         : integer;
<273>   g_arg_file_name     : t_string;
<274>   g_file_label_info   : t_file_label_info; { single file label info }
<275>   g_file_label_info_t : t_file_label_info; { total  file label info }
<276>   g_f_count_index     : integer;
<277>   g_f_count_index_a   : integer;
<278>   g_f_count_index_z   : integer;
<279>   g_total_file_sizes  : longint;
<280>   g_error             : integer;
<281>
<282> {$S SgAsmLabelInfo}
<283>
<284> { ------------------------------------------------------------------------ }
<285>
<286> procedure fetch_mac_error_message (    the_error     : integer;
<287>                                     var the_error_msg : t_string);
<288>
<289>   var
<290>     msg : string[99];
<291>
<292>   begin
<293>     if (the_error >= k_err_alpha) and (the_error <= k_err_omega) then
<294>       begin
<295>         case the_error of
<296>           k_err_not_text_file : msg := 'File is not a Macintosh text file.';
<297>           otherwise             msg := 'Unknown tool error (contact programmer).';
<298>         end; { case }
<299>
```

```
<300>         the_error_msg := msg;
<301>       end
<302>     else
<303>       begin
<304>         GetSysErrText (the_error,@the_error_msg);
<305>       end;
<306>   end;
<307>
<308> { ------------------------------------------------------------------------ }
<309>
<310> procedure write_error_message (the_error : integer);
<311>
<312>   var
<313>     err_message : t_string;
<314>
<315>   begin
<316>     fetch_mac_error_message(the_error,err_message);
<317>
<318>     writeln(err_message);
<319>   end;
<320>
<321> { ------------------------------------------------------------------------ }
<322>
<323> function file_is_text (the_file_name : t_string) : boolean;
<324>
<325>   var
<326>     is_text_file : boolean;
<327>     finder_info  : FInfo;
<328>     file_type    : string[4];
<329>     error        : integer;
<330>
<331>   begin
<332>     is_text_file := false;
<333>
<334>     error := GetFInfo(the_file_name,0,finder_info);
<335>
<336>     if error = 0 then
<337>       begin
<338>         file_type    := '????';
<339>         file_type[1] := finder_info.fdType[1];
<340>         file_type[2] := finder_info.fdType[2];
<341>         file_type[3] := finder_info.fdType[3];
<342>         file_type[4] := finder_info.fdType[4];
<343>
<344>         if g_debug_tool then
<345>           writeln('%%% FILE_IS_TEXT: fdType = "',file_type,'"');
<346>
<347>         if finder_info.fdType = 'TEXT' then
<348>           is_text_file := true;
<349>       end;
<350>
<351>     file_is_text := is_text_file;
<352>   end;
<353>
<354> { ------------------------------------------------------------------------ }
<355>
<356> function get_file_size (the_file_name : t_string) : longint;
<357>
<358>   var
<359>     f_size : longint;
<360>     f_ref  : integer;
<361>     error  : integer;
<362>
<363>   begin
<364>     f_size := 0;
```

```
<365>
<366>     error := FSOpen (the_file_name,0,f_ref);
<367>
<368>     if error = 0 then
<369>       begin
<370>         error := GetEOF  (f_ref,f_size);
<371>
<372>         if g_debug_tool and (error = 0) then
<373>           writeln('%%% GET_FILE_SIZE: f_size = ',f_size:0);
<374>
<375>         error := FSClose (f_ref);
<376>       end;
<377>
<378>     get_file_size := f_size;
<379>   end;
<380>
<381> { ------------------------------------------------------------------------ }
<382>
<383> procedure uppercase (var s : t_string);
<384>
<385>   var
<386>     i : integer;
<387>
<388>   begin
<389>     if length(s) > 0 then
<390>       for i := 1 to length(s) do
<391>         if (s[i] >= 'a') and (s[i] <= 'z') then
<392>           s[i] := chr( ord(s[i]) - ord('a') + ord('A') );
<393>   end;
<394>
<395> { ------------------------------------------------------------------------ }
<396>
<397> procedure inc_counter (var the_counter : t_counter);
<398>
<399>   begin
<400>     the_counter := the_counter + 1;
<401>   end;
<402>
<403> { ------------------------------------------------------------------------ }
<404>
<405> procedure fetch_label_from_line (    the_file_line    : t_string;
<406>                                  var the_label_phrase : t_string);
<407>
<408>   var
<409>     c_index : integer;
<410>     done    : boolean;
<411>
<412>   begin
<413>     the_label_phrase := '';
<414>
<415>     c_index := 1;
<416>     done    := false;
<417>
<418>     repeat
<419>       if c_index > length(the_file_line) then
<420>         done := true
<421>       else
<422>         begin
<423>           if the_file_line[c_index] <= chr(32) then
<424>             done := true
<425>           else
<426>             the_label_phrase := concat(the_label_phrase,
<427>                                        the_file_line[c_index]);
<428>         end;
<429>
```

```
<430>       c_index := c_index + 1;
<431>     until done;
<432>   end;
<433>
<434> { ---------------------------------------------------------------------------- }
<435>
<436> procedure process_asm_file (    the_file_name       : t_string;
<437>                                 the_file_number     : integer;
<438>                             var the_file_label_info : t_file_label_info;
<439>                             var the_error           : integer);
<440>
<441>   const
<442>     k_label_view_length = 20;
<443>     k_labels_per_line   = 3;
<444>
<445>   var
<446>     asm_file          : text;
<447>     asm_file_line     : t_string;
<448>     len_count_index   : integer;
<449>     label_phrase      : t_string;
<450>     label_length      : integer;
<451>     file_size         : longint;
<452>
<453>   begin
<454>     the_error := 0;
<455>
<456>     if g_debug_tool then
<457>       begin
<458>         writeln('%%% PROCESS_ASM_FILE: the_file_name   = "',the_file_name,'"');
<459>         writeln('%%% PROCESS_ASM_FILE: the_file_number = ',the_file_number:0);
<460>       end;
<461>
<462>     with the_file_label_info do
<463>       begin
<464>         i_file_lines   := 0;
<465>         i_label_count  := 0;
<466>         i_llabel_count := 0;
<467>         i_length_min   := maxint;
<468>         i_length_min_n := '';
<469>         i_length_max   := 0;
<470>         i_length_max_n := '';
<471>         i_length_total := 0;
<472>         i_length_avg   := 0;
<473>         i_length_std   := 0.0;
<474>         i_length_stdx2 := 0;
<475>         i_length_stdx  := 0;
<476>
<477>         for len_count_index := 1 to k_max_label_length do
<478>           i_length_list[len_count_index] := 0;
<479>       end;
<480>
<481>     writeln;
<482>     writeln(k_divider,k_divider);
<483>     writeln('* LABEL INFO FOR FILE : ',the_file_name);
<484>     writeln(k_divider,k_divider);
<485>
<486>     if file_is_text(the_file_name) then
<487>       begin
<488>         file_size:= get_file_size(the_file_name);
<489>
<490>         if file_size <= 0 then
<491>           begin
<492>             the_file_label_info.i_length_min := 0;
<493>
<494>             writeln('### WARNING: File "',the_file_name,'" contains no data');
```

```
<495>           end
<496>         else
<497>           begin
<498>             reset(asm_file,the_file_name); the_error := ioresult;
<499>
<500>             if g_debug_tool and (the_error <> 0) then
<501>               writeln('%%% PROCESS_ASM_FILE: RESET error = ',the_error:0);
<502>           end;
<503>       end
<504>     else
<505>       begin
<506>         the_error := k_err_not_text_file;
<507>       end;
<508>
<509>     if (the_error = 0) and (file_size > 0) then
<510>       begin
<511>         while not(eof(asm_file)) and (the_error = 0) do
<512>           begin
<513>             inc_counter(the_file_label_info.i_file_lines);
<514>             if the_file_label_info.i_file_lines mod 16 = 0 then
<515>               begin
<516>                 if odd(the_file_number) then
<517>                   SpinCursor(+1)
<518>                 else
<519>                   SpinCursor(-1);
<520>               end;
<521>
<522>             readln(asm_file,asm_file_line); the_error := ioresult;
<523>
<524>             if g_debug_tool and (the_error <> 0) then
<525>               writeln('%%% PROCESS_ASM_FILE: READLN error = ',the_error:0);
<526>
<527>             if the_error = 0 then
<528>               begin
<529>                 if length(asm_file_line) > 0 then
<530>                   begin
<531>                     if not(asm_file_line[1] in [';','*',chr(0)..chr(32)]) then
<532>                       begin
<533>                         with the_file_label_info do
<534>                           begin
<535>                             fetch_label_from_line (asm_file_line,label_phrase);
<536>
<537>                             if length(label_phrase) > k_max_label_length then
<538>                               label_phrase := copy(label_phrase,
<539>                                                    1,
<540>                                                    k_max_label_length);
<541>
<542>                             label_length := length(label_phrase);
<543>
<544>                             if label_phrase[1] = g_arg_local_label[1] then
<545>                               begin
<546>                                 { --- LOCAL LABEL --- }
<547>
<548>                                 inc_counter(i_llabel_count);
<549>                               end
<550>                             else
<551>                               begin
<552>                                 { --- REGULAR LABEL --- }
<553>
<554>                                 inc_counter(i_label_count);
<555>                                 inc_counter(i_length_list[label_length]);
<556>
<557>                                 i_length_total := i_length_total + label_length;
<558>
<559>                                 i_length_stdx2 :=
```

```
<560>                                 i_length_stdx2 + sqr(label_length);
<561>                               i_length_stdx  :=
<562>                                 i_length_stdx  +     label_length;
<563>
<564>                               if label_length < i_length_min then
<565>                                 begin
<566>                                   i_length_min   := label_length;
<567>                                   i_length_min_n := label_phrase;
<568>                                 end;
<569>
<570>                               if label_length > i_length_max then
<571>                                 begin
<572>                                   i_length_max   := label_length;
<573>                                   i_length_max_n := label_phrase;
<574>                                 end;
<575>
<576>                               if length(label_phrase) > k_label_view_length then
<577>                                 begin
<578>                                   label_phrase := copy(label_phrase,
<579>                                                        1,
<580>                                                        k_label_view_length);
<581>                                   label_phrase := concat(label_phrase,'?');
<582>                                 end;
<583>
<584>                               while length(label_phrase) < k_label_view_length do
<585>                                 label_phrase := concat(label_phrase,' ');
<586>
<587>                               write(i_file_lines:6,' ', label_phrase);
<588>                               the_error := ioresult;
<589>
<590>                               if g_debug_tool and (the_error <> 0) then
<591>                                 writeln('%%% PROCESS_ASM_FILE: WRITE error = ',
<592>                                                                the_error:0);
<593>
<594>                               if i_label_count mod k_labels_per_line = 0 then
<595>                                 writeln;
<596>                             end;
<597>                         end; { with the_file_label_info }
<598>                     end;
<599>                 end;
<600>             end;
<601>         end;
<602>
<603>       close(asm_file);
<604>
<605>       { make certain label list ends with a line feed }
<606>
<607>       if the_file_label_info.i_label_count mod k_labels_per_line <> 0 then
<608>         writeln;
<609>
<610>       { calculate label stats }
<611>
<612>       if g_debug_tool then
<613>         begin
<614>           with the_file_label_info do
<615>             begin
<616>               writeln('%%% PROCESS_ASM_FILE: i_length_total = ',i_length_total:0);
<617>               writeln('%%% PROCESS_ASM_FILE: i_label_count  = ',i_label_count:0);
<618>               writeln('%%% PROCESS_ASM_FILE: i_length_stdx2 = ',i_length_stdx2:0);
<619>               writeln('%%% PROCESS_ASM_FILE: i_length_stdx  = ',i_length_stdx:0);
<620>             end;
<621>         end;
<622>
<623>       with the_file_label_info do
<624>         begin
```

```
<625>             if i_length_total > 0 then
<626>               begin
<627>                 i_length_avg := i_length_total div i_label_count;
<628>
<629>                 i_length_std := sqrt( (i_length_stdx2 - sqr(i_length_stdx) /
<630>                                                           i_label_count)
<631>                                                          /
<632>                                                (i_label_count - 1) );
<633>               end;
<634>           end; { with the_file_label_info }
<635>       end;
<636>   end;
<637>
<638> { ---------------------------------------------------------------------------- }
<639>
<640> procedure get_current_date_time (var s : t_string);
<641>
<642>   var
<643>     mac_date_time_info : longint;
<644>     date_time_string   : str255;
<645>
<646>   begin
<647>     getdatetime  (mac_date_time_info);
<648>     iudatestring (mac_date_time_info,longdate,date_time_string);
<649>
<650>     s := date_time_string;
<651>
<652>     iutimestring (mac_date_time_info,true,date_time_string);
<653>
<654>     s := concat(s,'  ',date_time_string);
<655>   end;
<656>
<657> { ---------------------------------------------------------------------------- }
<658> {                              M   A   I   N                                 }
<659> { ---------------------------------------------------------------------------- }
<660>
<661> begin
<662>   PLSetVBuf     (output,nil,_iolbf,0); { mpw output buffer flushing control }
<663>   InitCursorCtl (nil);                 { mpw beach ball cursor }
<664>   InitErrMgr    ('','',false);         { mpw error message manager }
<665>
<666>   get_current_date_time(g_mac_date_time);
<667>
<668>   writeln;
<669>   writeln(k_pgm_title,'     Version: ',k_pgm_version);
<670>   writeln('Written by ',k_pgm_author,'  [',k_pgm_date,' ',k_pgm_time,']');
<671>   writeln(k_pgm_address);
<672>   writeln(k_pgm_copyright);
<673>   writeln;
<674>   writeln('Current Date and Time:  ',g_mac_date_time);
<675>   writeln;
<676>
<677>   g_arg_tool_name := argv^[0]^;
<678>   g_debug_tool    := false;
<679>
<680>   if argc < 3 then
<681>     begin
<682>       writeln('### ERROR : Tool argument list is wrong');
<683>       writeln('### SYNTAX: ',g_arg_tool_name,' local-label-char  asm-text-file-list');
<684>       writeln('###         where local-label-char is a character that exists at');
<685>       writeln('            the beginning of each local label (eg: "@"),');
<686>       writeln('            asm-text-file-list is a list of assembly language');
<687>       writeln('            file names (eg: FooBar.a Frodor.a)');
<688>       writeln;
<689>       writeln('            if local-label-char contains the phrase "DEBUG" then');
```

```
<690>       writeln('            special debugging information is outputted also');
<691>     end
<692>   else
<693>     begin
<694>       with g_file_label_info_t do  { label info for all the files }
<695>         begin
<696>           i_file_lines   := 0;
<697>           i_label_count  := 0;
<698>           i_llabel_count := 0;
<699>           i_length_min   := maxint;
<700>           i_length_min_n := '';
<701>           i_length_max   := 0;
<702>           i_length_max_n := '';
<703>           i_length_total := 0;
<704>           i_length_avg   := 0;
<705>           i_length_std   := 0.0;
<706>           i_length_stdx2 := 0;
<707>           i_length_stdx  := 0;
<708>
<709>           for g_f_count_index := 1 to k_max_label_length do
<710>             i_length_list[g_f_count_index] := 0;
<711>         end;
<712>
<713>       g_arg_local_label := argv^[1]^;
<714>       uppercase(g_arg_local_label);
<715>
<716>       if pos(k_debug_phrase,g_arg_local_label) > 0 then
<717>         begin
<718>           g_debug_tool := true;
<719>
<720>           delete(g_arg_local_label,
<721>                  pos(k_debug_phrase,g_arg_local_label),
<722>                  length(k_debug_phrase));
<723>
<724>           writeln('%%% local label arg string contains debug flag phrase');
<725>           writeln('%%% g_arg_local_label (after delete) = "',g_arg_local_label,'"');
<726>         end;
<727>
<728>       if g_debug_tool then
<729>         writeln('%%% g_arg_local_label = "',g_arg_local_label,'"    ',
<730>                 'length = ',length(g_arg_local_label):0);
<731>
<732>       if length(g_arg_local_label) = 0 then
<733>         begin
<734>           g_arg_local_label    := '?';
<735>           g_arg_local_label[1] := chr(255);
<736>         end;
<737>
<738>       if g_arg_local_label[1] <= ' ' then
<739>         g_arg_local_label := chr(255);
<740>
<741>       if g_debug_tool then
<742>         writeln('%%% argc = ',argc:0);
<743>
<744>       g_arg_index := 1;
<745>
<746>       while (g_arg_index < (argc - 1)) do
<747>         begin
<748>           g_arg_index     := g_arg_index + 1;
<749>           g_arg_file_name := argv^[g_arg_index]^;
<750>
<751>           if g_debug_tool then
<752>             writeln('%%% file arg # ',g_arg_index:3,' is "',g_arg_file_name,'"');
<753>
<754>           process_asm_file (g_arg_file_name, g_arg_index - 1, g_file_label_info, g_error);
```

```
<755>
<756>           if g_error <> 0 then
<757>             begin
<758>               writeln('### ERROR ',g_error:0,
<759>                       ' while processing file "',g_arg_file_name,'"');
<760>               write('          ');
<761>               write_error_message(g_error);
<762>             end
<763>           else
<764>             begin
<765>               writeln(k_divider,k_divider);
<766>               writeln('* FILE LABEL STATISTICS');
<767>               writeln(k_divider,k_divider);
<768>
<769>               with g_file_label_info do
<770>                 begin
<771>                   writeln('File Lines           = ',i_file_lines:8);
<772>                   writeln('Label Count          = ',i_label_count:8);
<773>                   writeln('Local Label Count    = ',i_llabel_count:8);
<774>                   writeln('Min     Label Length = ',i_length_min:8,' ',
<775>                                                     i_length_min_n);
<776>                   writeln('Max     Label Length = ',i_length_max:8,' ',
<777>                                                     i_length_max_n);
<778>                   writeln('Avg     Label Length = ',i_length_avg:8);
<779>                   writeln('Std Dev Label Length = ',i_length_std:8:2);
<780>
<781>                   writeln;
<782>                   writeln('Label Length Frequency Counts:');
<783>
<784>                   g_f_count_index_a := 0;
<785>                   g_f_count_index_z := 0;
<786>
<787>                   for g_f_count_index := 1 to k_max_label_length do
<788>                     if g_f_count_index_a = 0 then
<789>                       if i_length_list[g_f_count_index] <> 0 then
<790>                         g_f_count_index_a := g_f_count_index;
<791>
<792>                   for g_f_count_index := k_max_label_length downto 1 do
<793>                     if g_f_count_index_z = 0 then
<794>                       if i_length_list[g_f_count_index] <> 0 then
<795>                         g_f_count_index_z := g_f_count_index;
<796>
<797>                   if (g_f_count_index_a = 0) or (g_f_count_index_z = 0) then
<798>                     writeln('  There were no labels to count for this file')
<799>                   else
<800>                     for g_f_count_index := g_f_count_index_a to g_f_count_index_z do
<801>                       writeln('  No. Labels with Length ',g_f_count_index:3,': ',
<802>                                 i_length_list[g_f_count_index]:8);
<803>                 end; { with g_file_label_info }
<804>
<805>               writeln(k_divider,k_divider);
<806>               writeln('* END OF FILE: ',g_arg_file_name);
<807>               writeln(k_divider,k_divider);
<808>               writeln;
<809>
<810>               { accumulate total file label info }
<811>
<812>               with g_file_label_info_t do
<813>                 begin
<814>                   i_file_lines   := i_file_lines   + g_file_label_info.i_file_lines;
<815>                   i_label_count  := i_label_count  + g_file_label_info.i_label_count;
<816>                   i_llabel_count := i_llabel_count + g_file_label_info.i_llabel_count;
<817>                   i_length_total := i_length_total + g_file_label_info.i_length_total;
<818>
<819>                   if g_file_label_info.i_length_min < i_length_min then
```

```
<820>                   begin
<821>                     i_length_min   := g_file_label_info.i_length_min;
<822>                     i_length_min_n := g_file_label_info.i_length_min_n;
<823>                   end;
<824>
<825>                 if g_file_label_info.i_length_max > i_length_max then
<826>                   begin
<827>                     i_length_max   := g_file_label_info.i_length_max;
<828>                     i_length_max_n := g_file_label_info.i_length_max_n;
<829>                   end;
<830>
<831>                 i_length_stdx2 := i_length_stdx2 + g_file_label_info.i_length_stdx2;
<832>                 i_length_stdx  := i_length_stdx  + g_file_label_info.i_length_stdx;
<833>
<834>                 for g_f_count_index := 1 to k_max_label_length do
<835>                   i_length_list[g_f_count_index] := i_length_list[g_f_count_index] +
<836>                        g_file_label_info.i_length_list[g_f_count_index];
<837>               end; { with g_file_label_info_t }
<838>           end;
<839>         end; { while }
<840>
<841>         if g_error = 0 then
<842>           begin
<843>             { calculate final stats }
<844>
<845>             if g_debug_tool then
<846>               begin
<847>                 with g_file_label_info_t do
<848>                   begin
<849>                     writeln('%%% (total) i_length_total = ',i_length_total:0);
<850>                     writeln('%%% (total) i_label_count  = ',i_label_count:0);
<851>                     writeln('%%% (total) i_length_stdx2 = ',i_length_stdx2:0);
<852>                     writeln('%%% (total) i_length_stdx  = ',i_length_stdx:0);
<853>                   end;
<854>               end;
<855>
<856>             with g_file_label_info_t do
<857>               begin
<858>                 i_length_avg := i_length_total div i_label_count;
<859>
<860>                 i_length_std := sqrt( (i_length_stdx2 - sqr(i_length_stdx) /
<861>                                                         i_label_count)
<862>                                                        /
<863>                                                (i_label_count - 1) );
<864>               end; { with g_file_label_info_t }
<865>
<866>             { show final stats and list of file names }
<867>
<868>             writeln(k_divider,k_divider);
<869>             writeln('* SUMMARY FILE LABEL INFO:');
<870>             writeln(k_divider,k_divider);
<871>
<872>             writeln('File Name List:');
<873>
<874>             g_total_file_sizes := 0;
<875>
<876>             for g_f_count_index := 2 to (argc - 1) do
<877>               begin
<878>                 g_arg_file_name:= argv^[g_f_count_index]^;
<879>
<880>                 write('  [',(g_f_count_index-1):3,']  ');
<881>
<882>                 write('Text File? ');
<883>                 if file_is_text(g_arg_file_name) then
<884>                   write('Yes     ')
```

```
<885>                 else
<886>                   write('No      ');
<887>
<888>                 write('Size: ',get_file_size(g_arg_file_name):8,' bytes - ');
<889>
<890>                 g_total_file_sizes := g_total_file_sizes +
<891>                                       get_file_size(g_arg_file_name);
<892>
<893>                 writeln('"',g_arg_file_name,'"');
<894>               end;
<895>
<896>             writeln('                              --------------');
<897>             writeln('                                    ',g_total_file_sizes:8);
<898>
<899>             with g_file_label_info_t do
<900>               begin
<901>                 writeln;
<902>                 writeln('All File Label Statistics:');
<903>                 writeln('  File Lines           = ',i_file_lines:8);
<904>                 writeln('  Label Count          = ',i_label_count:8);
<905>                 writeln('  Local Label Count    = ',i_llabel_count:8);
<906>                 writeln('  Min     Label Length = ',i_length_min:8,' ',
<907>                                                     i_length_min_n);
<908>                 writeln('  Max     Label Length = ',i_length_max:8,' ',
<909>                                                     i_length_max_n);
<910>                 writeln('  Avg     Label Length = ',i_length_avg:8);
<911>                 writeln('  Std Dev Label Length = ',i_length_std:8:2);
<912>
<913>                 writeln;
<914>                 writeln('Label Length Frequency Counts:');
<915>
<916>                 g_f_count_index_a := 0;
<917>                 g_f_count_index_z := 0;
<918>
<919>                 for g_f_count_index := 1 to k_max_label_length do
<920>                   if g_f_count_index_a = 0 then
<921>                     if i_length_list[g_f_count_index] <> 0 then
<922>                       g_f_count_index_a := g_f_count_index;
<923>
<924>                 for g_f_count_index := k_max_label_length downto 1 do
<925>                   if g_f_count_index_z = 0 then
<926>                     if i_length_list[g_f_count_index] <> 0 then
<927>                       g_f_count_index_z := g_f_count_index;
<928>
<929>                 if (g_f_count_index_a = 0) or (g_f_count_index_z = 0) then
<930>                   writeln('  There were no labels to count for this file')
<931>                 else
<932>                   for g_f_count_index := g_f_count_index_a to g_f_count_index_z do
<933>                     writeln('  No. Labels with Length ',g_f_count_index:3,': ',
<934>                                 i_length_list[g_f_count_index]:8);
<935>               end; { with g_file_label_info_t }
<936>
<937>             writeln(k_divider,k_divider);
<938>             writeln('* END OF SUMMARY FILE LABEL INFO');
<939>             writeln(k_divider,k_divider);
<940>           end;
<941>     end;
<942>
<943>   writeln;
<944>   writeln('That''s all Folks !');
<945> end.
<946>
<947> { FINIS }
```

For those with a curiosity about the Lisa's 68000 assembler here's the assembler's progress information for the assembly of the Boot ROM sources (this assembler ran in the Lisa Workshop environment and was called either the Lisa Assembler or the TLA Assember [TLA stood for The Last Assembler and was used for this assembler since it was designed to use external definition data files containing opcode information which allowed the assembler to process different assembly languages depending upon the contents of the external definition data files]):

```
<  1> ################################################################################
<  2> #                                                                              #
<  3> #     APPLE LISA BOOT ROM 2.48 SOURCE CODE ASSEMBLER PROGRESS INFORMATION      #
<  4> #                                                                              #
<  5> ################################################################################
<  6>
<  7>
<  8> ASSEMBLER - MC68000 (Ver 3.77)    02-May-85
<  9> (C) 1984 Apple Computer Inc.
< 10>
< 11>
< 12> Options:     Meaning                    Current Value.
< 13> +P         Pretty Listing                   FALSE
< 14> +S         Print Space Avail                FALSE
< 15> +E         Code patching efficiency check   FALSE
< 16> ?          Print options
< 17> <ret>      Accept options
< 18>
< 19> Input file ? [.TEXT] ?
< 20>
< 21> options : +P
< 22> options :
< 23>
< 24> Input file ? [.TEXT] RM248.E
< 25> Listing file (<CR> for none) ? [.TEXT] -PROFILE-RM248.LIST
< 26> Output file ? [RM248.E] [.OBJ] RM248
< 27>
< 28> <    0>...............................................
< 29> <  500>...............................................
< 30> < 1000>...............................................
< 31> < 1500>................
< 32>         .INCLUDE RM248.K.TEXT
< 33>
< 34> < 1667>.
< 35>         .PROC   LISAROM,0
< 36>
< 37> < 1674>.............................
< 38> < 2000>...............................................
< 39> < 2500>...............................................
< 40> < 3000>...............................................
< 41> < 3500>......
< 42>         .INCLUDE RM248.S.TEXT
< 43>
< 44> < 3567>.......................................
< 45> < 4000>...............................................
< 46> < 4500>...............................................
< 47> < 5000>...............................................
< 48> < 5500>.....
< 49>         .INCLUDE RM248.B.TEXT
< 50>
< 51> < 5551>.......................................
< 52> < 6000>...............................................
< 53> < 6500>...............................................
< 54> < 7000>...............................................
< 55> < 7500>................
< 56>         .INCLUDE RM248.M.TEXT
```

```
< 57>
< 58> < 7667>.................................
< 59> < 8000>................................................
< 60> < 8500>................................................
< 61> < 9000>................................................
< 62> < 9500>..........
< 63>         .INCLUDE RM248.G.TEXT
< 64>
< 65> < 9602>.....................................
< 66> <10000>................................................
< 67> <10500>................................................
< 68> <11000>................................................
< 69> <11500>.................................
< 70>
< 71>   Assembly complete:     11840 lines
< 72>       0   Warnings
< 73>       0   Errors
< 74>
< 75> MC68000 Assembly Pretty Listing  (Ver 3.77)
< 76>
< 77> pass one - getting patches
< 78> <   0>   ................................................
< 79> < 100>   ................................................
< 80> < 200>   ................................................
< 81> < 300>   ................................................
< 82> < 400>   ................................................
< 83> < 500>   ................................................
< 84> < 600>   ................................................
< 85> < 700>   ................................................
< 86> < 800>   ................................................
< 87> < 900>   ................................................
< 88> <1000>   ................................................
< 89> <1100>   ................................................
< 90> <1200>   ................................................
< 91> <1300>   ................................................
< 92> <1400>   ................................................
< 93> <1500>   ................................................
< 94> <1600>   ................................................
< 95> <1700>   ................................................
< 96> <1800>   ........................
< 97> pass two - making updates
< 98> <   0>   ................................................
< 99> < 100>   ................................................
<100> < 200>   ................................................
<101> < 300>   ................................................
<102> < 400>   ................................................
<103> < 500>   ................................................
<104> < 600>   ................................................
<105> < 700>   ................................................
<106> < 800>   ................................................
<107> < 900>   ................................................
<108> <1000>   ................................................
<109> <1100>   ................................................
<110> <1200>   ................................................
<111> <1300>   ................................................
<112> <1400>   ................................................
<113> <1500>   ................................................
<114> <1600>   ................................................
<115> <1700>   ................................................
<116> <1800>   ........................
<117>
<118> ASSEMBLY COMPLETE.
```

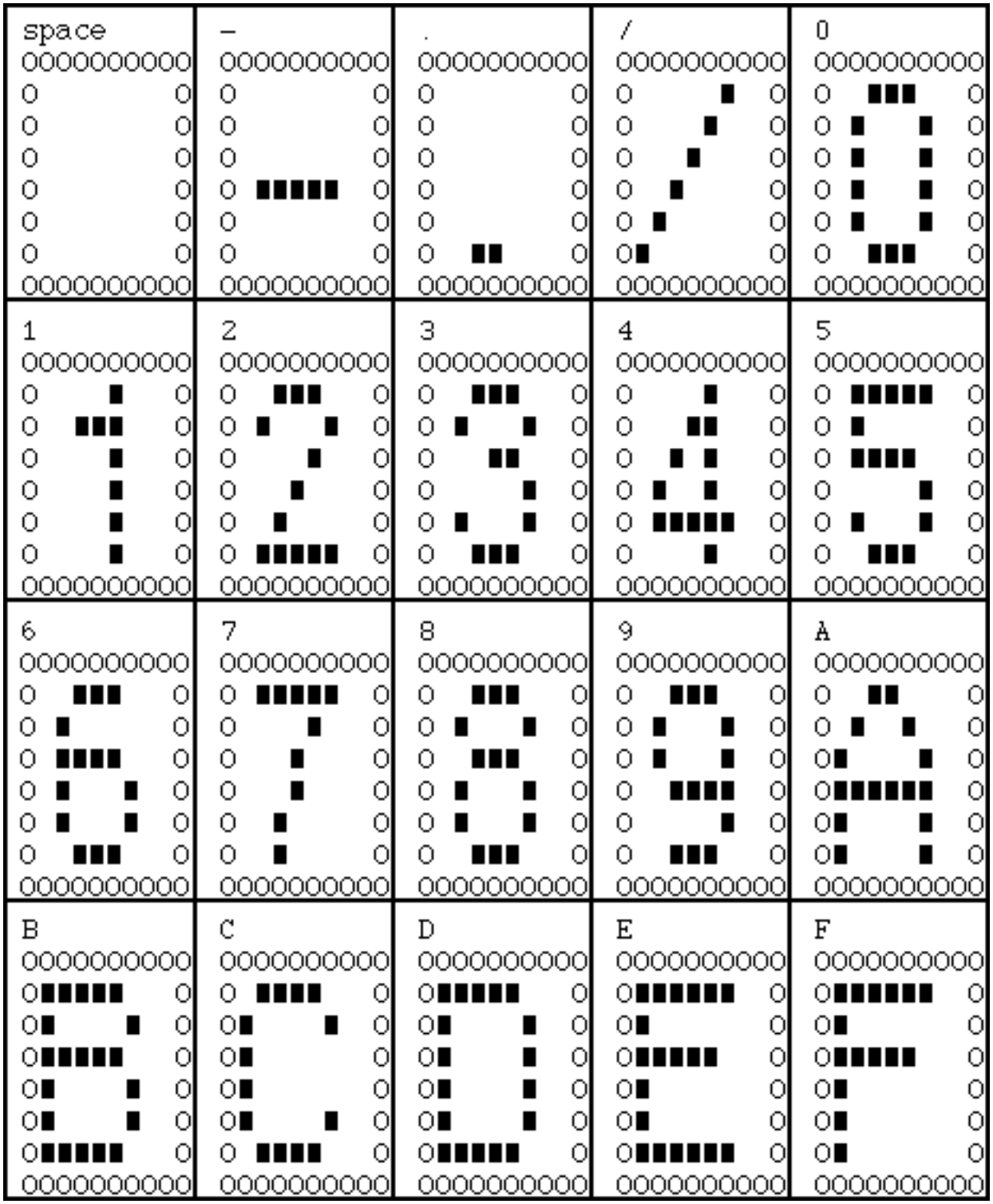
X.7 BOOT ROM CHARACTER FONT

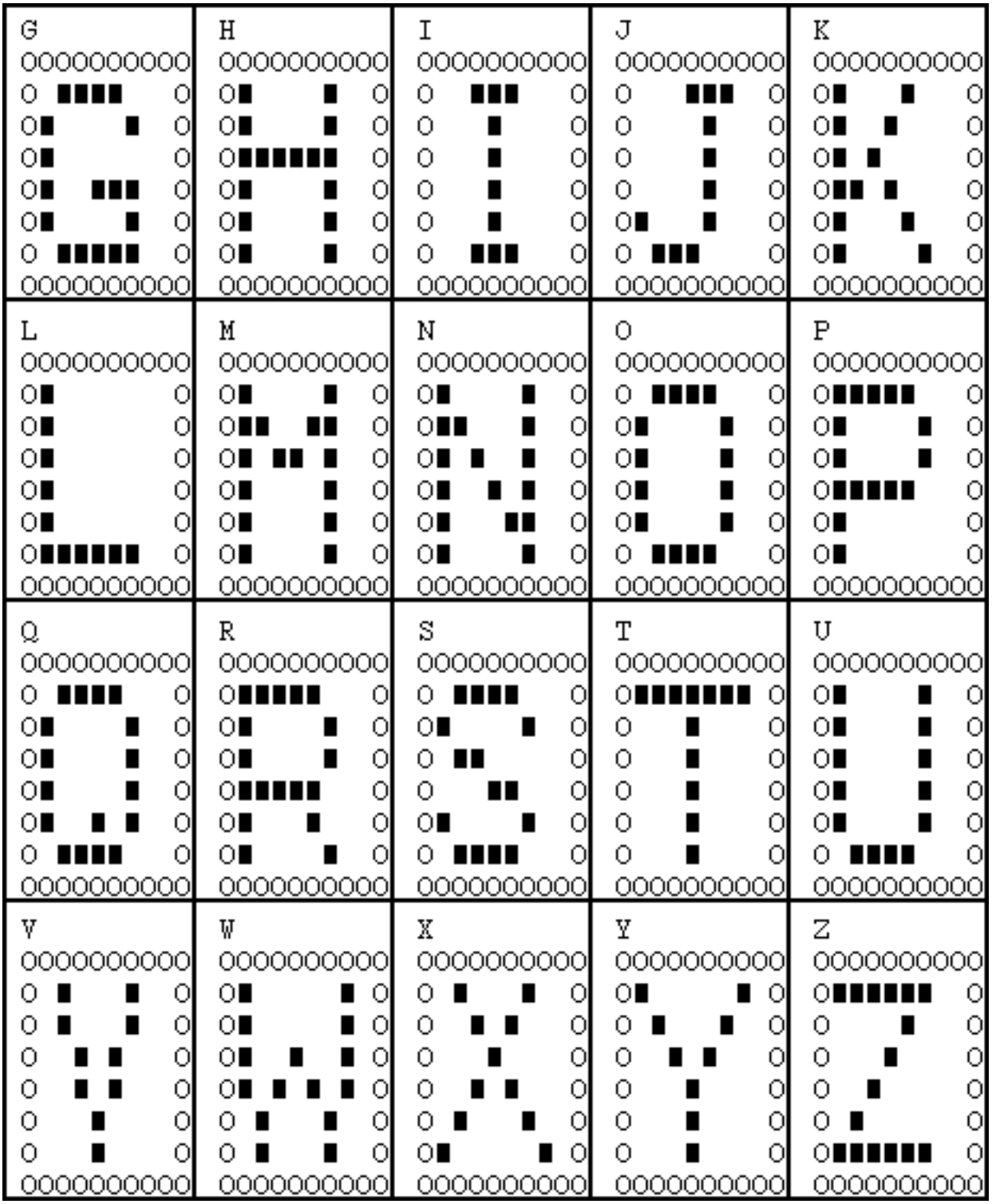
The boot ROM supported a character font which the ROM used for its menus, icons, and Service Mode. This font supported only a limited number of characters, which follow:

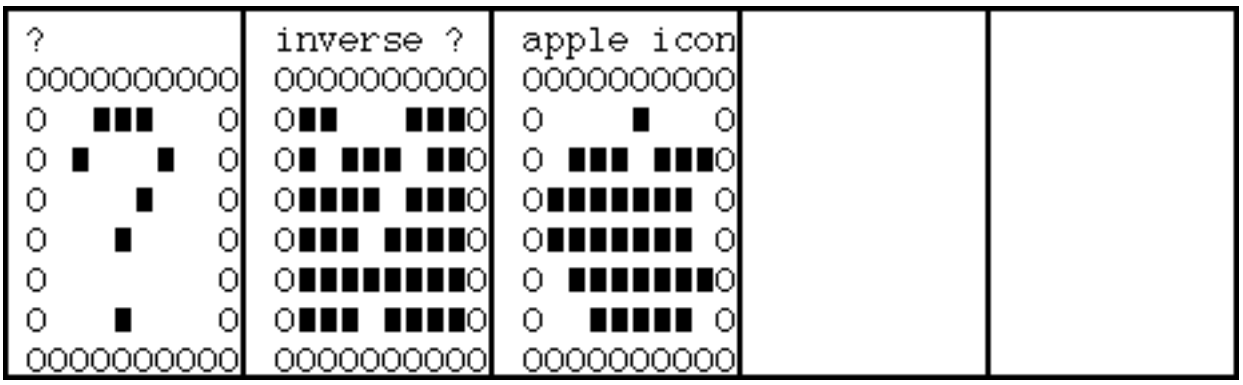
(space) . - / 0123456789 ABCDEFGHIJKLMNOP ? (inverse ?) (Apple symbol)

Note that no lowercase characters existed in this font, and only a few special characters (e.g. /) appeared.

Complete bitmaps of these characters follow. These were produced by David Craig's program DTCLisaBootROMFonts whose source code appears at the end of this section.


space
–
.
/
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F


G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z



The Macintosh MPW Pascal source code for the program DTCLisaBootROMFonts follows:

```
00001  {
00002     APPLE LISA COMPUTER BOOT ROM FONT TABLE DUMPER
00003
00004     THIS MACINTOSH MPW SHELL TOOL WRITES THE DATA CONTAINED IN THE APPLE
00005     LISA COMPUTER'S BOOT ROM FONT TABLE TO THE ACTIVE WINDOW AS TEXTUAL
00006     DATA.  THE LISA FONT TABLE DATA IS EMBEDDED WITHIN THIS TOOL.
00007
00008     WRITTEN BY DAVID T. CRAIG FOR HIS LISA LEGACY REVISION PAPER
00009     04 SEPTEMBER 1994
00010  }
00011
00012  program lisa_boot_rom_font_bitmaps;
00013
00014  const k_pgm_name     = 'Lisa Computer Boot ROM Font Bitmap Drawer';
00015        k_pgm_author   = 'David T. Craig';
00016        k_pgm_date     = '04 September 1994';
00017        k_font_count   = 43;
00018
00019  type  t_font_description = string[31];
00020        t_font_bitmaps     = array [1..k_font_count] of t_font_description;
00021                             { chars 1-12 = bitmaps, 14+ = char name }
00022        t_font_bitmap_data = array [1..6] of char;
00023        t_hex              = string[2];
00024        t_bit_image        = string[8];
00025
00026  var   v_font_bitmaps      : t_font_bitmaps;
00027        v_font_bitmap_index : 1..k_font_count;
00028        v_font_bitmap_data  : t_font_bitmap_data;
00029        v_font_char         : t_font_description;
00030        v_bit_image         : t_bit_image;
00031        v_bit_image_i       : 1..6;
00032
00033  { ____________________________________________________________________________ }
00034
00035  { initialize the boot rom font bitmaps, this data is from the rom
00036    source file "RM248.G.TEXT" and the table titled "AsciiTable" }
00037
00038  procedure init_font_bitmaps;
00039
00040  begin
00041    v_font_bitmaps[ 1] := '000000000000 space';
00042    v_font_bitmaps[ 2] := '0000007C0000 -';
00043    v_font_bitmaps[ 3] := '000000000030 .';
00044    v_font_bitmaps[ 4] := '040810204080 /';
00045    v_font_bitmaps[ 5] := '384444444438 0';
00046    v_font_bitmaps[ 6] := '083808080808 1';
```

```
00047    v_font_bitmaps[ 7] := '38440810207C 2';
00048    v_font_bitmaps[ 8] := '384418044438 3';
00049    v_font_bitmaps[ 9] := '081828487C08 4';
00050    v_font_bitmaps[10] := '7C4078044438 5';
00051    v_font_bitmaps[11] := '384078444438 6';
00052    v_font_bitmaps[12] := '7C0810102020 7';
00053    v_font_bitmaps[13] := '384438444438 8';
00054    v_font_bitmaps[14] := '3844443C0438 9';
00055    v_font_bitmaps[15] := '304884FC8484 A';
00056    v_font_bitmaps[16] := 'F884F88484F8 B';
00057    v_font_bitmaps[17] := '788480808478 C';
00058    v_font_bitmaps[18] := 'F884848484F8 D';
00059    v_font_bitmaps[19] := 'FC80F88080FC E';
00060    v_font_bitmaps[20] := 'FC80F8808080 F';
00061    v_font_bitmaps[21] := '7884809C847C G';
00062    v_font_bitmaps[22] := '8484FC848484 H';
00063    v_font_bitmaps[23] := '381010101038 I';
00064    v_font_bitmaps[24] := '1C0808088870 J';
00065    v_font_bitmaps[25] := '8890A0D08884 K';
00066    v_font_bitmaps[26] := '8080808080FC L';
00067    v_font_bitmaps[27] := '84CCB4848484 M';
00068    v_font_bitmaps[28] := '84C4A4948C84 N';
00069    v_font_bitmaps[29] := '788484848478 O';
00070    v_font_bitmaps[30] := 'F88484F88080 P';
00071    v_font_bitmaps[31] := '788484849478 Q';
00072    v_font_bitmaps[32] := 'F88484F88884 R';
00073    v_font_bitmaps[33] := '788460188478 S';
00074    v_font_bitmaps[34] := 'FE1010101010 T';
00075    v_font_bitmaps[35] := '848484848478 U';
00076    v_font_bitmaps[36] := '444428281010 V';
00077    v_font_bitmaps[37] := '828292AA4444 W';
00078    v_font_bitmaps[38] := '442810284482 X';
00079    v_font_bitmaps[39] := '824428101010 Y';
00080    v_font_bitmaps[40] := 'FC08102040FC Z';
00081    v_font_bitmaps[41] := '384408100010 ?';
00082    v_font_bitmaps[42] := 'C7BBF7EFFFEF inverse ?';
00083    v_font_bitmaps[43] := '0877FEFE7F3E apple icon';
00084  end;
00085
00086  { ____________________________________________________________________________ }
00087
00088  function cvt_hex2byte (h : t_hex) : char; { eg: '1C' -- > chr(28) }
00089
00090  type t_nibble = 0..15;
00091  var  msn, lsn : t_nibble;
00092
00093  function digit2nibble (d : char) : t_nibble;
00094
00095  var n : t_nibble;
00096
00097  begin
00098    case d of
00099      '0' : n :=  0;      '1' : n :=  1;      '2' : n :=  2;      '3' : n :=  3;
00100     '4' : n :=  4;      '5' : n :=  5;      '6' : n :=  6;      '7' : n :=  7;
00101     '8' : n :=  8;      '9' : n :=  9;      'A' : n := 10;      'B' : n := 11;
00102     'C' : n := 12;      'D' : n := 13;      'E' : n := 14;      'F' : n := 15;
00103    end;
00104    digit2nibble := n;
00105  end;
00106
00107  begin
00108    msn := digit2nibble(h[1]);
00109    lsn := digit2nibble(h[2]);
00110    cvt_hex2byte := chr( (msn * 16) + lsn );
00111  end;
```

```
00112
00113  { ____________________________________________________________________________ }
00114
00115  procedure convert_bitmap_to_data (    f_desc : t_font_description;
00116                                   var f_data : t_font_bitmap_data);
00117
00118  begin
00119    f_data[1] := cvt_hex2byte (copy(f_desc, 1,2));
00120    f_data[2] := cvt_hex2byte (copy(f_desc, 3,2));
00121    f_data[3] := cvt_hex2byte (copy(f_desc, 5,2));
00122    f_data[4] := cvt_hex2byte (copy(f_desc, 7,2));
00123    f_data[5] := cvt_hex2byte (copy(f_desc, 9,2));
00124    f_data[6] := cvt_hex2byte (copy(f_desc,11,2));
00125  end;
00126
00127  { ____________________________________________________________________________ }
00128
00129  procedure get_bit_image (bite : char; var bit_image : t_bit_image);
00130
00131  var l : longint;
00132      i : 0..7;
00133
00134  begin
00135    bit_image := concat(chr(127),chr(127),chr(127),chr(127),
00136                        chr(127),chr(127),chr(127),chr(127));
00137                        { should appear as an empty square }
00138
00139    l := ord(bite);
00140
00141    for i := 0 to 7 do begin
00142      if BTST(l,i) = false then
00143        bit_image[8-i] := ' ';
00144    end;
00145  end;
00146
00147  { ____________________________________________________________________________ }
00148  { ____________________________________________________________________________ }
00149  { ____________________________________________________________________________ }
00150
00151  begin
00152    writeln(k_pgm_name);
00153    writeln(k_pgm_author,' ',k_pgm_date);
00154
00155    init_font_bitmaps;
00156
00157    for v_font_bitmap_index := 1 to k_font_count do begin
00158      v_font_char := v_font_bitmaps[v_font_bitmap_index];
00159      convert_bitmap_to_data(v_font_char,v_font_bitmap_data);
00160
00161      writeln(copy(v_font_char,14,length(v_font_char)-13));
00162
00163      writeln('OOOOOOOOOO');
00164      for v_bit_image_i := 1 to 6 do begin
00165        get_bit_image(v_font_bitmap_data[v_bit_image_i],v_bit_image);
00166        writeln('O',v_bit_image,'O');
00167      end;
00168      writeln('OOOOOOOOOO');
00169    end;
00170
00171    writeln('That''s all folks!');
00172  end.
```

X.8 BOOT ROM FOREIGN PHRASES

The Boot ROM supported the display of phrases in the following languages: English, French, or German.

The language phrases that appeared were based upon the type of keyboard connected to the Lisa. All Lisa keyboards were self-identifiying. As such, a Lisa user could attach for example a German Lisa keyboard and the Boot ROM phrases would appear in German. English phrases appeared for US, UK, and Canadian keyboards. If a keyboard type was not US, UK, Canadian, German, or French, then the Boot ROM displayed its phrases in English, German, and French (the Hardware Interface appendix in the Lisa Language Workshop manual listed the following keyboards that could be attached to the Lisa: US, Swiss-German, Swiss-French, Portuguese, Spanish-Latin America, Danish, Swedish, Italian, French, German, UK, French-Canadian, US-Dvorak, and APL).

Service Mode phrases were not translated since this mode was ment for use only by Apple manufacturing and repair people who would most likely be English speakers (these people tended to reside in either Cupertino California or Carrollton Texas, Apple's main Lisa manufacturing facility).

The English, French, and German language phrases existed in several tables:

```
<  1> CHKMSG  .ASCII  'TESTING'
<  2>         .BYTE   0
<  3>         .ASCII  'TEST'                  ;French translation
<  4>         .BYTE   0
<  5>         .ASCII  'ES WIRD GETESTET'      ;German translation
<  6>         .BYTE   0
<  7>
<  8> RTRYMSG .ASCII  'RESTART'
<  9>         .BYTE   0
< 10>         .ASCII  'RECOMMENCER'           ;French
< 11>         .BYTE   0
< 12>         .ASCII  'NEU STARTEN'           ;German
< 13>         .BYTE   0
< 14>
< 15> CONTMSG .ASCII  'CONTINUE'
< 16>         .BYTE   0
< 17>         .ASCII  'CONTINUER'             ;French
< 18>         .BYTE   0
< 19>         .ASCII  'WEITERMACHEN'          ;German
< 20>         .BYTE   0
< 21>
< 22> STRTMSG .ASCII  'STARTUP FROM'
< 23>         .BYTE   0
< 24>         .ASCII  'DEMARRER DE'           ;French
< 25>         .BYTE   0
< 26>         .ASCII  'STARTEN VON'           ;German
< 27>         .BYTE   0
```

The Boot ROM routine (DSPSTRING, "Display String") that displayed all language messages follows (note the reference to Italian and Spanish keyboards):

```
<  1> ;---------------------------------------------------------------------
<  2> ;  Subroutine to display text string according to keyboard id
<  3> ;  Inputs:
<  4> ;       A3 = ptr to message
<  5> ;       D1 = nonzero if '...' string to be appended
<  6> ;  Outputs:
<  7> ;       A2 = ptr to start of string
<  8> ;       A3 = ptr to end of string
<  9> ;  Side Effects:
< 10> ;       D5-D6, A3 trashed
< 11> ;---------------------------------------------------------------------
```

```
< 12>
< 13> DSPSTRING
< 14>         MOVEM.L D0/D2,-(SP)     ;save regs
< 15>         LEA     MENUHDG,A2      ;don't translate service mode messages
< 16>         CMPA.L  A2,A3
< 17>         BEQ.S   DSPOUT          ;skip if it is
< 18>         MOVE.L  A3,A2           ;else save starting point
< 19>         MOVE.B  KEYID,D0        ;get keyboard id
< 20>         BEQ.S   DSPOUT          ;skip if no id available
< 21>         ANDI.B  #$3F,D0         ;clear mfg code
< 22>         MOVE.B  D0,D2           ;move to working reg
< 23>
< 24> ;  Search for US, UK or Canadian keyboard
< 25>
< 26>         ANDI.B  #$F0,D2         ;old US keyboard?
< 27>         BEQ.S   DSPOUT          ;yes - go do English display
< 28>         CMPI.B  #$30,D2         ;US or Canadian layout?
< 29>         BNE.S   @1
< 30>         CMPI.B  #$3D,D0         ;Canadian?
< 31>         BEQ.S   DSPALL          ;yes - display all languages
< 32>         BRA.S   DSPOUT          ;else just English
< 33>
< 34> @1      CMPI.B  #$20,D2         ;European keyboard?
< 35>         BNE.S   DSPALL          ;no - display all languages
< 36>         CMPI.B  #$2F,D0         ;UK?
< 37>         BEQ.S   DSPOUT          ;yes - display English
< 38>
< 39> ;  Search for German type keyboard
< 40>
< 41>         CMPI.B  #$2E,D0         ;German?
< 42>         BEQ.S   DSPGERMN
< 43>         CMPI.B  #$26,D0         ;Swiss-German?
< 44>         BEQ.S   DSPGERMN
< 45>
< 46> ;  Search for French type keyboard
< 47>
< 48>         CMPI.B  #$2D,D0         ;French?
< 49>         BEQ.S   DSPFRNCH
< 50>         CMPI.B  #$27,D0         ;Swiss-French?
< 51>         BEQ.S   DSPFRNCH
< 52>
< 53> ;  Display all languages for any other keyboard (e.g., Italian, Spanish, etc.)
< 54>
< 55> DSPALL  BTST    #MENU,STATFLGS  ;doing menu?
< 56>         BNE.S   @1              ;skip if yes
< 57>         SUB     #CHRSPC,D5      ;back up one row
< 58>         BSR.S   DSPIT           ;display English string
< 59>         ADD     #CHRSPC,D5      ;incr to next row
< 60>         BSR.S   DSPIT           ;display French translation
< 61>         ADD     #CHRSPC,D5      ;bump another row
< 62>         BRA.S   DSPOUT          ;go do final display of German
< 63>
< 64> @1      BSR.S   DSPMSLSH        ;display English followed by /
< 65>         BSR.S   DSPMSLSH        ;display French followed by /
< 66>         BRA.S   DSPOUT          ;and go do final German display
< 67>
< 68> DSPGERMN
< 69>         TST.B   (A3)+           ;skip two strings before output
< 70>         BNE.S   DSPGERMN
< 71>
< 72> DSPFRNCH
< 73>         TST.B   (A3)+           ;skip one string before output
< 74>         BNE.S   DSPFRNCH
< 75>         MOVE.L  A3,A2           ;save new beginning ptr
< 76>
```

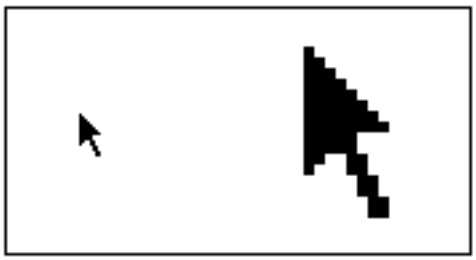
```
< 77> DSPOUT  BSR.S   DSPIT           ;do display
< 78>         MOVEM.L (SP)+,D0/D2     ;restore regs
< 79>         RTS                     ; and exit
```

X.9 BOOT ROM ICONS

xxxxxx

X.10 BOOT ROM MOUSE CURSOR BITMAP

The Boot ROM supported the mouse for menu command selections. The mouse had a pointer graphic associated with it which had the following bitmap image (the image on the left is the actual size, the image on the right is magnified for clarity):



The bitmap definition for the mouse cursor from the Boot ROM source code follows:

```
<  1> CrsrData                                          ;arrow for mouse cursor
<  2> CrsrMask                                          ;same for mask
<  3>          .WORD   $8000,$C000,$E000,$F000
<  4>          .WORD   $F800,$FC00,$FE00,$FF00
<  5>          .WORD   $F800,$F800,$CC00,$8C00
<  6>          .WORD   $0600,$0600,$0300,$0300
```

--- End of Chapter ---