```
SOURCE   FILE #01 =>L.MAIN
 INCLUDE FILE #02 =>ROM.SEQ.INFO
 INCLUDE FILE #03 =>MACROS
 INCLUDE FILE #04 =>BANKFF.EQUATES
 INCLUDE FILE #05 =>BANKFF.EQUATES2
 INCLUDE FILE #06 =>ROM.SEQ.EQU
 INCLUDE FILE #07 =>ROM.SEQ
 INCLUDE FILE #08 =>COUT.PORT1
 INCLUDE FILE #09 =>MOVIRAM
 INCLUDE FILE #10 =>SER.TST
 INCLUDE FILE #11 =>SOFT.SW
 INCLUDE FILE #12 =>FPI.SPEED
 INCLUDE FILE #13 =>CUSTOM.IRQ
 INCLUDE FILE #14 =>ROM.CHECKSUM
 INCLUDE FILE #15 =>CLOCK
 INCLUDE FILE #16 =>SHADOW
 INCLUDE FILE #17 =>BAT.RAM
 INCLUDE FILE #18 =>FDB.EQU
 INCLUDE FILE #19 =>FDB
 INCLUDE FILE #20 =>RAM.ADDR
0000:        0000    1          X65816
0000:                2 *LST OFF,NOA,NOV,NOEXP,SIXUP

0000:                3          LST   On,A,V,EXP,SIXUP
0000:                4          INCLUDE ROM.SEQ.INFO
```

```
0000:              3 ****************************************************************
0000:              4 *
0000:              5 * This is the ROM Test Sequencer for Burn-IN & Self-Test
0000:              6 *
0000:              7 * File Name: ROM.SEQ.              By Karl Grabe.  Cork.
0000:              8 *
0000:              9 *
0000:             10 * Documentation: TSKA 0047
0000:             11 * Rev: Beta 2.0 14 Apr 86
0000:             12 *
0000:             13 * Modification History:
0000:             14 *
0000:             15 *       20-Jan-86: Created, KG
0000:             16 *       17-Feb-86: Alpha 1.0, KG
0000:             17 *       10-Mar-86: Alpha 2.0, KG.  Add Clock, Interrupt,Speed.
0000:             18 *       24-Mar-86: Beta 1.0, KG. Add Interrupt, Staggered error status
0000:             19 *       23-Apr-86: Beta 2.0, KG. Add NVR, Shadow, Ram Address
0000:             20 *                                  Front Desk Bus
0000:             21 *                                  Port 1 status output
0000:             22 *
0000:             23 * Tests Available for Beta 2.0 and later revs
0000:             24 *
0000:             25 *       Error No          Test
0000:             26 *       ========          ================
0000:             27 *       01                ROM CHECKSUM
0000:             28 *       02                MOVING INVERSION RAM
0000:             29 *       03                SOFTSWITCH
0000:             30 *       04                RAM ADDRESS
0000:             31 *       05                FPI SPEED
0000:             32 *       06                SERIAL PORTS
0000:             33 *       07                REAL TIME CLOCK
0000:             34 *       08                BATTERY RAM
0000:             35 *       09                FRONT DESK BUS
0000:             36 *       0A                SHADOW
0000:             37 *       0B                INTERRUPTS
0000:             38 *
0000:             39 *            Copyright Apple Computer Inc. 1986
0000:             40 *                  All Rights Reserved
0000:             41 ****************************************************************
0000:             42 ;
0000:             43 ;
0000:             44 ****************************************************************
0000:             45 *
0000:             46 * The Cortland Test Diagnostics operate in Burn-In and
0000:             47 * Self Test modes:
0000:             48 *
0000:             49 * IN BURN-IN:
0000:             50 * The sequencer looks to Non Volatile Ram for a certain
0000:             51 * sequence, put there by pre burn-in functional tester.
0000:             52 * If this sequence is not found then the Pre-BI tester failed to set
0000:             53 * up NVR or the UUT/NVR is bad so the test halts setting the FAIL.LED(D).
0000:             54 * IF this sequence is correct then a tests mask (= what tests to run) in NVR
0000:             55 * is used instead of the default mask. This allows the Pre-BI tester
0000:             56 * to determine what tests are to be run in Burn-In.
0000:             57 *
0000:             58 * The entire test sequence is repeated for a number of cycles and this number
0000:             59 * is also programmable by the Pre-BI tester. When the cycle completes
0000:             60 * the power off annunciator is activated to tell the BI controller that
```

```
0000:                61 * the UUT is inactive and ready to power down. Test fail status ( if any )
0000:                62 * is stored in NVR to be read by the Post-BI tester. The tester can thus
0000:                63 * determine if a board failed in Burn-IN and if so what test failed,
0000:                64 * it's error code, and during what cycle the last fail occurred.
0000:                65 *
0000:                66 *.....................................................................
0000:                67 *
0000:                68 * For Self Test (User):
0000:                69 * The test mask (ie what tests to run) for self test
0000:                70 * is a subest of the Burn-IN test mask.
0000:                71 *
0000:                72 * After all tests have been sucessfully completed a
0000:                73 * continous 1 KHz frequence beep is emmitted.
0000:                74 * A 'System Good' message is displayed on the screen.
0000:                75 *
0000:                76 * If any tests fail then a 6 KHz beep is emmitted.
0000:                77 * A 'System Bad: AABBCCDD' message is displayed.
0000:                78 * where AA is the test number that failed and BB-DD is
0000:                79 * the fail code. The fail code is output to port 1
0000:                80 * at 1200 baud preceded by a '*'.
0000:                81 *
0000:                82 * AN0/1 and disk port phases 1 & 2 are used to tell
0000:                83 * the outside world that the test sequence is ongoing/complete and
0000:                84 * and that the UUT has passed/failed the self test.
0000:                85 *
0000:                86 * NVR is never read or written to by the sequencer.
0000:                87 * Some of the tests may use NVR but if so they will restore
0000:                88 * its contents after the test is complete.
0000:                89 *
0000:                90 *.....................................................................
0000:                91 *
0000:                92 * External test software can use its own mask ( store in
0000:                93 * C register ) by calling EXT.SEQ. The EXT.SEQ address is
0000:                94 * stored just after the last test pointer in the Test
0000:                95 * pointer table.
0000:                96 * All 11 ( at present ) tests are represented by an on/off
0000:                97 : bit in the C register when using the EXT.SEQ call.
0000:                98 * The C LSB is the first test in the test table etc.
0000:                99 *
0000:               100 ***********************************************************************
```

```
0000:  ;.....................................................................
0000:  ;
0000:  ; Soft Switches for Burn In:
0000:  ;
0000:  ; Note: a) 'wr NVR' = write test STATUS (6) registers
0000:  ;          to Non Volatile Ram.
0000:  ;
0000:  ;       b) DIAGTYPE (MSB $C046) is active lo eg MSB=0 is a hi
0000:  ;          on the Burn In connector
0000:  ;
0000:  ; KG 04 Feb 86
0000:  ;_____________________________________________________________________
0000:  ;   |DIAGTYPE*|   SW0   |   SW1   |          MODE                  |
0000:  ;=====================================================================
0000:  ;   |    0    |    0    |    0    | Normal Power UP                |
0000:  ;---------------------------------------------------------------------
0000:  ;   |    0    |    0    |    1    | Control Panel                  |
0000:  ;---------------------------------------------------------------------
0000:  ;   |    0    |    1    |    0    | Cold Restart                   |
0000:  ;---------------------------------------------------------------------
0000:  ;   |    0    |    1    |    1    | Self Diagnostic, no wr NVR     |
0000:  ;---------------------------------------------------------------------
0000:  ;   |    1    |    0    |    X    | BI Diagnostic 1, wr NVR        |
0000:  ;---------------------------------------------------------------------
0000:  ;   |    1    |    1    |    X    | Self Diagnostic, no wr NVR     |
0000:  ;---------------------------------------------------------------------
0000:  ;   | BI.FLAG | BI.ALT  |  SELF   | <-- Test s/w Equate Names      |
0000:  ;---------------------------------------------------------------------
0000:  ;   |  9/10   |   5/6   |  ----   | <-- BI Connector pins          |
0000:  ;---------------------------------------------------------------------
0000:  ;
0000:  ; Burn In Annunciator/Disk Port Truth Table:
0000:  ;
0000:  ; Note: The PASS/FAIL.LED (AN0 & Phase 1) signal also operates
0000:  ;       in self test. Use an LED connected via a buffer to game
0000:  ;       I/O connector/disk port to verify UUT without VDU.
0000:  ;
0000:  ;_____________________________________________________________________
0000:  ;   |   AN0   |   AN1   |          MODE                              |
0000:  ;=====================================================================
0000:  ;   |    1    |    0    | Passed last cycle, continue                |
0000:  ;---------------------------------------------------------------------
0000:  ;   |    0    |    0    | Failed last cycle, continue                |
0000:  ;---------------------------------------------------------------------
0000:  ;   |    1    |    1    | Pass, ready to power down                  |
0000:  ;---------------------------------------------------------------------
0000:  ;   |    0    |    1    | Fail, ready to power down                  |
0000:  ;---------------------------------------------------------------------
0000:  ;   |PASS.LED |POWER.ON | <-- Equate Names to turn on  (1)           |
0000:  ;   |FAIL.LED |POWER.OFF| <-- Equate Names to turn off (0)           |
0000:  ;---------------------------------------------------------------------
0000:  ;   | Phase 1 | Phase 2 | <-- Final System Test with cover on       |
0000:  ;---------------------------------------------------------------------
0000:  ;   | 13/14   |  17/18  | <-- BI Connector Pins                      |
0000:  ;---------------------------------------------------------------------
0000:  ;
0000:  ; Note: Cork uses disk port phase lines P1 & P2 to match AN0 & AN1
0000:  ;
```

```
0000:  ;.....................................................................
0000:  ;
0000:  ; Tests available:
0000:  ;
0000:  ; Note: Each TEST NO corresponds to a bit in the 16 bit
0000:  ;       C register on calling EXT.SEQ. If the bit is true
0000:  ;       then the test is run, else it is skipped.
0000:  ;       The test subroutine must clear the carry if the
0000:  ;       test passed, else set the carry. The sequencer
0000:  ;       automatically  passes a skipped test. Two different
0000:  ;       groups of tests can be run as defined in the equates file
0000:  ;       ( a third set can be defined in Battery backed up ram):
0000:  ;
0000:  ;_____________________________________________________________
0000:  ;   | TST NO  |  Test(s) performed                              |
0000:  ;=================================================================
0000:  ;   | LSB: 0  |  Bank $FE/$FF Roms Checksum                     |
0000:  ;-----------------------------------------------------------------
0000:  ;   |      1  |  Softswitch/STATEREG test                       |
0000:  ;-----------------------------------------------------------------
0000:  ;   |      2  |  Ram:  Moving Inversions                        |
0000:  ;-----------------------------------------------------------------
0000:  ;   |      3  |  Ram: Addressing                                |
0000:  ;-----------------------------------------------------------------|
0000:  ;   |      4  |  FPI/Video Counters Speed verification          |
0000:  ;-----------------------------------------------------------------
0000:  ;   |      5  |  Serial: Internal Loopback                      |
0000:  ;-----------------------------------------------------------------
0000:  ;   |      6  |  Real Time Clock                                |
0000:  ;-----------------------------------------------------------------
0000:  ;   |      7  |  Battery Ram                                    |
0000:  ;-----------------------------------------------------------------
0000:  ;   |      8  |  Front Desk Bus Processor & Rom                 |
0000:  ;-----------------------------------------------------------------
0000:  ;   |     09  |  Shadow register                                |
0000:  ;-----------------------------------------------------------------
0000:  ;   |     10  |  Interrupts                                     |
0000:  ;-----------------------------------------------------------------
0000:  ;   |     11  |  -                                              |
0000:  ;-----------------------------------------------------------------
0000:  ;   |     12  |  -                                              |
0000:  ;-----------------------------------------------------------------
0000:  ;   |     13  |  -                                              |
0000:  ;-----------------------------------------------------------------
0000:  ;   |     14  |  -                                              |
0000:  ;-----------------------------------------------------------------
0000:  ;   | MSB:15  |  -                                              |
0000:  ;-----------------------------------------------------------------
```

```
0000:   ****************************************************************
0000:   : Error Message: "System Bad: AABBCCDD
0000:   ; Where AA is the test number that failed.
0000:   :
0000:   ;_____________________________________________________________
0000:   :   | Test     | Fail Codes                                    |
0000:   ;=================================================================
0000:   :   | ROM      |  BBCC= Failed Checksum, DD=01->Ram err        |
0000:   ;-----------------------------------------------------------------
0000:   :   | Ram 1    |  BB=BANK, CC=ADDRESS                          |
0000:   ;-----------------------------------------------------------------
0000:   :   | Speed    |  BB=1:Stuck slow, BB=2: Stuck fast. CC=Bad Value|
0000:   ;-----------------------------------------------------------------
0000:   :   | Soft.sw  |  BB=STATEREG BIT, CC=READ SOFTSWITCH ADDRESS  |
0000:   ;-----------------------------------------------------------------
0000:   :   | Ram MOVI|  BB=BANK, CC=BIT                               |
0000:   ;-----------------------------------------------------------------
0000:   :   | Ser.tst  |  See Test code                                |
0000:   ;-----------------------------------------------------------------
0000:   :   | IRQ      |  See Test code                                |
0000:   ;-----------------------------------------------------------------
0000:   :   | CLOCK    | If DD=01-> Tool Error encountered, BB= error #  |
0000:   ;-----------------------------------------------------------------
0000:   :   | FDB      | BBCC= Failed Checksum, DD=01-> Tool error     |
0000:   ;-----------------------------------------------------------------
0000:   :   | RAM addr| As Moviram                                     |
0000:   ;-----------------------------------------------------------------
0000:   :   | BAT.RAM | BB=1-> Address test, CC=Failed Address         |
0000:   ;   |          | BB=2-> Pattern, CC=Pattern, DD= Address        |
0000:   ;-----------------------------------------------------------------
0000:   ;   | SHADOW   | None                                          |
0000:   ;-----------------------------------------------------------------
0000:   ;   |          |                                               |
0000:   ;-----------------------------------------------------------------
0000:   ;   |          |                                               |
0000:   ;-----------------------------------------------------------------
0000:   ;   |          |                                               |
0000:   ;-----------------------------------------------------------------
0000:   ;   |          |                                               |
0000:   ;-----------------------------------------------------------------
0000:   ;
0000:   ****************************************************************
0000:   *
0000:   * Note:  For debugging the sequencer code Use SEQ.DEBUG =1
0000:   *        This swaps DIAGTYPE for SW2. Note that SW2 for
0000:   *        debugging is inverted to DIAGTYPE ie a 0 is
0000:   *        required to be in Burn In. Also do a JMP $FF7400
0000:   *        to get into the diagnostics code. SEQ.DEBUG is a
0000:   *        variable at the beginning of file ROM.SEQ and is used
0000:   *        for conditional assembly.
0000:   *
0000:   ****************************************************************
0000:          INCLUDE MACROS
```

```
0000:          2 ********************************************************
0000:          3 *
0000:          4 * 65816 Macros for word size switching
0000:          5 *
0000:          6 * kg Sep 85
0000:          7 *
0000:          8 ********************************************************
0000:          9          MACLIB MACRO.TEMP
0000:         10 ****************************************
0000:         11 NTV      MACRO              ;Native mode ( E=0 )
 M            12          CLC
 M            13          XCE                ;Get into native mode
0000:         14          mend
0000:         15 *
0000:         16 *
0000:         17 NTV16    MACRO              GET INT NATIVE MODE
 M            18          CLC
 M            19          XCE                ;SET E = 0 ( Native mode )
 M            20          REP   #%00110000
 M            21          MEMORY16
 M            22          INDEX16
0000:         23          MEND
0000:         24 *
0000:         25 *
0000:         26 EML      MACRO              SET TO EMULATION MODE
 M            27          SEC
 M            28          XCE                ;SET E=1 ( Emulation mode )
 M            29          MEMORY8
 M            30          INDEX8
0000:         31          MEND
0000:         32 ****************************************
0000:         33 *
0000:         34 *
0000:         35 DATA16   MACRO              SET 16 BIT DATA WORDS, MEMORY & X
 M            36          REP   #%00110000   ;SET M=0, X=0 (16 BIT)
 M            37          MEMORY16
 M            38          INDEX16
0000:         39          MEND
0000:         40 *
0000:         41 *
0000:         42 DATA8    MACRO              ;SET 8 BIT MEMORY,INDEX
 M            43          SEP   #%00110000   ;SET M=1, X=1 (8 BIT )
 M            44          MEMORY8
 M            45          INDEX8
0000:         46          MEND
0000:         47 ****************************************
0000:         48 *
0000:         49 *
0000:         50 IND16    MACRO              ;Set X=0 ( 16 bit index )
 M            51          REP   #%00010000   ;X=0 ( 16 bit index )
 M            52          INDEX16
0000:         53          MEND
0000:         54 *
0000:         55 *
0000:         56 MEM16    MACRO              ;Set M=0 ( 16 bit memory )
 M            57          REP   #%00100000   ;M=0 (16 bit memory)
 M            58          MEMORY16
0000:         59          MEND
```

```
0000:                60 *
0000:                61 *
0000:                62 ****************************************
0000:                63 IND8      MACRO             ;Set X=1 ( 8 bit index )
 M                   64           SEP   #%00010000  ;Set X=1 ( 8bit index )
 M                   65           INDEX8
0000:                66           MEND
0000:                67 *
0000:                68 *
0000:                69 MEM8      MACRO             ;Set M=1 ( 8 bit maemory )
 M                   70           SEP   #%00100000  ;Set M=1 ( 8 bit memory )
 M                   71           MEMORY8
0000:                72           MEND
0000:                73 ****************************************
0000:                74 *
0000:                75 * FPI FAST/SLOW macros.
0000:                76 *
0000:                77 SLOW      MACRO             ;Set fpi slow clock
 M                   78           STZ   CYAREG      ;Go into slow mode
0000:                79           MEND
0000:                80 FAST      MACRO
 M                   81           LDA   #%10000000
 M                   82           STA   CYAREG      ;Set fpi fast
0000:                83           MEND
0000:                84 ****************************************
0000:                85 *
0000:                86 * Message length macro
0000:                87 * Uses same format as ProDOS pathname length
0000:                88 *
0000:                89 * Use:
0000:                90 *      MSG.0 MSG.LEN
0000:                91 *           ASC "<Fist message 0>"
0000:                92 *      MSG.1 ASC "The 2nd message"
0000:                93 *      .
0000:                94 *      .
0000:                95 *      MSG.n DFB $00  ;Last message
0000:                96 *
0000:                97 ****************************************
0000:                98 MSG.LEN   MACRO
 M                   99           DFB   MSG.&X-*-1
0000:               100           MEND
0000:                 6           INCLUDE BANKFF.EQUATES
```

```
0000:                3 ********************************************************
0000:                4 *                                                      *
0000:                5 *                  Bank $FF Variables                  *
0000:                6 *                                                      *
0000:                7 *                       by                             *
0000:                8 *                    Fern Bachman                      *
0000:                9 *                                                      *
0000:               10 *         Copyright Apple Computer, Inc. 1985         *
0000:               11 *               All Rights Reserved.                   *
0000:               12 *                                                      *
0000:               13 ********************************************************

0000:        0006   15 F8VERSION EQU   $06         ;F8 ROM //e ID byte

0000:               17 * End of variables used by miniassembler

0000:               19 *
0000:               20 * ZERO PAGE EQUATES
0000:               21 *
0000:        0000   22 LOC0      EQU   $00         ;vector for autostart from disk
0000:        0001   23 LOC1      EQU   $01
0000:        0006   24 PRINTTMP  EQU   $06         ;$06-$08 saved and restore by print rtne
0000:        0020   25 WNDLFT    EQU   $20         ;left edge of text window
0000:        0021   26 WNDWDTH   EQU   $21         ;width of text window
0000:        0022   27 WNDTOP    EQU   $22         ;top of text window
0000:        0023   28 WNDBTM    EQU   $23         ;bottom+1 of text window
0000:        0024   29 CH        EQU   $24         ;cursor horizontal position
0000:        0025   30 CV        EQU   $25         ;cursor vertical position
0000:        0026   31 GBASL     EQU   $26         ;lo-res graphics base addr.
0000:        0027   32 GBASH     EQU   $27
0000:        0028   33 BASL      EQU   $28         ;text base address
0000:        0029   34 BASH      EQU   $29
0000:        002A   35 BAS2L     EQU   $2A         ;temp base for scrolling
0000:        002B   36 BAS2H     EQU   $2B
0000:        002C   37 H2        EQU   $2C         ;temp for lo-res graphics
0000:        002C   38 LMNEM     EQU   $2C         ;temp for mnemonic decoding
0000:        002D   39 V2        EQU   $2D         ;temp for lo-res graphics
0000:        002D   40 RMNEM     EQU   $2D         ;temp for mnemonic decoding
0000:        002E   41 MASK      EQU   $2E         ;color mask for lo-res gr.
0000:        002E   42 FORMAT    EQU   $2E         ;temp for opcode decode
0000:        002F   43 LENGTH    EQU   $2F         ;temp for opcode decode
0000:        0030   44 COLOR     EQU   $30         ;color for lo-res graphics
0000:        0031   45 MONMODE   EQU   $31         ;Monitor mode
0000:        0032   46 INVFLG    EQU   $32         ;normal/inverse(/flash)
0000:        0033   47 PROMPT    EQU   $33         ;prompt character
0000:        0034   48 YSAV      EQU   $34         ;position in Monitor command
0000:        0035   49 YSAV1     EQU   $35         ;temp for Y register
0000:        0036   50 CSWL      EQU   $36         ;character output hook
0000:        0037   51 CSWH      EQU   $37
0000:        0038   52 KSWL      EQU   $38         ;character input hook
0000:        0039   53 KSWH      EQU   $39
0000:        003A   54 PCL       EQU   $3A         ;temp for program counter
0000:        003B   55 PCH       EQU   $3B
0000:        003C   56 A1L       EQU   $3C         ;Monitor temp
0000:        003D   57 A1H       EQU   $3D         ;Monitor temp
```

```
0000:        003E   58 A2L       EQU   $3E         ;Monitor temp
0000:        003F   59 A2H       EQU   $3F         ;Monitor temp
0000:        0040   60 A3L       EQU   $40         ;Monitor temp
0000:        0041   61 A3H       EQU   $41         ;Monitor temp
0000:        0042   62 A4L       EQU   $42         ;Monitor temp
0000:        0043   63 A4H       EQU   $43         ;Monitor temp
0000:               64 *A5L EQU $44 ;Monitor temp
0000:        0045   65 A5H       EQU   $45         ;Monitor temp
0000:               66 *ACC EQU $45 ;Acc after break (destroys A5H)
0000:        0046   67 XREG      EQU   $46         ;X reg after break
0000:        0047   68 YREG      EQU   $47         ;Y reg after break
0000:        0048   69 STATUS    EQU   $48         ;P reg after break
0000:        0049   70 SPNT      EQU   $49         ;SP after break
0000:        004E   71 RNDL      EQU   $4E         ;random counter low
0000:        004F   72 RNDH      EQU   $4F         ;random counter high

0000:               74 *
0000:               75 * Value equates
0000:               76 *

0000:        0006   78 GOODFB    EQU   $06         ;//e ID byte
0000:        0010   79 X8        EQU   $10         ;Mask for interrupt code
0000:        0020   80 M8        EQU   $20         ;Mask for interrupt code
0000:        0038   81 EXMEMCMD  EQU   $38         ;Mask for extnd mem designator/sctr # cmd
0000:        AAAA   82 CHKSMCNST EQU   $AAAA       ;BATTERYRAM checksum EOR constant

0000:               84 *
0000:               85 * Bank 1 / Page 1 permanent storage
0000:               86 *

0000:        0100   88 MNEMSTKPTR EQU  $010100     ;Main stack pointer
0000:        0101   89 ALEMSTKPTR EQU  $010101     ;Alternate stack pointer

0000:               91 *
0000:               92 * Characters read by GETLN are placed in
0000:               93 * IN, terminated by a carriage return.
0000:               94 *

0000:        0200   96 IN        EQU   $0200       ;input buffer for GETLN

0000:               98 *
0000:               99 * Page 3 vectors
0000:              100 *

0000:        03D0  102 OSWMSTRT  EQU   $3D0        ;QUITMON jumps here
0000:        03F0  103 BRKV      EQU   $03F0       ;vectors here after break
0000:        03F2  104 SOFTEV    EQU   $03F2       ;vector for warm start
0000:        03F4  105 PWREDUP   EQU   $03F4       ;THIS MUST = EOR #$A5 OF SOFTEV+1
0000:              106 *AMPERV EQU $03F5 ;APPLESOFT & EXIT VECTOR
0000:        03F8  107 USRADR    EQU   $03F8       ;APPLESOFT USR function vector
0000:        03FB  108 NMI       EQU   $03FB       ;NMI vector
0000:        03FE  109 IRQLOC    EQU   $03FE       ;Maskable interrupt vector
0000:        0400  110 LINE1     EQU   $0400       ;first line of text screen
```

```
0000:                 112 * HARDWARE EQUATES

0000:                 114          DSECT
C000:          C000   115          ORG   $C000
C000:          C000   116 IOADR     EQU   *          :All I/O is at $Cxxx
C000:          C000   117 KBD       EQU   *          ;Bit 7=1 if keystroke
C000:00               118 CLR80COL  DFB   0          :disable 80 column store
C001:00               119 SET80COL  DFB   0          :enable 80 column store
C002:00               120 RDMAINRAM DFB   0          :read from main 48K RAM
C003:00               121 RDCARDRAM DFB   0          ;read from alt. 48K RAM
C004:00               122 WRMAINRAM DFB   0          :write to main 48K RAM
C005:00               123 WRCARDRAM DFB   0          ;write to alt. 48K RAM
C006:00               124 SETSLOTCXROM DFB 0         :use ROMS on cards
C007:00               125 SETINTCXROM DFB 0          ;use internal ROM
C008:00               126 SETSTDZP  DFB   0          :use main zero page/stack
C009:00               127 SETALTZP  DFB   0          ;use alt. zero page/stack
C00A:00               128 SETINTC3ROM DFB 0          ;Enable internal slot 3 ROM
C00B:00               129 SETSLOTC3ROM DFB 0         ;Enable external slot 3 ROM
C00C:00               130 CLR80VID  DFB   0          :disable 80 column hardware
C00D:00               131 SET80VID  DFB   0          ;enable 80 column hardware
C00E:00               132 CLRALTCHAR DFB  0          ;normal LC, flashing UC
C00F:00               133 SETALTCHAR DFB  0          ;normal inverse, LC; no flash
C010:00               134 KBDSTRB   DFB   0          :turn off key pressed flag
C011:00               135 RDLCBNK2  DFB   0          ;Bit 7=1 if LC bank 2 is in
C012:00               136 RDLCRAM   DFB   0          :Bit 7=1 if LC RAM read enabled
C013:00               137 RDRAMRD   DFB   0          ;Bit 7=1 if reading alt 48K
C014:00               138 RDRAMWRT  DFB   0          ;Bit 7=1 if writing alt 48K
C015:00               139 RDCXROM   DFB   0          ;Bit 7=1 if using int rom
C016:00               140 RDALTZP   DFB   0          ;Bit 7=1 if slot zp enabled
C017:00               141 RDC3ROM   DFB   0          ;Bit 7=1 if slot c3 space enabled
C018:00               142 RD80COL   DFB   0          ;Bit 7=1 if 80 column store
C019:00               143 RDVBLBAR  DFB   0          ;Bit 7=1 if not VBL
C01A:00               144 RDTEXT    DFB   0          :Bit 7=1 if text (not graphics)
C01B:00               145 RDMIX     DFB   0          ;Bit 7=1 if mixed mode on
C01C:00               146 RDPAGE2   DFB   0          :Bit 7=1 if TXTPAGE2 switched in
C01D:00               147 RDHIRES   DFB   0          ;Bit 7=1 if HIRES is on
C01E:00               148 ALTCHARSET DFB  0          :Bit 7=1 if alternate char set in use
C01F:00               149 RD80VID   DFB   0          ;Bit 7=1 if 80 column hardware in
C020:00               150           DFB   0          ;Reserved for future system expansion

C021:                 152 * _7_____6_____5_____4_____3_____2_____1_____0__
C021:                 153 *|      |      |      |      |      |      |      |      |
C021:                 154 *|Enable|      |      |      |      |      |      |      |
C021:                 155 *|color/|   0  |   0  |   0  |   0  |   0  |   0  |   0  |
C021:                 156 *| mono |      |      |      |      |      |      |      |
C021:                 157 *|______|______|______|______|______|______|______|______|
C021:                 158 *           ^^^^^  MONOCOLOR Status Byte ^^^^^

C021:                 160 * MONOCOLOR bits defined as follows
C021:                 161 *    bit 7= 0 enables color -- 1 disables color
C021:                 162 *    bit 6,5,4,3,2,1,0  Must be 0

C021:00               164 MONOCOLOR DFB   0          ;Monochrome/color select register

C022:                 166 * _7_____6_____5_____4_____3_____2_____1_____0__
C022:                 167 *|      |      |      |      |      |      |      |      |
C022:                 168 *|      |      |      |      |      |      |      |      |
C022:                 169 *|      Text Color Bits     |   Background Color Bits   |
```

```
C022:                170 *|      |      |      |      |      |      |      |      |
C022:                171 *|______|______|______|______|______|______|______|______|
C022:                172 *              ^^^^^  TBCOLOR Byte ^^^^^

C022:                174 * TBCOLOR bits defined as follows
C022:                175 *    bit 7,6,5,4 = Text color bits
C022:                176 *    bits 3,2,1,0 = Background color bits
C022:                177 *
C022:                178 *   Color bits =
C022:                179 *                $0 = Black
C022:                180 *                $1 = Magenta
C022:                181 *                $2 = Dark Blue
C022:                182 *                $3 = Purple
C022:                183 *                $4 = Dark green
C022:                184 *                $5 = Dark Gray
C022:                185 *                $6 = Medium Blue
C022:                186 *                $7 = Light Blue
C022:                187 *                $8 = Brown
C022:                188 *                $9 = Orange
C022:                189 *                $A = Light Gray
C022:                190 *                $B = Pink
C022:                191 *                $C = Green
C022:                192 *                $D = Yellow
C022:                193 *                $E = Aquamarine
C022:                194 *                $F = White

C022:00              196 TBCOLOR   DFB   0          ;Text/background color select register

C023:                198 * _7_____6_____5_____4_____3_____2_____1_____0__
C023:                199 *|      |      |      |      |      |      |      |      |
C023:                200 *| VGC  | 1sec | Scan | Ext  |      | 1sec | Scan | Ext  |
C023:                201 *|  int |  int |  int | int  |Unused|  int |  int | int  |
C023:                202 *|active|active|active|active|      |enable|enable|enable|
C023:                203 *|______|______|______|______|______|______|______|______|
C023:                204 *           ^^^^^  VGCINT Status Byte ^^^^^

C023:                206 * VGCINT bits defined as follows
C023:                207 *    bit 7= 1 if interrupt generated by VGC
C023:                208 *    bit 6= 1 if 1 second timer interrupt
C023:                209 *    bit 5= 1 if scan line interrupt
C023:                210 *    bit 4= 1 if external interrupt
C023:                211 *    bit 3= unused
C023:                212 *    bit 2= 1 second timer interrupt enable
C023:                213 *    bit 1= scan line interrupt enable
C023:                214 *    bit 0= external interrupt enable

C023:00              216 VGCINT    DFB   0          ;VGC interrupt register

C024:                218 * _7_____6_____5_____4_____3_____2_____1_____0__
C024:                219 *|      |      |      |      |      |      |      |      |
C024:                220 *|Button|      |                                          |
C024:                221 *|status|Delta |            Delta movement                |
C024:                222 *| now  | sign |                                          |
C024:                223 *|______|______|______|______|______|______|______|______|
C024:                224 *           ^^^^^  MOUSEDATA Byte ^^^^^

C024:                226 * MOUSEDATA bits defined as follows
C024:                227 *    bit 7= if reading X data = button 1 status
```

```
C024:                228 *           if reading Y data = button 0 status
C024:                229 *    bit 6= sign of delta 0='+' -- 1='-'
C024:                230 *    bit 5,4,3,2,1,0 = delta movement

C024:00              232 MOUSEDATA DFB   0           ;X or Y mouse data register

C025:                234 * _7_____6_____5_____4_____3_____2_____1_____0__
C025:                235 *|      |      |Update|      |      |      |      |
C025:                236 *| Open |Closed| mod  |Keypad|Repeat| Caps | Ctrl |Shift |
C025:                237 *| Apple| Apple|no key|  key |active| lock |  key | key  |
C025:                238 *|  key |  key | press|active|      |active|active|active|
C025:                239 *|______|______|______|______|______|______|______|______|
C025:                240 *           ^^^^^  KEYMODREG Status Byte ^^^^^

C025:                242 * KEYMODREG bits defined as follows
C025:                243 *    bit 7= Open Apple key active
C025:                244 *    bit 6= Closed Apple key active
C025:                245 *    bit 5= Updated modifier latch without keypress
C025:                246 *    bit 4= Keypad key active
C025:                247 *    bit 3= Repeat active
C025:                248 *    bit 2= Caps lock active
C025:                249 *    bit 1= Control key active
C025:                250 *    bit 0= Shift key active

C025:00              252 KEYMODREG DFB   0           ;Key modifier register

C026:                254 * _7_____6_____5_____4_____3_____2_____1_____0__
C026:                255 *|      |      |      |      |      |      |      |      |
C026:                256 *|                                                       |
C026:                257 *|              Data to/from keyboard micro              |
C026:                258 *|                                                       |
C026:                259 *|______|______|______|______|______|______|______|______|
C026:                260 *              ^^^^^  DATAREG Byte ^^^^^

C026:                262 * DATAREG bits defined as follows
C026:                263 *    bit 7,6,5,4,3,2,1,0 = data to/from keyboard micro
C026:                264 *
C026:                265 * Data at interrupt time in this register is defined as
C026:                266 *    bit 7= Response byte if set, otherwise status byte
C026:                267 *    bit 6= ABORT valid if set and all other bits reset
C026:                268 *    bit 5= Desktop manager key sequence pressed
C026:                269 *    bit 4= Flush buffer key sequence pressed
C026:                270 *    bit 3= SRQ valid if set
C026:                271 *    bit 2,1,0 If all bits clear then no FDB data valid,
C026:                272 *              else the bits indicate the number of valid
C026:                273 *              bytes received minus 1. (2-8 bytes total)

C026:00              275 DATAREG   DFB   0           ;Data register in KeyGlu chip

C027:                277 * _7_____6_____5_____4_____3_____2_____1_____0__
C027:                278 *|      |      |      |      |      |      |      |      |
C027:                279 *| Mouse| Mouse| Data | Data | Key  | Key  | Mouse| Cmd  |
C027:                280 *|  reg |  int |  reg |  int | data | int  |X/Yreg| reg  |
C027:                281 *| full |enable| full |enable| full |enable| data | full |
C027:                282 *|______|______|______|______|______|______|______|______|
C027:                283 *           ^^^^^  KMSTATUS Status Byte ^^^^^

C027:                285 * KMSTATUS bits defined as follows
```

```
C027:            286 *    bit 7= 1 if mouse register full
C027:            287 *    bit 6= mouse interrupt disable/enable
C027:            288 *    bit 5= 1 if data register full
C027:            289 *    bit 4= data interrupt enable
C027:            290 *    bit 3= 1 if key data full
C027:            291 *    bit 2= key data interrupt enable
C027:            292 *    bit 1= 0 = mouse 'X' register data available
C027:            293 *           1 = mouse 'Y' register data available
C027:            294 *    bit 0= Command register full

C027:00          296 KMSTATUS  DFB   0          ;Keyboard/mouse status register
C028:00          297 ROMBANK   DFB   0          ;ROM bank select toggle

C029:            299 * _7_____6_____5_____4_____3_____2_____1_____0__
C029:            300 *|      |      |      |      |      |      |      |
C029:            301 *|Enable|Linear| B/W  |      |      |      |      |
C029:            302 *| super| video|Color |  0   |  0   |  0   |  0   |
C029:            303 *|hi-res|      |DHires|      |      |      |      |
C029:            304 *|______|______|______|______|______|______|______|
C029:            305 *                ^^^^^  NEWVIDEO Byte ^^^^^

C029:            307 * NEWVIDEO bits defined as follows
C029:            308 *    bit 7= 1 to disable Apple //e video (enables super hi-res)
C029:            309 *    bit 6= 1 to linearize for super hi-res
C029:            310 *    bit 5= 0 for color double hi-res -- 1 for B/W hi-res
C029:            311 *    bit 4,3,2,1= MUST be pregrammed as 0
C029:            312 *    bit 0= Enable bank 1 latch to allow long instructions
C029:            313 *           to access bank 1 directly.

C029:00          315 NEWVIDEO  DFB   0          ;Video/enable read alt mem with long instr
C02A:00          316           DFB   0          ;Reserved for future system expansion

C02B:            318 * _7_____6_____5_____4_____3_____2_____1_____0__
C02B:            319 *|      |      |      |      |      |      |      |
C02B:            320 *| Character Generator| NTSC/| Lang |      |      |
C02B:            321 *|   language select  |  PAL |select|   0  |   0  |   0  |
C02B:            322 *|      |      |      |      | bit  |      |      |
C02B:            323 *|______|______|______|______|______|______|______|
C02B:            324 *                ^^^^^  LANGSEL Byte ^^^^^

C02B:            326 * LANGSEL bits defined as follows
C02B:            327 *    bit 7,6,5= Character generator language select
C02B:            328 *                    Primary language    Secondary language
C02B:            329 *               $0 =       USA                 Dvorak
C02B:            330 *               $1 =       UK                  USA
C02B:            331 *               $2 =     French                USA
C02B:            332 *               $3 =     Danish                USA
C02B:            333 *               $4 =     Spanish               USA
C02B:            334 *               $5 =     Italian               USA
C02B:            335 *               $6 =     German                USA
C02B:            336 *               $7 =     Swedish               USA
C02B:            337 *    bit 4= 0 if NTSC video mode -- 1 if PAL video mode
C02B:            338 *    bit 3= LANGUAGE switch bit 1 if primary lang set selected
C02B:            339 *    bit 2,1,0 ;MUST be programmed as 0's

C02B:00          341 LANGSEL   DFB   0          ;Language/PAL/NTSC select register
C02C:00          342 CHARROM   DFB   0          ;Addr for tst mode read of character ROM
```

```
C02D:            344 * _7_____6_____5_____4_____3_____2_____1_____0__
C02D:            345 *|      |      |      |      |      |      |      |
C02D:            346 *|Slot7 |Slot6 |Slot5 |Slot4 |      |Slot2 |Slot1 |      |
C02D:            347 *|intext|intext|intext|intext|   0  |intext|intext|   0  |
C02D:            348 *|enable|enable|enable|enable|      |enable|enable|      |
C02D:            349 *|______|______|______|______|______|______|______|______|
C02D:            350 *               ^^^^^  SLTROMSEL Byte ^^^^^

C02D:            352 * SLTROMSEL bits defined as follows
C02D:            353 *    bit 7= 0 enables internal slot 7 -- 1 enables slot ROM
C02D:            354 *    bit 6= 0 enables internal slot 6 -- 1 enables slot ROM
C02D:            355 *    bit 5= 0 enables internal slot 5 -- 1 enables slot ROM
C02D:            356 *    bit 4= 0 enables internal slot 4 -- 1 enables slot ROM
C02D:            357 *    bit 3= MUST be 0
C02D:            358 *    bit 2= 0 enables internal slot 2 -- 1 enables slot ROM
C02D:            359 *    bit 1= 0 enables internal slot 1 -- 1 enables slot ROM
C02D:            360 *    bit 0= Must be 0

C02D:00          362 SLTROMSEL DFB   0          ;Slot ROM select
C02E:00          363 VERTCNT   DFB   0          ;Addr for read of video cntr bits V5-V8
C02F:00          364 HORIZCNT  DFB   0          ;Addr for read of video cntr bits VA-H0
C030:00          365 SPKR      DFB   0          ;clicks the speaker

C031:            367 * _7_____6_____5_____4_____3_____2_____1_____0__
C031:            368 *|      |      |      |      |      |      |      |      |
C031:            369 *| 3.5" | 3.5" |      |      |      |      |      |      |
C031:            370 *| head | drive|  0   |  0   |  0   |  0   |  0   |  0   |
C031:            371 *|select|enable|      |      |      |      |      |      |
C031:            372 *|______|______|______|______|______|______|______|______|
C031:            373 *           ^^^^^  DISKREG Status Byte ^^^^^

C031:            375 * DISKREG bits defined as follows
C031:            376 *    bit 7= 1 to select head on 3.5" drive to use
C031:            377 *    bit 6= 1 to enable 3.5" drive
C031:            378 *    bit 5,4,3,2,1,0= unused

C031:00          380 DISKREG   DFB   0          ;Used for 3.5" disk drives

C032:            382 * _7_____6_____5_____4_____3_____2_____1_____0__
C032:            383 *|      |      |      |      |      |      |      |      |
C032:            384 *|      |Clear |Clear |      |      |      |      |      |
C032:            385 *|   0  |1 sec | scan |   0  |   0  |   0  |   0  |   0  |
C032:            386 *|      | int  |In int|      |      |      |      |      |
C032:            387 *|______|______|______|______|______|______|______|______|
C032:            388 *                ^^^^^  SCANINT Byte ^^^^^

C032:            390 * SCANINT bits defined as follows
C032:            391 *    bit 7= unused
C032:            392 *    bit 6= write a 0 here to reset 1 second interrupt
C032:            393 *    bit 5= write a 0 here to clear scan line interrupt
C032:            394 *    bit 4= unused
C032:            395 *    bit 3= unused
C032:            396 *    bit 2= unused
C032:            397 *    bit 1= unused
C032:            398 *    bit 0= unused

C032:00          400 SCANINT   DFB   0          ;Scan line interrupt register
```

```
C033:              * _7_____6_____5_____4_____3_____2_____1_____0__
C033:              *|      |      |      |      |      |      |      |      |
C033:              *|                                                        |
C033:              *|                  Clock data register                   |
C033:              *|                                                        |
C033:              *|_____|_____|_____|_____|_____|_____|_____|_____|
C033:              *               ^^^^^  CLOCKDATA Byte ^^^^^

C033:              * CLOCKDATA bits defined as follows
C033:              *    bit 7,6,5,4,3,2,1,0 -- Data passed to/from clock chip

C033:00            CLOCKDATA DFB   0           ;Clock data register

C034:              * _7_____6_____5_____4_____3_____2_____1_____0__
C034:              *|      |      |      |      |      |      |      |      |
C034:              *| Clock| Read/|Chip  |      |                           |
C034:              *| xfer |Write |enable|Unused|        Border Color       |
C034:              *|      | chip |assert|      |                           |
C034:              *|_____|_____|_____|_____|_____|_____|_____|_____|
C034:              *               ^^^^^  CLOCKCTL Byte ^^^^^

C034:              * CLOCKCTL bits defined as follows
C034:              *    bit 7= Set =1 to start transfer to clock
C034:              *           Read =0 when transfer to clock is complete
C034:              *    bit 6= 0= write to clock chip -- 1= read from clock chip
C034:              *    bit 5= Clock chip enable asserted after transfer 0=no 1=yes
C034:              *    bit 4= unused
C034:              *    bit 3,2,1,0 = select border color (see TBCOLOR for values)

C034:00            CLOCKCTL  DFB   0           ;Clock control register

C035:              * _7_____6_____5_____4_____3_____2_____1_____0__
C035:              *|      |      |      |      |      |      |      |      |
C035:              *|      | Stop |      | Stop | Stop | Stop | Stop | Stop |
C035:              *|   0  |I/O/LC|   0  |auxh-r|suprhr|hires2|hires1|txt pg|
C035:              *|      |shadow|      |shadow|shadow|shadow|shadow|shadow|
C035:              *|_____|_____|_____|_____|_____|_____|_____|_____|
C035:              *               ^^^^^  SHADOW Byte ^^^^^

C035:              * SHADOW bits defined as follows
C035:              *    bit 7= Must write 0
C035:              *    bit 6= 1 to inhibit I/O and language card operation
C035:              *    bit 5= Must write 0
C035:              *    bit 4= 1 to inhibit shadowing aux hi-res page
C035:              *    bit 3= 1 to inhibit shadowing 32k video buffer
C035:              *    bit 2= 1 to inhibit shadowing hires page 2
C035:              *    bit 1= 1 to inhibit shadowing hires page 1
C035:              *    bit 0= 1 to inhibit shadowing text pages

C035:00            SHADOW    DFB   0           ;Shadow register

C036:              * _7_____6_____5_____4_____3_____2_____1_____0__
C036:              *|      |      |      |      |      |      |      |      |
C036:              *| Slow/|      |      |Shadow|Slot 7|Slot 6|Slot 5|Slot 4|
C036:              *| fast |   0  |   0  |in all|motor |motor | motor| motor|
C036:              *| speed|      |      | RAM  |detect|detect|detect|detect|
C036:              *|_____|_____|_____|_____|_____|_____|_____|_____|
C036:              *               ^^^^^  CYAREG Byte ^^^^^
```

```
C036:              461 * CYAREG bits defined as follows
C036:              462 *    bit 7= 0=slow system speed -- 1=fast system speed
C036:              463 *    bit 6= Must write 0
C036:              464 *    bit 5= Must write 0
C036:              465 *    bit 4= Shadow in all RAM banks
C036:              466 *    bit 3= Slot 7 disk motor on detect
C036:              467 *    bit 2= Slot 6 disk motor on detect
C036:              468 *    bit 1= Slot 5 disk motor on detect
C036:              469 *    bit 0= Slot 4 disk motor on detect

C036:00            471 CYAREG    DFB   0          ;Speed and motor on detect
C037:00            472 DMAREG    DFB   0          ;Used during DMA as bank address
C038:00            473 SCCBREG   DFB   0          ;SCC channel B cmd register
C039:00            474 SCCAREG   DFB   0          ;SCC channel A cmd register
C03A:00            475 SCCBDATA  DFB   0          ;SCC channel B data register
C03B:00            476 SCCADATA  DFB   0          ;SCC channel A data register

C03C:              478 * _7_____6_____5_____4_____3_____2_____1_____0__
C03C:              479 *|      |      |      |      |      |      |      |
C03C:              480 *| Busy | Auto |Access|      |                    |
C03C:              481 *| flag | inc  | doc/ |   0  |        Volume DAC  |
C03C:              482 *|      |adrptr| RAM  |      |                    |
C03C:              483 *|_____|_____|_____|_____|_____|_____|_____|_____|
C03C:              484 *              ^^^^^  SOUNDCTL Byte ^^^^^

C03C:              486 * SOUNDCTL bits defined as follows
C03C:              487 *    bit 7= 0 if not busy -- 1 if busy
C03C:              488 *    bit 6= 0=disable auto incrementing of address pointer
C03C:              489 *           1=enable auto incrementing of address pointer
C03C:              490 *    bit 5= 0=access doc -- 1=access RAM
C03C:              491 *    bit 4= Must be 0
C03C:              492 *    bit 3,2,1,0= volume DAC $0=off $F=full volume

C03C:00            494 SOUNDCTL  DFB   0          ;Sound control register

C03D:              496 * _7_____6_____5_____4_____3_____2_____1_____0__
C03D:              497 *|      |      |      |      |      |      |      |
C03D:              498 *|                                                |
C03D:              499 *|               Sound data read/written          |
C03D:              500 *|                                                |
C03D:              501 *|_____|_____|_____|_____|_____|_____|_____|_____|
C03D:              502 *           ^^^^^  SOUNDDATA Byte ^^^^^

C03D:              504 * SOUNDDATA bits defined as follows
C03D:              505 *    bit 7,6,5,4,3,2,1,0 = Data read from/written to sound RAM

C03D:00            507 SOUNDDATA DFB   0          ;Sound data register

C03E:              509 * _7_____6_____5_____4_____3_____2_____1_____0__
C03E:              510 *|      |      |      |      |      |      |      |
C03E:              511 *|                                                |
C03E:              512 *|            Low byte of sound address pointer   |
C03E:              513 *|                                                |
C03E:              514 *|_____|_____|_____|_____|_____|_____|_____|_____|
C03E:              515 *           ^^^^^  SOUNDADRL Byte ^^^^^

C03E:              517 * SOUNDADRL bits defined as follows
```

```
C03E:           518 *    bit 7,6,5,4,3,2,1,0 = Address into sound RAM low byte

C03E:00         520 SOUNDADRL DFB   0          ;Sound address pointer, low byte

C03F:           522 * _7_____6_____5_____4_____3_____2_____1_____0__
C03F:           523 *|      |     |     |     |     |     |     |     |
C03F:           524 *|                                                 |
C03F:           525 *|           High byte of sound address pointer    |
C03F:           526 *|                                                 |
C03F:           527 *|_____|_____|_____|_____|_____|_____|_____|_____|
C03F:           528 *            ^^^^^  SOUNDADRH Byte ^^^^^

C03F:           530 * SOUNDADRH bits defined as follows
C03F:           531 *    bit 7,6,5,4,3,2,1,0 = Address into sound RAM high byte

C03F:00         533 SOUNDADRH DFB   0          ;Sound address pointer, high byte
C040:00         534           DFB   0          ;Reserved for future system expansion

C041:           536 * _7_____6_____5_____4_____3_____2_____1_____0__
C041:           537 *|      |     |     |     |     |     |     |     |
C041:           538 *|      |     |     |Enable|Enable|Enable|Enable|Enable|
C041:           539 *|  0   |  0  |  0  |1/4sec| VBL  |switch| move | mouse|
C041:           540 *|      |     |     | ints | ints | ints | ints |     |
C041:           541 *|_____|_____|_____|_____|_____|_____|_____|_____|
C041:           542 *              ^^^^^  INTEN Byte ^^^^^

C041:           544 * INTEN bits defined as follows
C041:           545 *    bit 7= Must be 0
C041:           546 *    bit 6= Must be 0
C041:           547 *    bit 5= Must be 0
C041:           548 *    bit 4= 1 to enable 1/4 second interrupts
C041:           549 *    bit 3= 1 to enable VBL interrupts
C041:           550 *    bit 2= 1 to enable Mega // mouse switch interrupts
C041:           551 *    bit 1= 1 to enable Mega // mouse movement interrupts
C041:           552 *    bit 0= 1 to enable Mega // mouse operation

C041:00         554 INTEN     DFB   0          ;Interrupt enable register
C042:00         555           DFB   0          ;Reserved for future system expansion
C043:00         556           DFB   0          ;Reserved for future system expansion

C044:           558 * _7_____6_____5_____4_____3_____2_____1_____0__
C044:           559 *|      |     |     |     |     |     |     |     |
C044:           560 *|                                                 |
C044:           561 *|           Mega // mouse delta movement byte     |
C044:           562 *|                                                 |
C044:           563 *|_____|_____|_____|_____|_____|_____|_____|_____|
C044:           564 *              ^^^^^  MMDELTAX Byte ^^^^^

C044:           566 * MMDELTAX bits defined as follows
C044:           567 *    bit 7,6,5,4,3,2,1,0 = delta movement in 2's complement notation

C044:00         569 MMDELTAX  DFB   0          ;Mega // mouse delta X register

C045:           571 * _7_____6_____5_____4_____3_____2_____1_____0__
C045:           572 *|      |     |     |     |     |     |     |     |
C045:           573 *|                                                 |
C045:           574 *|           Mega // mouse delta movement byte     |
C045:           575 *|                                                 |
```

```
C045:              576 *|_____|_____|_____|_____|_____|_____|_____|_____|
C045:              577 *              ^^^^^  MMDELTAY Byte ^^^^^

C045:              579 * MMDELTAY bits defined as follows
C045:              580 *    bit 7,6,5,4,3,2,1,0 = delta movement in 2's complement notation

C045:00            582 MMDELTAY  DFB   0          ;Mega // mouse delta Y register

C046:              584 * _7_____6_____5_____4_____3_____2_____1_____0___
C046:              585 *|      |      |      |      |      |      |      |      |
C046:              586 *|Self/ |MMouse|Status|Status|Status|Status|Status|Status|
C046:              587 *|burnin| last |  AN3 |1/4sec| VBL  |switch| move |system|
C046:              588 *| diags|button|      | int  | int  | int  | int  | IRQ  |
C046:              589 *|_____|_____|_____|_____|_____|_____|_____|_____|
C046:              590 *              ^^^^^  DIAGTYPE Byte ^^^^^

C046:              592 * DIAGTYPE bits defined as follows
C046:              593 *    bit 7= 0 if self diagnostics get used if BUTN0=1 and BUTN1=1
C046:              594 *    bit 7= 1 if burn-in diagnostics get used if BUTN0=1 and BUTN1=1
C046:              595 *    bits 6-0 = same as INTFLAG

C046:              597 * _7_____6_____5_____4_____3_____2_____1_____0___
C046:              598 *|      |      |      |      |      |      |      |      |
C046:              599 *|MMouse|MMouse|Status|Status|Status|Status|Status|Status|
C046:              600 *|  now | last |  AN3 |1/4sec| VBL  |switch| move |system|
C046:              601 *|button|button|      | int  | int  | int  | int  | IRQ  |
C046:              602 *|_____|_____|_____|_____|_____|_____|_____|_____|
C046:              603 *              ^^^^^  INTFLAG Byte ^^^^^

C046:              605 * INTFLAG bits defined as follows
C046:              606 *    bit 7= 1 if mouse button currently down
C046:              607 *    bit 6= 1 if mouse button was down on last read
C046:              608 *    bit 5= status of AN3
C046:              609 *    bit 4= 1 if 1/4 second interrupted
C046:              610 *    bit 3= 1 if VBL interrupted
C046:              611 *    bit 2= 1 if Mega // mouse switch interrupted
C046:              612 *    bit 1= 1 if Mega // mouse movement interrupted
C046:              613 *    bit 0= 1 if system IRQ line is asserted

C046:        C046  615 DIAGTYPE  EQU   *          ;0/1 Self/burn-in diagnostics
C046:00            616 INTFLAG   DFB   0          ;Interrupt flag register
C047:00            617 CLRVBLINT DFB   0          ;Clear the VBL and 3.75Hz interrupt flags
C048:00            618 CLRXYINT  DFB   0          ;Clear Mega // mouse interrupt flags
C049:00            619           DFB   0          ;Reserved for future system expansion
C04A:00            620           DFB   0          ;Reserved for future system expansion
C04B:00            621           DFB   0          ;Reserved for future system expansion
C04C:00            622           DFB   0          ;Reserved for future system expansion
C04D:00            623           DFB   0          ;Reserved for future system expansion
C04E:00            624           DFB   0          ;Reserved for future system expansion
C04F:00            625           DFB   0          ;Reserved for future system expansion
C050:00            626 TXTCLR    DFB   0          ;switch in graphics (not text)
C051:00            627 TXTSET    DFB   0          ;switch in text (not graphics)
C052:00            628 MIXCLR    DFB   0          ;clear mixed-mode
C053:00            629 MIXSET    DFB   0          ;set mixed-mode (4 lines text)
C054:00            630 TXTPAGE1  DFB   0          ;switch in text page 1
C055:00            631 TXTPAGE2  DFB   0          ;switch in text page 2
C056:00            632 LORES     DFB   0          ;low-resolution graphics
C057:00            633 HIRES     DFB   0          ;high-resolution graphics
```

```
C058:00          634 SETAN0    DFB   0          ;Clear annunciator 0
C059:00          635 CLRAN0    DFB   0          ;Set annunciator 0
C05A:00          636 SETAN1    DFB   0          ;Clear annunciator1
C05B:00          637 CLRAN1    DFB   0          ;Set annunciator 1
C05C:00          638 SETAN2    DFB   0          ;Clear annunciator 2
C05D:00          639 CLRAN2    DFB   0          ;Set annunciator 2
C05E:00          640 SETAN3    DFB   0          ;Clear annunciator 3
C05F:00          641 CLRAN3    DFB   0          ;Set annunciator 3
C060:00          642 BUTN3     DFB   0          ;Read switch 3
C061:00          643 BUTN0     DFB   0          ;open apple key
C062:00          644 BUTN1     DFB   0          ;closed apple key
C063:00          645 BUTN2     DFB   0          ;Read switch 2
C064:00          646 PADDL0    DFB   0          ;Read paddle 0
C065:00          647           DFB   0          ;Read paddle 1
C066:00          648           DFB   0          ;Read paddle 2
C067:00          649           DFB   0          ;Read paddle 3

C068:            651 * _7_____6_____5_____4_____3_____2_____1_____0__
C068:            652 *|      |      |      |      |      |      |      |      |
C068:            653 *| ALZP | PAGE2| RAMRD|RAMWRT| RDROM|LCBNK2|ROMBNK| INTCX|
C068:            654 *|status|status|status|status|status|status|status|status|
C068:            655 *|      |      |      |      |      |      |      |      |
C068:            656 *|_____|_____|_____|_____|_____|_____|_____|_____|
C068:            657 *           ^^^^^  STATEREG Status Byte ^^^^^

C068:            659 * STATEREG bits defined as follows
C068:            660 *    bit 7= ALZP status
C068:            661 *    bit 6= PAGE2 status
C068:            662 *    bit 5= RAMRD status
C068:            663 *    bit 4= RAMWRT status
C068:            664 *    bit 3= RDROM status (Read only RAM/ROM (0/1))

C068:            666 * IMPORTANT NOTE:
C068:            667 *          Do two reads to $C083 then change STATEREG
C068:            668 *          to change LCRAM/ROM banks (0/1) and still
C068:            669 *          have the language card write enabled.

C068:            671 *    bit 2= LCBNK2 status 0=LC bank 0 - 1=LC bank 1
C068:            672 *    bit 1= ROMBANK status
C068:            673 *    bit 0= INTCXROM status

C068:00          675 STATEREG  DFB   0          ;State register
C069:00          676           DFB   0          ;Reserved for future system expansion
C06A:00          677           DFB   0          ;Reserved for future system expansion
C06B:00          678           DFB   0          ;Reserved for future system expansion
C06C:00          679           DFB   0          ;Reserved for future system expansion
C06D:00          680 TESTREG   DFB   0          ;Test mode bit register
C06E:00          681 CLRTM     DFB   0          ;Clear test mode
C06F:00          682 ENTM      DFB   0          ;Enable test mode
C070:00          683 PTRIG     DFB   0          ;trigger the paddles
C071:      000F  684           DS    15,0       ;ROM interrupt code jump table

C080:00          686           DFB   0          ;Sel LC RAM bank2 rd, wrt protect LC RAM
C081:00          687 ROMIN     DFB   0          ;Enable ROM read, 2 reads write enb LC RAM
C082:00          688           DFB   0          ;Enable ROM read, wrt protect LC RAM
C083:00          689 LCBANK2   DFB   0          ;Sel LC RAM bank2, 2 rds wrt enb LC RAM

C084:00          691           DFB   0          ;Sel LC RAM bank2 rd, wrt protect LC RAM
```

```
C085:00            692          DFB   0          ;Enable ROM read, 2 reads wrt enb LC RAM
C086:00            693          DFB   0          ;Enable ROM read, wrt protect LC RAM
C087:00            694          DFB   0          ;Sel LC RAM bank2, 2 rds wrt enb LC RAM

C088:00            696          DFB   0          ;Sel LC RAM bank1 rd, wrt protect LC RAM
C089:00            697          DFB   0          ;Enable ROM read, 2 reads wrt enb LC RAM
C08A:00            698          DFB   0          ;Enable ROM read, wrt protect LC RAM
C08B:00            699 LCBANK1  DFB   0          ;Sel LC RAM bank1, 2 rds wrt enb LC RAM

C08C:00            701          DFB   0          ;Sel LC RAM bank1 rd, wrt protect LC RAM
C08D:00            702          DFB   0          ;Enable ROM read, 2 reads wrt enb LC RAM
C08E:00            703          DFB   0          ;Enable ROM read, wrt protect LC RAM
C08F:00            704          DFB   0          ;Sel LC RAM bank1, 2 rds wrt enb LC RAM

0000:              706          DEND

0000:       CFFF   708 CLRROM   EQU   $CFFF      ;switch out $C8 ROMs


0000:              710 * Applesoft equates

0000:       E000   712 BASIC    EQU   $E000      ;BASIC entry point
0000:       E003   713 BASIC2   EQU   $E003      ;BASIC warm entry point


0000:              715 * Permanent data in screenholes
0000:              716 *
0000:              717 * Note: these screenholes are only used by
0000:              718 * the 80 column firmware if an 80 column card
0000:              719 * is detected or if the user explicitly activates
0000:              720 * the firmware.  If the 80 column card is not
0000:              721 * present, only MODE is trashed on RESET.
0000:              722 *
0000:              723 * The success of these routines rely on the
0000:              724 * fact that if 80 column store is on (as it
0000:              725 * normally is during 80 column operation), that
0000:              726 * text page 1 is switched in.  Do not call the
0000:              727 * video firmware if video page 2 is switched in!!

0000:       07F8   729 MSLOT    EQU   $7F8       ;=$Cn ;n=slot using $C800

0000:       047B   731 OLDCH    EQU   $478+3     ;LAST CH used by video firmware
0000:       04FB   732 MODE     EQU   $4F8+3     ;video firmware operating mode
0000:       057B   733 OURCH    EQU   $578+3     ;80 column CH
0000:       05FB   734 OURCV    EQU   $5F8+3     ;80 column CV
0000:       067B   735 CHAR     EQU   $678+3     ;character to be printed/read
0000:       06FB   736 XCOORD   EQU   $6F8+3     ;GOTOXY X-coord (pascal only)
0000:       077B   737 TEMP1    EQU   $778+3     ;temp
0000:       077B   738 OLDBASL  EQU   $778+3     ;last BASL (pascal only)
0000:       07FB   739 OLDBASH  EQU   $7F8+3     ;last BASH (pascal only)

0000:              741 *
0000:              742 *
0000:              743 * BASIC VMODE BITS
0000:              744 *
0000:              745 * 0....... - BASIC active
0000:              746 * 1....... - Pascal active
0000:              747 * .0...... - Print mouse chars
```

```
0000:                748 * .1...... - Don't print mouse characters
0000:                749 * ..0..... - Print control characters
0000:                750 * ..1..... - Don't print ctrl chars
0000:                751 * ...0.... -
0000:                752 * ...1.... -
0000:                753 * ....0... - Print control characters
0000:                754 * ....1... - Don't print ctrl chars.
0000:                755 * .....0.. -
0000:                756 * .....1.. -
0000:                757 * ......0. - Scroll normally
0000:                758 * ......1. - 1200 baud: home instead of scroll
0000:                759 * .......0 - Mouse text inactive
0000:                760 * .......1 - Mouse text active
0000:                761 *
0000:         0040   762 M.6       EQU   $40         ;
0000:         0020   763 M.CTL2    EQU   $20         ;Don't print controls
0000:         0010   764 M.4       EQU   $10         ;
0000:         0008   765 M.CTL     EQU   $08         ;Don't print controls
0000:         0004   766 M.2       EQU   $04         ;
0000:         0002   767 M.1       EQU   $02         ;
0000:         0001   768 M.MOUSE   EQU   $01         ;Don't print mouse chars
0000:                769 *
0000:                770 * Pascal Mode Bits
0000:                771 *
0000:                772 * 0.......  - BASIC active
0000:                773 * 1.......  - Pascal active
0000:                774 * .0......  - Print mouse chars
0000:                775 * .1......  - Don't print mouse characters
0000:                776 * ..0.....  -
0000:                777 * ..1.....  -
0000:                778 * ...0....  - Cursor always on
0000:                779 * ...1....  - Cursor always off
0000:                780 * ....0...  - GOTOXY n/a
0000:                781 * ....1...  - GOTOXY in progress
0000:                782 * .....0..  - Normal Video
0000:                783 * .....1..  - Inverse Video
0000:                784 * ......0.  - Scroll normally
0000:                785 * ......1.  - 1200 baud: home instead of scroll
0000:                786 * .......0  -
0000:                787 * .......1  -
0000:                788 *
0000:         0080   789 M.PASCAL  EQU   $80         ;Pascal active
0000:         0010   790 M.CURSOR  EQU   $10         ;Don't print cursor
0000:         0008   791 M.GOXY    EQU   $08         ;GOTOXY IN PROGRESS
0000:         0004   792 M.VMODE   EQU   $04
0000:         0002   793 M.PAS1.0  EQU   $02         ;Pascal 1.0 mode


0000:                  7           INCLUDE BANKFF.EQUATES2
```

```
0000:                4 ********************************************************
0000:                5 *                                                      *
0000:                6 *                  Bank $FF Variables2                 *
0000:                7 *                                                      *
0000:                8 *                       by                             *
0000:                9 *                    Fern Bachman                      *
0000:               10 *                                                      *
0000:               11 *         Copyright Apple Computer, Inc. 1985          *
0000:               12 *               All Rights Reserved.                   *
0000:               13 *                                                      *
0000:               14 ********************************************************

0000:        0000   16 INITDISMGR EQU  $FE0000     ;Initialize DISPATCH handler
0000:        7400   17 DIAGS     EQU  $FF7400     ;Diagnostics start here
0000:        5A00   18 PROCONINIT EQU  $FF5A00     ;Initialize protocol converter

0000:               20 * Bank $E1 equates

0000:        1E00   22 THBUF     EQU  $E11E00     ;Type ahead buffer (NOT IMPLEMENTED)

0000:               24           DSECT
0000:        00E1   25           SEG  $E1
02C0:        02C0   26           ORG  $02C0       ;1 page image of BATTERYRAM
02C0:        02C0   27 BRAMBUFFER EQU  *

02C0:               29 * Port 1 selections

02C0:00             31 USERPORT1 DFB  $00         ;This port is a: Printer
02C1:00             32           DFB  $00         ;Line Length: Unlimited
02C2:00             33           DFB  $00         ;Delete First LF after CR: No
02C3:01             34           DFB  $01         ;Add LF after CR: Yes
02C4:00             35           DFB  $00         ;Echo: No
02C5:00             36           DFB  $00         ;Buffering: No
02C6:0D             37           DFB  $0D         ;Baud: 9600
02C7:07             38           DFB  $07         ;Data/Stop Bits: 8/2
02C8:02             39           DFB  $02         ;Parity: None
02C9:01             40           DFB  $01         ;DCD Handshake: Yes
02CA:01             41           DFB  $01         ;DSR Handshake: Yes
02CB:00             42           DFB  $00         ;XON/XOFF Handshake: No

02CC:               44 * Port 2 selections

02CC:01             46 USERPORT2 DFB  $01         ;This port is a: Modem
02CD:00             47           DFB  $00         ;Line Length: Unlimited
02CE:00             48           DFB  $00         ;Delete First LF after CR: No
02CF:00             49           DFB  $00         ;Add LF after CR: No
02D0:00             50           DFB  $00         ;Echo: No
02D1:00             51           DFB  $00         ;Buffering: No
02D2:07             52           DFB  $07         ;Baud: 1200
02D3:06             53           DFB  $06         ;Data/Stop Bits: 8/1
02D4:02             54           DFB  $02         ;Parity: None
02D5:01             55           DFB  $01         ;DCD Handshake: Yes
02D6:01             56           DFB  $01         ;DSR Handshake: Yes
02D7:00             57           DFB  $00         ;XON/XOFF Handshake: No
```

```
02D8:                59 * Display selections

02D8:01              61 USERCOLOR DFB   $01        ;Type: Monochrome
02D9:00              62 USERCOL   DFB   $00        ;Columns: 40
02DA:01              63 USERHERTZ DFB   $01        ;Hertz: 60
02DB:0F              64 USERTEXT  DFB   $0F        ;Text: (white)
02DC:06              65 USERBACK  DFB   $06        ;Background: (medium blue)
02DD:05              66 USERBORDER DFB  $05        ;Border: (dark gray)

02DE:                68 * Sound selections

02DE:08              70 USERVOL   DFB   $08        ;Volume: |-------*-------|
02DF:08              71 USERBELL  DFB   $08        ;Chimes: |-------*-------|

02E0:                73 * Speed selections

02E0:00              75 USERSPEED DFB   $00        ;Speed: Normal

02E1:                77 * Slots' selections

02E1:00              79 USERSLT1  DFB   $00        ;Slot 1: Port 1
02E2:00              80 USERSLT2  DFB   $00        ;Slot 2: Port 2
02E3:00              81 USERSLT3  DFB   $00        ;Slot 3: Text Display
02E4:00              82           DFB   $00        ;Slot 4: Mouse
02E5:00              83           DFB   $00        ;Slot 5: Smart Port
02E6:00              84           DFB   $00        ;Slot 6: Disk Port
02E7:01              85 USERSLT7  DFB   $01        ;Slot 7: Your Card
02E8:00              86 USERSLTST DFB   $00        ;Startup Slot: Scan

02E9:                88 * Language selections

02E9:00              90 USERDISP  DFB   $00        ;Text Display: English (U.S.)
02EA:00              91 USERLANG  DFB   $00        ;Keyboard: English (U.S.)
02EB:00              92 USERKBUFF DFB   $00        ;Keyboard Buffering: No
02EC:03              93 USERRSPD  DFB   $03        ;Repeat Speed: |-------*-------|
02ED:02              94 USERRDLY  DFB   $02        ;Repeat Delay: |-----*---------|
02EE:02              95           DFB   $02        ;Double Click: |-----*---------|
02EF:02              96           DFB   $02        ;Cursor Flash: |-----*---------|
02F0:00              97 USERCAPS  DFB   $00        ;Shift Caps/Lower Case: No
02F1:00              98 USERFSPDE DFB   $00        ;Fast Space/Delete Keys: No
02F2:00              99 USERDUAL  DFB   $00        ;Dual Speed: Normal
02F3:00             100 USERHMR   DFB   $00        ;High Mouse Resolution: No

02F4:               102 * Clock selections

02F4:00             104 USERDATE  DFB   $00        ;Format: MM/DD/YY
02F5:00             105 USERAMPM  DFB   $00        ;Format: AM-PM

02F6:               107 * Ramdisk selections

02F6:00             109 USERMRAM  DFB   $00        ;Minimum RAM for Ramdisk option number
02F7:00             110 USERXRAM  DFB   $00        ;Maximum RAM for Ramdisk option number

02F8:       0009    112 LANGS     DS    9,0        ;Count/Number of languages

0301:       0011    114 LAYOUTS   DS    17,0       ;Count/Numbers of layouts

0312:       002E    116           DS    $2C0+$80-*,$00 ;Pad to AppleTalk node number
```

```
0340:00          118 APTALKNODE DFB  0          ;AppleTalk node number

0341:      0020  120 OSVARIABLES DS  32,0       ;Operating system variables

0361:      005B  122           DS   $2C0+$FC-*,$00 ;Pad to check sums

03BC:00 00       124 BRAMCHKSUM1 DW  0          ;Main BATTERYRAM checksum
03BE:00 00       125 BRAMCHKSUM2 DW  0          ;= BRAMCHKSUM1 EOR BRAMCNST

0000:            127           DEND
```

```
0000:              129          DSECT
0000:        00E1  130          SEG   $E1
0000:        0000  131          ORG   $0000       ;Monitor $E1 permanent storage area

0000:              133 * Important vectors the user may change
0000:              134 * RAM space from locations $0000->$02B7 are allocated for
0000:              135 * monitor and RAM vectors use.

0000:        0004  137 DISPATCH1 DS    4,0        ;Jmp vector to Tool Locator1
0004:        0004  138 DISPATCH2 DS    4,0        ;Jmp vector to Tool Locator2
0008:        0004  139 UDISPATCH1 DS   4,0        ;Jmp vector to User's tool locator1
000C:        0004  140 UDISPATCH2 DS   4,0        ;Jmp vector to User's tool locator2
0010:        0004  141 INTMGRV   DS    4,0        ;Jmp vector to Interrupt manager
0014:        0004  142 COPMGRV   DS    4,0        ;Jmp vector to COP manager
0018:        0004  143 ABORTMGRV DS    4,0        ;Jmp vector to ABORT manager
001C:        0004  144 SYSDMGRV  DS    4,0        ;Jmp vector to system death manager

0020:              146 * Tool/internal interrupt vectors

0020:        0004  148 IRQ.APTLK DS    4,0        ;Jmp vector to APTLK interrupt handler
0024:        0004  149 IRQ.SERIAL DS   4,0        ;Jmp vector to serial port int handler
0028:        0004  150 IRQ.SCAN  DS    4,0        ;Jmp vector to scan line interrupt handler
002C:        0004  151 IRQ.SOUND DS    4,0        ;Jmp vector to sound interrupt handler
0030:        0004  152 IRQ.VBL   DS    4,0        ;Jmp vector to VBL interrupt handler
0034:        0004  153 IRQ.MOUSE DS    4,0        ;Jmp vector to mouse interrupt handler
0038:        0004  154 IRQ.QTR   DS    4,0        ;Jmp vector to 1/4 sec interrupt handler
003C:        0004  155 IRQ.KBD   DS    4,0        ;Jmp vector to keyboard interrupt handler
0040:        0004  156 IRQ.RESPONSE DS 4,0        ;Jmp vctr to FDB response byte int hndlr
0044:        0004  157 IRQ.SRQ   DS    4,0        ;Jmp vector to SRQ int handler
0048:        0004  158 IRQ.DSKACC DS   4,0        ;Jmp vector to desk accessory int handler
004C:        0004  159 IRQ.FLUSH DS    4,0        ;Jmp vector to flush buffer int handler
0050:        0004  160 IRQ.MICRO DS    4,0        ;Jmp vector to key micro int handler
0054:        0004  161 IRQ.1SEC  DS    4,0        ;Jmp vector to 1 second interrupt handler
0058:        0004  162 IRQ.EXT   DS    4,0        ;Jmp vector to EXT VGC interrupt handler
005C:        0004  163 IRQ.OTHER DS    4,0        ;Jmp vector to other interrupt handler

0060:        0004  165 CUPDATE   DS    4,0        ;Jmp vector to cursor update handler

0064:        0004  167 INCBUSYFLG DS   4,0        ;Jmp vector to increment BUSYFLAG routine
0068:        0004  168 DECBUSYFLG DS   4,0        ;Jmp vector to decrement BUSYFLAG routine
006C:        0004  169 RDBUSYFLG DS    4,0        ;Jmp vector to read BUSYFLAG routine

0070:        0010  171 FILL2MVCTRS EQU $80-*
0070:        0010  172          DS    FILL2MVCTRS.$00 ;Pad for more vectors


0080:        0060  174 IRQ.LEN   EQU   *-IRQ.APTLK


0080:              176 * End of user changable interrupt vectors.


0080:        0004  178 TOWRITEBR DS    4,0        ;Jmp vector to write BATTERYRAM routine
0084:        0004  179 TOREADBR  DS    4,0        ;Jmp vector to read BATTERYRAM routine
0088:        0004  180 TOWRITETIME DS  4,0        ;Jmp vector to write time routine
008C:        0004  181 TOREADTIME DS   4,0        ;Jmp vector to read time routine
0090:        0004  182 TOCTRL.PANEL DS 4,0        ;Jmp vctr to Control Panel desk accessory
0094:        0004  183 TOBRAMSETUP DS  4,0        ;Jmp vector for ext Menu to setup stuff
```

```
05 B400FF.EQUATES2 Bank $FF Variables2                    23-JUL-87  13:17 PAGE 27

0098:       0004  184 TOPRINTMSG8 DS  4,0        ;Jmp vector to print a msg- 8 bit entry
009C:       0004  185 TOPRINTMSG16 DS 4,0        ;Jmp vector to print a msg- 16 bit entry

00A0:       0008  187 FILL2MSGPTR EQU $A8-*
00A0:       0008  188          DS    FILL2MSGPTR,$00 ;Pad for more vectors

00A8:       0018  190 OSVECTORS DS    24,0       ;Operating system 4 byte jump vectors

00C0:       0003  192 MSGPOINTER DS   3,0        ;Pointer to messages table

00C3:             194 * IRQ.SECFLAG, IRQ.INTFLAG, IRQ.DATAREG are setup
00C3:             195 * during interrupt time.  They are defined as follows
00C3:             196 *
00C3:             197 * IRQ.INTFLAG = value of INTFLAG read during interrupt
00C3:             198 *               processing.
00C3:             199 * IRQ.DATAREG = value of DATAREG read during interrupt
00C3:             200 *               processing.
00C3:             201 * IRQ.KMSTATUS value of KMSTATUS read during interrupt
00C3:             202 *            processing.
00C3:             203 * IRQ.SECFLAG =0 if all interrupts processed by handlers
00C3:             204 *             =<>0 if any interrupt not processed by handlers
00C3:             205 * IRQ.SERIAL1 is set up by serial interrupt manager
00C3:             206 *             for the user to read.
00C3:             207 * IRQ.SERIAL2 is set up by serial interrupt manager
00C3:             208 *             for the user to read.
00C3:             209 * IRQ.APTLKHI is set up by AppleTalk interrupt manager
00C3:             210 *             for the user to read.
00C3:             211 *
00C3:             212 * IRQ.SECFLAG, IRQ.INTFLAG, IRQ.DATAREG, IRQ.KMSTATUS are set
00C3:             213 * to zero in irq.done2.

00C3:       0001  215 IRQ.INTFLAG DS  1,0        ;Status of INTFLAG at interrupt time
00C4:       0001  216 IRQ.DATAREG DS  1,0        ;Status of DATAREG at interrupt time
00C5:       0001  217 IRQ.KMSTATUS DS 1,0        ;Status of KMSTATUS at interrupt time
00C6:       0001  218 IRQ.SECFLAG DS  1,0        ;0/<>0 = int not processed/processed
00C7:       0001  219 IRQ.SERIAL1 DS  1,0        ;Interrupt info for users
00C8:       0001  220 IRQ.SERIAL2 DS  1,0        ;Interrupt info for users
00C9:       0001  221 IRQ.APTLKHI DS  1,0        ;Interrupt info for users
00CA:       0001  222 IRQ.VOLUME DS   1,0        ;Volume used during interrupt time
00CB:       0001  223 IRQ.ACTIVE DS   1,0        ;<>0 if called during irq

00CC:       0003  225 FILL2IOMSK EQU  $CF-*
00CC:       0003  226          DS    FILL2IOMSK,$00 ;Pad for more irq bytes

00CF:       0001  228 SIOFLAG   DS    1,0        ;Slts enabled via init call to Misc Tl Loc
00D0:       0003  229 INMASK    DS    3,0        ;AND mask / OR mask / slot #
00D3:       0003  230 OUTMASK   DS    3,0        ;AND mask / OR mask / slot #

00D6:       0001  232 ENSONIQFLG DS   1,0        ;0=chip installed / 1=no chip installed
00D7:       0001  233 IRQ.MOUSVBL DS  1,0        ;0 if mouse not using VBL
```

```
00D8:       0004  235 HBPTR     DS    4,0         ;Heart beat pointer to 1st item in list
00DC:       0004  236 TICKCNT   DS    4,0         ;Tick count updated by heart beat
00E0:       0001  237 MOUSESLOT DS    1,0         ;Mouse slot in use

00E1:       001E  239 EVMGRDATA DS    30,$00      ;Event manager permanent information

00FF:       0002  241 BUSYFLAG  DS    2,$00       ;System busy flag


0101:       0001  243 SCCSTAT   DS    1,0         ;=0 then not SCC int- <>0 then SCC int
0102:       0001  244 SCCFLAG   DS    1,0         ;0=no ApTalk-$3C=chanA-$3B=chanB=ApTalk
0103:       0001  245 SCCFLAG2  DS    1,0         ;Aptalk int mask -$07=chanB-$38=chanA
0104:       0001  246 SERFLAG   DS    1,0         ;Ser int msk-$07=chanB-$38=chanA-$3F=both
0105:       0001  247 SCC1      DS    1,0         ;1st data byte read in by int handler
0106:       0001  248 SCC2      DS    1,0         ;2nd data byte read in by int handler
0107:       0001  249 SCC3      DS    1,0         ;3rd data byte read in by int handler


0108:             251 * Interrupt uses all variables with 'irq.'
0108:             252 *
0108:             253 * Interrupt buffer is used by monitor to save
0108:             254 * ASCIITIME in during '=T' calls.  Interrupts
0108:             255 * are disabled during this time so it is safe
0108:             256 * to use the same buffer.

0108:       0108  258 ASCIITIME EQU   *           ;Time in ASCII 'mm/dd/yy hh:mm:ss am/pm'
0108:       0002  259 irq.A     DS    2
010A:       0002  260 irq.X     DS    2
010C:       0002  261 irq.Y     DS    2
010E:       0002  262 irq.S     DS    2
0110:       0002  263 irq.D     DS    2
0112:       0001  264 irq.P     DS    1
0113:       0001  265 irq.DB    DS    1           ;irq.DB and irq.e canNOT be separated
0114:       000C  266 IRQLEN    EQU   *-irq.A
0114:       0001  267 irq.e     DS    1
0115:       0001  268 irq.K     DS    1
0116:       0002  269 irq.PC    DS    2
0118:       0001  270 irq.state DS    1
0119:       0002  271 irq.shadow DS   2
011B:       0001  272 irq.mslot DS    1


011C:       0004  274 FILL2MONVAR EQU $120-*
011C:       0004  275           DS    FILL2MONVAR,$00 ;Pad for more interrupt info


0120:00 00        277 AVAL      DW    0           ;A register value
0122:00 00        278           DW    0           ;X register value
0124:00 00        279           DW    0           ;Y register value
0126:00 00        280 SVAL      DW    0           ;S (stack pointer) value
0128:00 00        281 DVAL      DW    0           ;D (direct (zero page) register) value
012A:00           282 PVAL      DFB   0           ;P (processor status) value
012B:00           283 BVAL      DFB   0           ;B (data bank register) value
012C:00           284 KVAL      DFB   0           ;K (program bank register) value
012D:00           285 MVAL      DFB   0           ;M (memory status) value
012E:00           286 QVAL      DFB   0           ;Q (bit7=speed - bits6-0 match shadow reg)
012F:00           287 LVAL      DFB   0           ;L (language card bank 0/1 selected) value
```

```
0130:00            288 MPVAL     DFB   0          ;m memory access mode bit value
0131:00            289 XPVAL     DFB   0          ;x index mode bit value
0132:00            290 EPVAL     DFB   0          ;e (emulation mode status) value
0133:00            291 ASCFILTER DFB   0          ;F (filter mask for ASCII input in mon)

0134:00            293 CURSOR    DFB   0          ;Cursor
0135:00            294 NXTCUR    DFB   0          ;Cursor
0136:00            295 CHGCURFLG DFB   0          ;<>0=change cursor/ 0=no change cursor

0137:00            297 SYSTEMSPD DFB   0          ;$00=slow speed -- $80=high speed mask
0138:00            298 SLOTSFLG  DFB   0          ;SLTROMSEL value for reset
0139:00            299 ORGMEMORY DFB   0          ;Used in executed BRK/COP disassembly

013A:00 00         301 E1PCLH    DW    0          ;Program counter for 'G'o routine
013C:00 00         302 GOS       DW    0          ;'S' value to return to after a 'G'o cmd

013E:00            304 BANKS     DFB   0          ;Source bank
013F:00            305 BANKD     DFB   0          ;Destination bank

0140:00 00         307 EXTENDBNK DW    0          ;Extend+bank bytes that user entered
0142:00            308 DIGITCNT  DFB   0          ;Used by GETNUM - how many digits entered

0143:00            310 TWKEY     DFB   0          ;Storage pointer for type ahead buffer
0144:00            311 TRKEY     DFB   0          ;Retrieve pointer for type ahead buffer
0145:00            312 TYPHED    DFB   0          ;Used by keyboard buffering routine

0146:00 00         314 FIXST1    DW    0          ;Used by FIXSTATEREG routine
0148:00 00         315 FIXST2    DW    0          ;
014A:00 00         316 FIXST3    DW    0          ;

0000:              318           DEND


0000:              320           DSECT
0000:       00E1   321           SEG   $E1
02B8:       02B8   322           ORG   $02B8
02B8:00 00         323 LOWXCLAMP DW    0          ;Low X clamp
02BA:00 00         324 LOWYCLAMP DW    0          ;Low Y clamp
02BC:00 00         325 HIXCLAMP  DW    0          ;High X clamp
02BE:00 00         326 HIYCLAMP  DW    0          ;High Y clamp
0000:              327           DEND


0000:              329           DSECT
0000:       00E1   330           SEG   $E1
03E0:       03E0   331           ORG   $03E0
03E0:00            332 TIMEOUT   DFB   0          ;Timeout counter
03E1:       0004   333 CLKRDATA  DS    4,0        ;Time read from clock in seconds
03E5:       0004   334 CLKWDATA  DS    4,0        ;Time to write to clock in seconds
03E9:       0017   335 TIMEBUFFER DS   23,0       ;Working buffer to time tools
0000:              336           DEND


0000:                8           MSB   ON
```

```
----- NEXT OBJECT FILE NAME IS L.MAIN.O
7400:          7400   11          ORG   $7400
7400:          00FF   12 RUNBANK  EQU   $FF          ;Bank code will run in
7400:          FFFF   13 PEA.BANK  EQU   RUNBANK*16*16+RUNBANK
7400:          00FF   14          SEG   RUNBANK      ;ROM Bank code resides in
7400:          0000   15 BAUD.SWITCH EQU 0           ;0 (No room in rom) See serial test
7400:          0000   16 VGC.SWITCH EQU  0           ;0 (No room in rom) See Custom.Irq file
7400:          0000   17 SEQ.DEBUG EQU   0           ;Normally 0 ( Use DIAGTYPE and NOT SW2 )
7400:          0000   18 CHECKSUM  EQU   0           ;Normally 1, 0 -> skip 128K checksum tst
7400:          0001   19 USE.FDB.TOOL EQU 1          ;Normally 1 to use Bank $FF FDB tool code

7400:80 19     741B   21          BRA   ST.SEQ.B     ;Skip over test pointer table


7402:                 23 ********************************************************************
7402:                 24 *
7402:                 25 * Test pointer table
7402:                 26 * Points to tests in Bank $FF
7402:                 27 * Tests must return with RTL instruction
7402:                 28 * (This allows disk s/w to look up the pointer table and
7402:                 29 *  call the tests from any bank).
7402:                 30 * kg
7402:                 31 ********************************************************************


7402:14               33 TST.TAB   DFB   TST.TAB.E-*-3 :Number of pointers times 2

7403:AE 7C            35          DW    ROM.CHECKSUM  ;Test 1  128K rom
7405:6C 77            36          DW    MOVIRAM       ;Test 2  Ram: Moving Inversions
7407:5A 7A            37          DW    SOFT.SW       ;Test 3  Mega // & Statereg softswitch test
7409:54 7F            38          DW    RAM.ADDR      ;TEST 4  Ram: Addressing
740B:F9 7A            39          DW    FPI.SPEED     ;Test 5  Fpi fast/slow mode check
740D:02 79            40          DW    SER.TST       ;Test 6  Serial Chip
740F:FA 7C            41          DW    CLOCK         ;Test 7  Real Time Clock
7411:D3 7D            42          DW    BAT.RAM       ;Test 8  Battery ram
7413:C9 7E            43          DW    FDB           ;Test 9 Front Desk Bus
7415:5F 7D            44          DW    SHADOW.TST    ;Test 0A Shadow
7417:          0000   45          DO    SEQ.DEBUG     ;Do following 'test' for debug only
 S                    46          DW    TEST1         ;Fails if a key is pressed
7417:                 47          FIN
7417:3E 7B            48          DW    CUSTOM.IRQ    ;Test 0B Interrupts

7419:          7419   50 TST.TAB.E EQU   *           ;End of test pointer table
7419:9F 74            51          DW    EXT.SEQ       ;Pointer for Disk s/w

741B:          FFFF   53 BI.MASK   EQU   %1111111111111111  ;Tests to run if in BI and BTN0 = 1
741B:          FFFF   54 SELF.MASK EQU   %1111111111111111  ;Tests to run in self test
741B:          FF3F   55 BI.SELF.MASK EQU %1111111100111111  ;Tests to run in BI and BTN0 =0

741B:                 57          INCLUDE ROM.SEQ.EQU
```

```
741B:                 3 ********************************************************************
741B:                 4 *
741B:                 5 * Hardware equates:
741B:                 6 *
741B:                 7 * (Game I/O) & B/I pad Signals:
741B:                 8 * Note: We in Cork use Disk phase signals for
741B:                 9 * Final System Test pass/fail & test ongoing/complete
741B:                10 * status. Singapore will use AN0 & AN1 instead.
741B:                11 *
741B:                12 * By Karl Grabe. Apple Computer Inc.
741B:                13 *
741B:                14 ********************************************************************


741B:        0000    16          SEG   $00
741B:        C058    17 PASS.LED  EQU   SETAN0       ;Turn on pass LED
741B:        C0E3    18 PASS.LEDD EQU   $C0E3        ;Phase 1 on disk port hi (for CORK FST)
741B:        C059    19 FAIL.LED  EQU   CLRAN0       ;Turn off pass LED
741B:        C0E2    20 FAIL.LEDD EQU   $C0E2        ;Phase 1 on disk port lo
741B:        C05B    21 POWER.ON  EQU   CLRAN1       ;Turn on power cycle LED (Test in progress)
741B:        C0E4    22 POWER.OND EQU   $C0E4        ;Phase 2 disk port hi (Test in progress)
741B:        C05A    23 POWER.OFF EQU   SETAN1       ;Turn off power cycle LED (Ok to power off: tst fin)
741B:        C0E5    24 POWER.OFFD EQU  $C0E5        ;Phase 2 lo (test finished)
741B:        C063    25 BI.FLAG   EQU   BUTN2        ;True: UUT is in Burn-In (For debuggin,Norm use DIAGTYPE)
741B:        C061    26 SELF      EQU   BUTN0        ;True: we are in self/user diagnostic (NO wr to NVR)
741B:        0361    27 NVR.FREE  EQU   $0361        ;Start of free battery ram space

0000:                29          DSECT              :Zero page vars
0000:                30 *
0000:                31 * ZP variables:
0000:                32 *
0000:        0000    33          SEG   $00
0010:        0010    34          ORG   $10
0010:        0004    35 BAT.RAM.PTR DS  4            :Used by READ.NVR routine
00F9:        00F9    36          ORG   $F9
00F9:        0002    37 ZP.PNTR   DS    2            :Zero page pointer
00FB:        0002    38 TST.PTR   DS    2            ;Points to test
00FD:        0002    39 CHK.BEG   DS    2            :Start address to checksum

00FF:                41 *
00FF:                42 * Non ZP variables:
00FF:                43 *
0300:        0300    44          ORG   $300

0300:        0300    46 NVR.BEG   EQU   *            ;Non vol ram image starts here
0300:        0006    47 BI.STATUS DS    6            :Test Results -- Currently use only 4 bytes
0306:        0002    48 BI.COUNTER DS   2            ;Counts the number of cycles done
0308:        0002    49 BI.PASSES DS    2            ;Number of passes in BI (=Board level Burn-In)
030A:        0002    50 BI.FAILS  DS    2            ;Number of fails
030C:        0002    51 BI.LASTF  DS    2            ;SEQ.COUNTER value at last fail
030E:        0002    52 CYCLES.MAX DS   2            ;Number if test cycles to run in BI (Programmable)
0310:        0003    53 VALID.CHK DS    3            ;3 byte signature to validate next bytes BI.ALT.MASK
0313:        0002    54 BI.ALT.MASK DS  2            ;The PreBI tester can program what tests to run in BI
0315:        0315    55 NVR.END   EQU   *            ;Non vol ram image ends here

0315:        0006    57 TST.STATUS DS   6            ;Temp storage of fail status in BI
031B:        0002    58 BI.FAIL   DS    2            ;True if any test fails during BI sequence (1 byte)
031D:        0002    59 ST.FAIL   DS    2            ;True if a test fails in self test
```

```
031F:       0002   60 TST.MASK  DS   2          ;Two byte test mask - what tests to run
0321:       0002   61 CYCLE.COUNT DS  2          :One byte BI cycle counter: init with CYCLES.MAX
0323:       0002   62 TST.COUNT DS   2          ;Test counter for sequencer (use only one byte)
0325:       0002   63 NO.TESTS  DS   2          :Initial value of TST.COUNT, ie total no of tests
0327:       0002   64 CUR       DS   2          ;"Curser" for ROM.COUT ( use one byte only )
0329:       0002   65 PORT.ON   DS   2          :True for ROM.COUT to print to Port 1

032B:              67 ;
032B:              68 ; For Battery ram tests:
032B:              69 ;
0800:       0800   70          ORG   $0800      ;Disk s/w must avoid this area
0800:       0100   71 BRAM.TEMP DS   256        ;Used to hold contents during tests
0900:       0100   72 BRAM.WORK DS   256        ;Working Bram area
0A00:       0100   73 BRAM.WORK1 DS  256        ;Working Bram area 2
0B00:              74 ;
0B00:              75 ; FDB rom checksum:
0B00:              76 ;
0B00:       0002   77 POINTER   DS   2          ;What FDB up byte ram/rom to read
0B02:       0002   78 RESULT    DS   2          ;Where tool leaves rom/ram byte read
0B04:       0C1C   79 FDB.BUF   DS   3100       ;Read FDB rom code in here NB: 3k Buffer
1720:              80 ;
1720:              81 ; Tool vars in bank $E1  (Must be cleared to zero)
1720:              82 ; See FDB tool source code on following:
1720:       00E1   83          SEG   $E1
0FD6:       0FD6   84          ORG   $0FD6
0FD6:       0001   85 FLAGS     DS   1
0FD7:       0001   86 STATUSTMP DS   1
0FD8:       0001   87 SRQNUM    DS   1
0FD9:       0001   88 SRQCNT    DS   1
0FDA:       0001   89 SRQPTR    DS   1
0FDB:       0001   90 SRQMRQ    DS   1
0FDC:       0000   91          SEG   $00

741B:              93          DEND


741B:              95 ;
741B:              96 ; Serial PORT1.COUT routine:
741B:              97 ;
741B:       0001   98 SLOT      EQU   1          ;Port 1 is in slot 1
741B:       C100   99 SLOT.ADDR EQU   SLOT*16*16+$C000
741B:       C10D  100 INIT.OFF  EQU   SLOT.ADDR+$D  ;Pascal 1.1 init pointer offset
741B:       C10F  101 WRITE.OFF EQU   SLOT.ADDR+$F  ;Pascal 1.1 write pointer offset

741B:       0010  103 INIT.JMP  EQU   $10        ;10-12  ;
741B:       0012  104 WRITE.JMP EQU   $12        ;12-13  :Zero page locations used
741B:       2000  105 SER.ZBANK EQU   $2000      ;Where slot 1 IO rom is called from in zero bank
741B:             106 :
741B:             107 ; Moving Inversions Ram
741B:             108 :
741B:       0004  109 loc       equ   $04
741B:       00F0  110 RamBank   equ   $F0
741B:       00F2  111 RamEndAddr equ  $f2
741B:       00F4  112 RamBegAddr equ  $f4
741B:       0007  113 thedata   equ   $07
741B:       0009  114 mod       equ   $09
741B:       000B  115 flip      equ   $0B
741B:       0316  116 ErrCode1  equ   TST.STATUS+1
```

```
741B:        0317  117 ErrCode2  equ   TST.STATUS+2
741B:              118 ;
741B:              119 ; Serial Test Equates
741B:              120 ;

741B:        BFFD  122 SCCCmd    EQU   $BFFD
741B:        BFFF  123 SCCData   EQU   $BFFF
741B:        003C  124 ChAIndex  EQU   $3C
741B:        003B  125 ChBIndex  EQU   $3B

741B:        0800  127 CNT1      EQU   $0800
741B:        0801  128 CNT2      EQU   $0801
741B:        0802  129 CNT3      EQU   $0802
741B:        0803  130 CNT4      EQU   $0803
741B:        0804  131 STCNT1    EQU   $0804
741B:        0805  132 STCNT2    EQU   $0805
741B:        0806  133 STCNT3    EQU   $0806
741B:        0807  134 STCNT4    EQU   $0807
741B:        0809  135 SerIRQCnt EQU   $0809
741B:        080A  136 ChHsk     EQU   $080A
741B:        080B  137 ChDTR     EQU   $080B
741B:        080C  138 DATA      EQU   $080C
741B:        080D  139 RegNo     EQU   $080D
741B:        080E  140 SerMask   EQU   $080E
741B:        080F  141 MaskData  EQU   $080F
741B:        0810  142 SerIRQtemp1 EQU $0810
741B:        0811  143 SerIRQtemp2 EQU $0811
741B:        0812  144 SerIRQtemp3 EQU $0812

741B:        00BE  146 Zer1      EQU   $BE
741B:        00E2  147 Zer2      EQU   $E2
741B:        00FF  148 Limit     EQU   $FF         ;??????
741B:               58          INCLUDE ROM.SEQ
```

```
741B:                2 ;.............................................................
741B:                3 :
741B:                4 ; Main entry point to Rom Test Sequencer
741B:                5 :
741B:                6 ; Call:    In native
741B:                7 ; Return: Not defined
741B:                8 ;
741B:                9 ; Created  : 07 Jan 86 Karl Grabe
741B:               10 ; Alpha 1.0: 17 Feb 86 KRG
741B:               11 ; Alpha 2.0: 10 Mar 86 KRG
741B:               12 ; Beta 1.0 : 26 Mar 86 KRG
741B:               13 ; Beta 2.0 : 23 Apr 86 KRG
741B:               14 ; Beta 3.0 : 14 May 86  ;OA change FB
741B:               15 ; VER 1.0  : 23 May 86
741B:               16 ;
741B:               17 :
741B:               18 ; This code determines whether the UUT is in Burn-In or Self test
741B:               19 ; and then executes the selected test. See file: ROM.SEQ.INFO
741B:               20 ; for details.
741B:               21 :
741B:               22 ;
741B:               23 :
741B:               24 ;            Copyright Apple Computer Inc. 1986
741B:               25 ;                    All Rights Reserved
741B:               26 ;.............................................................

741B:        741B   28 ST.SEQ.B  EQU   *
741B:78             29           SEI               ;No interrupts
741C:               30           DATA16           ;Set 16 bit memory and index
741C:C2 30           1+          REP   #%00110000  ;SET M=0, X=0 (16 BIT)
741E:                2+          MEMORY16
741E:                3+          INDEX16
741E:                4+          MEND
741E:A9 EB 76       31           LDA   #TST.DEATH.MGR  ;Test death manager ..
7421:8F 1D 00 E1    32           STA   =SYSDMGRV+1  ; .. replaces system death manager
7425:               33           DATA8
7425:E2 30           1+          SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
7427:                2+          MEMORY8
7427:                3+          INDEX8
7427:                4+          MEND
7427:A9 FF          34           LDA   #^TST.DEATH.MGR  ;Bank for test death manager
7429:8F 1F 00 E1    35           STA   =SYSDMGRV+3  ;Save it
742D:20 DF 76       36           JSR   BANK.0       ;Set Data Bank reg = 0
7430:9C 34 C0       37           STZ   CLOCKCTL     ;Initialise border color
7433:9C 21 C0       38           STZ   MONOCOLOR    ;Turn on colour
7436:0E F4 03       39           ASL   PWREDUP      ;Destroy powerup byte (cold boot if test interrupted)
7439:9C 2D C0       40           STZ   SLTROMSEL    ;Enable all internal slots (for Port1 & Disk Port)
743C:A9 04          41           LDA   #$04         ;Not too loud ... for speaker beep at end
743E:8D 3C C0       42           STA   SOUNDCTL     ;To speaker volume DAC

7441:9C 59 C0       44           STZ   FAIL.LED     ;Start by assuming the system is bad
7444:9C E2 C0       45           STZ   FAIL.LEDD    ;Duplicate FAIL.LED in disk port (CORK FST)
7447:9C 5B C0       46           STZ   POWER.ON     ;Tell BI controller NOT to power off!
744A:               47 :                            ;(... so we don't trash NVR)
744A:9C E4 C0       48           STZ   POWER.OND    ;Disk port (For FST in Cork)
744D:               49 :                            ;(for self test: test in progress LED)
744D:               50 ;Find out where we are and do relevant tests ...
744D:        0000   51           DO    SEQ.DEBUG    ;For debugging test code use SW2 instead of DIAGTYPE
```

```
 S                52          LDA   BI.FLAG      ;In Burn In ?
 S                53          BMI   SELF.TST     ;We're not in Burn In so do self test
744D:             54          ELSE
744D:AD 46 C0     55          LDA   DIAGTYPE     ;Read Burn In connector if not in debug
7450:10 48   749A 56          BPL   SELF.TST     ;We're not in Burn In so do self test
7452:             57          FIN
7452:20 B0 76     58          JSR   RD.BI.NVR.88.66  ;Restore NVR to bank zero ram
7455:             59          INDEX16           ;
7455:             60          MEMORY16          ;
7455:AD 0E 03     61          LDA   CYCLES.MAX   ;Max sequence run per BI power on cycle
7458:8D 21 03     62          STA   CYCLE.COUNT  ;BI Cycle counter decrements to 0
745B:             63          DATA8
745B:E2 30         1+         SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
745D:              2+         MEMORY8
745D:              3+         INDEX8
745D:              4+         MEND
745D:AD 61 C0     64          LDA   SELF         ;Self test in Burn In?
7460:30 2E   7490   65          BMI   BI.SELF.TST  ;Yes .. so self and no wr NVR

7462:        7462  67 NXT.BI.CYCLE EQU *          ;Run BI tests again NTV16
7462:              68 ;
7462:              69 ; UUT is in burn in if here.
7462:              70 ; First check if NVR is correctly set by pre bi tester
7462:              71 ; up by checking that the check bytes are intact
7462:              72 ;
7462:              73          DATA16
7462:C2 30         1+         REP   #%00110000  ;SET M=0, X=0 (16 BIT)
7464:              2+         MEMORY16
7464:              3+         INDEX16
7464:              4+         MEND
7464:9C 1B 03     74          STZ   BI.FAIL      ;Zero the fail flag
7467:AD 10 03     75          LDA   VALID.CHK    ;First two (0,1)
746A:CF FD 75 FF  76          CMP   =VALID.MASK  ;Check with what we expect
746E:D0 0E   747E 77          BNE   BAD.NVR      ;Not correct - Stop with self test fail. NVR bad?
7470:AD 11 03     78          LDA   VALID.CHK+1  ;Last two (1,2)
7473:CF FE 75 FF  79          CMP   =VALID.MASK+1  ;What we expect
7477:D0 05   747E 80          BNE   BAD.NVR      ;Not correct - Stop with self test fail. NVR bad?
7479:AD 13 03     81          LDA   BI.ALT.MASK  ;Mask put there by Pre BI tester in NVR
747C:80 21   749F 82          BRA   EXT.SEQ      ;Go do the sequence
747E:        747E 83 BAD.NVR  EQU   *            ;NVR bad or not set up properly
747E:A9 FF FF     84          LDA   #$FFFF       ;AA= FF, BB=FF : NVR NOT SETUP
7481:8D 15 03     85          STA   TST.STATUS   ;Put bad Battery Ram msg on screen.
7484:             86          DATA8
7484:E2 30         1+         SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
7486:              2+         MEMORY8
7486:              3+         INDEX8
7486:              4+         MEND
7486:82 8E 00 7517 87          BRL   FAIL.SELF    ;Goto self test fail routine and die.
7489:        7489  88 BI       EQU   *            ;Normal Burn in tests
7489:             89          DATA16
7489:C2 30         1+         REP   #%00110000  ;SET M=0, X=0 (16 BIT)
748B:              2+         MEMORY16
748B:              3+         INDEX16
748B:              4+         MEND
748B:A9 FF FF     90          LDA   #BI.MASK     ;Tests to run in BI
748E:80 0F   749F 91          BRA   EXT.SEQ
7490:        7490 92 BI.SELF.TST EQU *
7490:             93          DATA16            ;Tests to run in User/Self test
```

```
7490:C2 30          1+        REP   #%00110000  ;SET M=0, X=0 (16 BIT)
7492:               2+        MEMORY16
7492:               3+        INDEX16
7492:               4+        MEND
7492:9C 1B 03       94        STZ   BI.FAIL     ;Zero fail flag
7495:A9 3F FF       95        LDA   #BI.SELF.MASK  ;Tests to run in self test when in BI
7498:80 05   749F   96        BRA   EXT.SEQ     ;Goto sequencer
749A:        749A   97 SELF.TST EQU  *
749A:               98        DATA16
749A:C2 30          1+        REP   #%00110000  ;SET M=0, X=0 (16 BIT)
749C:               2+        MEMORY16
749C:               3+        INDEX16
749C:               4+        MEND
749C:A9 FF FF       99        LDA   #SELF.MASK

749F:              101 ;Be if full 16 bit mode if entering here:
749F:        749F  102 EXT.SEQ  EQU   *           ;Disk s/w can enter here (c= tests to run)
749F:8D 1F 03      103        STA   TST.MASK     ;Save what tests to run
74A2:A9 02 74      104        LDA   #TST.TAB     ;Address of top of pointer table
74A5:85 F9         105        STA   ZP.PNTR      ;Use zero page pointer
74A7:AF 02 74 FF   106        LDA   =TST.TAB     ;No of bytes in table
74AB:29 FF 00      107        AND   #$00FF       ;Drop hi byte
74AE:4A            108        LSR   A            ;Devide by 2 to get number if tests
74AF:8D 23 03      109        STA   TST.COUNT    ;Save as test counter (decrement)
74B2:8D 25 03      110        STA   NO.TESTS     ;Save as number of tests

74B5:        74B5  112 NXT.TST  EQU   *           ;Do next test
74B5:              113        NTV16              ;Get back to ntv16 (2nd ..)
74B5:18             1+        CLC
74B6:FB             2+        XCE                ;SET E = 0 ( Native mode )
74B7:C2 30          3+        REP   #%00110000
74B9:               4+        MEMORY16
74B9:               5+        INDEX16
74B9:               6+        MEND
74B9:A2 FF 01      114        LDX   #$01FF
74BC:9A            115        TXS                ;Set up the stack
74BD:9C 29 03      116        STZ   PORT.ON      ;Tell ROM.COUT not tok write to Port 1
74C0:9C 15 03      117        STZ   TST.STATUS   ;Zero out all fail status regs
74C3:9C 17 03      118        STZ   TST.STATUS+2
74C6:E6 F9         119        INC   ZP.PNTR      ;Point to test pointer
74C8:20 E5 76      120        JSR   BANK.RUN     ;Set Data bank reg to bank code is running in
74CB:B2 F9         121        LDA   (ZP.PNTR)    ;Get test pointer
74CD:20 DF 76      122        JSR   BANK.0       ;Data Bank reg = 0
74D0:85 FB         123        STA   TST.PTR      ;Save it for JMP (TST.PTR) to test
74D2:E6 F9         124        INC   ZP.PNTR      ;Increment for next time round
74D4:38            125        SEC                ;Now get NO.TESTS-TST.COUNT to get test we're on
74D5:AD 25 03      126        LDA   NO.TESTS     ;Total number of tests n-1
74D8:ED 23 03      127        SBC   TST.COUNT    ;Current test number
74DB:1A            128        INC   A            ;Bring into range 1 ... n
74DC:8D 15 03      129        STA   TST.STATUS   ;First byte of status is test number
74DF:22 B2 75 FF   130        JSL   =DO.TEST     ;Do/skip (TST.PTR) test ** NATIVE 16 **
74E3:78            131        SEI                 ;Make sure interrupts still off
74E4:20 DF 76      132        JSR   BANK.0       ;Data Bank reg = 0
74E7:90 7C   7565  133        BCC   TEST.PASS    ;Current test has passed
```

```
74E9:                135 *
74E9:                136 * Test fail handler
74E9:                137 *
74E9:                138 * In Burn-In:
74E9:                139 *    The NVR fail counter and cycle number are incremented
74E9:                140 *    A note is made of the test # that failed + errcodes
74E9:                141 *    The test sequence continues.
74E9:                142 *
74E9:                143 * Self Test:
74E9:                144 *    The test number and fail status is printed on screen:
74E9:                145 *     "AABBCCDD" ( 3 times and staggered ),
74E9:                146 *      where AA is test no and BBCCDD is fail status.
74E9:                147 *
74E9:                148 *    "System Bad: AABBCCDD" message is printed
74E9:                149 *    A high ( 6KHz ) tone is generated.
74E9:                150 *    "AABBCCDD" is sent to serial port 1 at 9600 baud
74E9:                151 *    The NVR is NOT updated.
74E9:                152 *

74E9:          74E9  154 TST.FAIL  EQU   *
74E9:                155           NTV
74E9:18               1+           CLC
74EA:FB               2+           XCE                 :Get into native mode
74EB:                 3+           mend
74EB:                156           DATA8
74EB:E2 30            1+           SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
74ED:                 2+           MEMORY8
74ED:                 3+           INDEX8
74ED:                 4+           MEND
74ED:A9 00           157           LDA   #0
74EF:8F 35 C0 E0     158           STA   $E0C035       :Real shadow location
74F3:A9 FF           159           LDA   #$FF
74F5:8D 1B 03        160           STA   BI.FAIL       :Set BI.FAIL flag to non zero
74F8:20 35 76        161           JSR   ROM.HOME.86   ;Clr txt page returns in 86
74FB:AE 15 03        162           LDX   TST.STATUS    ;Get the test number than failed
74FE:8E 00 03        163           STX   BI.STATUS     ;Save for storage in NVR later (BI only)
7501:AE 17 03        164           LDX   TST.STATUS+2  ;Get detailed fail status
7504:8E 02 03        165           STX   BI.STATUS+2   ;Put fail status aside for NVR later (BI only)
7507:                166           IND8
7507:E2 10            1+           SEP   #%00010000    ;Set X=1 ( 8bit index )
7509:                 2+           INDEX8
7509:                 3+           MEND
7509:20 99 76        167           JSR   PRINT.STATUS88  ;Put fail status on scrn
750C:9C 59 C0        168           STZ   FAIL.LED      ;Tell outside world there's trouble
750F:9C E2 C0        169           STZ   FAIL.LED0     ;Disk port also (Corcaig)
7512:          0000  170           DO    SEQ.DEBUG     ;Use SW2 or DIAGTYPE? (SW2 for debugging)
 S                   171           LDA   BI.FLAG       ;We in BI?
 S                   172           BPL   FAIL.BI.CONT  ;Yes failed current test but continue .....
7512:                173           ELSE
7512:AD 46 C0        174           LDA   DIAGTYPE      ;We in BI?
7515:30 4E    7565   175           BMI   FAIL.BI.CONT  ;Yes failed current test but continue .....
7517:                176           FIN

7517:                178 :If here then UUT has failed self test
7517:A2 00           179 FAIL.SELF LDX   #0
7519:9C 27 03        180           STZ   CUR           :Set curser to zero
751C:9C 5A C0        181           STZ   POWER.OFF     :For LED on Game IO or disk port
751F:9C E5 C0        182           STZ   POWER.OFF0    :Disk port phase (agus Corcaig)
```

```
7522:          00FF  183           SEG   RUNBANK      ;Tell assembler bank $FF for bad message
7522:BF 58 75 FF     184 NXT.FAIL.CHAR LDA =FAIL.SELF.MSG,X
7526:F0 05     752D  185           BEQ   FAIL.SELF.TONE  ;Check for end of msg
7528:20 56 76        186           JSR   ROM.COUT88   ;Print "System Bad message"
752B:80 F5     7522  187           BRA   NXT.FAIL.CHAR  ;For all characters
752D:          752D  188 FAIL.SELF.TONE EQU *
752D:A9 FF           189           LDA   #$FF         ;Non zero
752F:8D 1D 03        190           STA   :ST.FAIL     ;Make a note that a fail occurred.
7532:20 99 76        191           JSR   PRINT.STATUS88  ;Print stagered on screen
7535:20 02 77        192           JSR   INIT.PORT1   ;Setup Port 1 for writing (Cork's process)
7538:A9 FF           193           LDA   #$FF         ;Non zero
753A:8D 29 03        194           STA   :PORT.ON     ;Enable port1
753D:20 99 76        195           JSR   PRINT.STATUS88  ;Print to Port 1
7540:A9 8D           196           LDA   #$8D         ;CR
7542:20 49 77        197           JSR   COUT.PORT1   ;Send Carrage Return to port 1
7545:A9 8A           198           LDA   #$8A         ;LF
7547:20 49 77        199           JSR   COUT.PORT1   ;And a line feed
754A:                200           NTV               ;Make sure we're still in native mode
754A:18                1+          CLC
754B:FB                2+          XCE                ;Get into native mode
754C:                  3+          mend
754C:                201           DATA8
754C:E2 30             1+          SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
754E:                  2+          MEMORY8
754E:                  3+          INDEX8
754E:                  4+          MEND
754E:                202           IND16
754E:C2 10             1+          REP   #%00010000   ;X=0 ( 16 bit index )
7550:                  2+          INDEX16
7550:                  3+          MEND
7550:A2 10 27        203           LDX   #10000       ;No of cycles
7553:A9 05           204           LDA   #5          ;Period for 6 KHz or so
7555:82 AF 00 7607   205           BRL   TONE86       ;Go do tone and die.
7558:          7558  206 FAIL.SELF.MSG EQU *
7558:A0 D3 F9 F3     207           ASC   "           System Bad:"
7564:00              208           DFB   $00
7565:          0000  209           SEG   $00
```

```
7565:                211 *
7565:                212 *
7565:                213 * Test returns here even if test was skipped
7565:                214 *         -- OR --
7565:                215 * if test failed AND UUT is in BI.
7565:                216 *
7565:                217 * Note: carry is clear ('passed') when test skipped
7565:                218 *
7565:          7565  219 FAIL.BI.CONT EQU *          ;Failed test but in BI so continue
7565:          7565  220 TEST.PASS EQU   *          ;Passed test in BI or Self Test so continue
7565:                221           NTV              ;In case test ended up in emulation mode
7565:18              1+            CLC
7566:FB              2+            XCE               ;Get into native mode
7567:                3+            mend
7567:                222           DATA8
7567:E2 30           1+            SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
7569:                2+            MEMORY8
7569:                3+            INDEX8
7569:                4+            MEND
7569:A9 00           223           LDA   #0          ;Turn on all shadowing
756B:8F 35 C0 E0     224           STA   $E0C035     ;Real shadow location
756F:20 99 76        225           JSR   PRINT.STATUS88  ;Show user tests progressing
7572:CE 23 03        226           DEC   TST.COUNT   ;Decrement single byte counter
7575:30 03    757A   227           BMI   END.TST.SEQ ;All tests done
7577:82 3B FF 74B5   228           BRL   NXT.TST     ;Do another test if TST.COUNT not zero
757A:          757A  229 END.TST.SEQ EQU *
757A:          0000  230           DO    SEQ.DEBUG   ;Test SW2 or DIAGTYPE for BI
 S                   231           LDA   BI.FLAG     ;In BI?
 S                   232           BMI   PASS.SELF.STOP88  ;No so unit has passed self test
757A:                233           ELSE
757A:AD 46 C0        234           LDA   DIAGTYPE    ;In BI?
757D:10 50    75CF   235           BPL   PASS.SELF.STOP88  ;No so unit has passed self test
757F:                236           FIN
757F:AD 1B 03        237           LDA   BI.FAIL     ;Check if any test failed
7582:30 20    75A4   238           BMI   BI.CYCLE.FAIL  ;Yes so handle it
7584:9C 58 C0        239           STZ   PASS.LED    ;Last cycle passed all tests
7587:9C E3 C0        240           STZ   PASS.LEDD   ;Disk port also
758A:                241           DATA16
758A:C2 30           1+            REP   #%00110000  ;SET M=0, X=0 (16 BIT)
758C:                2+            MEMORY16
758C:                3+            INDEX16
758C:                4+            MEND
758C:EE 08 03        242           INC   BI.PASSES   ;Record as a passed cycle
758F:          758F  243 CONT.CYC  EQU   *
758F:EE 06 03        244           INC   BI.COUNTER  ;Cumulative counter for ENTIRE time in BI
7592:CE 21 03        245           DEC   CYCLE.COUNT ;Reset to CYCLES.MAX on each BI power up
7595:F0 03    759A   246           BEQ   FIN.CYCLES  ;All BI test cycles complete
7597:82 C8 FE 7462   247           BRL   NXT.BI.CYCLE  ;Continue doing BI cycles

759A:                249 ;Required no of BI cycles complete so stop and wait
759A:          759A  250 FIN.CYCLES EQU  *
759A:20 C5 76        251           JSR   WR.BI.NVR.66.88  ;Save counters status to NVR for next time or tester
759D:                252           INDEX8
759D:                253           MEMORY8
759D:9C 5A C0        254           STZ   POWER.OFF   ;Tell BI controller it's ok to power down.
75A0:9C E5 C0        255           STZ   POWER.OFFD  ; .. And Cork FST controller via disk port
75A3:DB              256           STP               ;Wait 'til power goes out.
75A4:          75A4  257 BI.CYCLE.FAIL EQU *         ;Here if a cycle has failed a test
```

```
75A4:                258          DATA16
75A4:C2 30             1+         REP   #%00110000  ;SET M=0, X=0 (16 BIT)
75A6:                  2+         MEMORY16
75A6:                  3+         INDEX16
75A6:                  4+         MEND
75A6:EE 0A 03        259          INC   BI.FAILS     ;Count it
75A9:AD 06 03        260          LDA   BI.COUNTER
75AC:8D 0C 03        261          STA   BI.LASTF     ;Record what cycle failure occurred
75AF:80 DE   758F    262          BRA   CONT.CYC     ;Go on to next cycle if necessary

75B1:60              264 TST.RTS  RTS                ;Any rts instruction

75B2:          00FF  266          SEG   RUNBANK      ;All tests are normally in Bank $FF
75B2:          75B2  267 DO.TEST  EQU   *            ; ** Native 16 **
75B2:          0000  268          SEG   $00          ;Data and softswitches in Bank $00
75B2:9C 1D 03        269          STZ   ST.FAIL      ;Clear test fail status to pass
75B5:6E 1F 03        270          ROR   TST.MASK     ;Put test 0/1 in carry, 16 bit.
75B8:90 F7   75B1    271          BCC   TST.RTS      ;Skip this test  *** Note carry CLR => passed ***
75BA:                272          DATA8
75BA:E2 30             1+         SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
75BC:                  2+         MEMORY8
75BC:                  3+         INDEX8
75BC:                  4+         MEND
75BC:EE 34 C0        273          INC   :CLOCKCTL    ;Each test has a new border color
75BF:          0000  274          DO    SEQ.DEBUG
 S                   275          JSR   TEMP.WAIT     ;Slow down display while debugging only
75BF:                276          FIN
75BF:20 35 76        277          JSR   ROM.HOME.86  ;Routine sets 86 (8bit memory, 16 bit index)
75C2:                278          DATA8
75C2:E2 30             1+         SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
75C4:                  2+         MEMORY8
75C4:                  3+         INDEX8
75C4:                  4+         MEND
75C4:20 99 76        279          JSR   PRINT.STATUS88  ;Prt tst running (useful if test crashes)
75C7:                280          FAST                ;Hurry things along a little (8 bit macro)
75C7:A9 80             1+         LDA   #%10000000
75C9:8D 36 C0          2+         STA   CYAREG       ;Set fpi fast
75CC:                  3+         MEND
75CC:6C FB 00        281          JMP   (TST.PTR)    ;Go do test ** MEM8 IND8 ** Return with RTL
```

```
75CF:                283 ****************************************************************
75CF:                284 *
75CF:                285 * All following Subroutines are MEMORY8 & INDEX16
75CF:                286 * (subname)86
75CF:                287 * KG JAN 86
75CF:                288 *
75CF:                289 ****************************************************************

75CF:                291 ;.............................................................
75CF:                292 ; PASS.SELF.STOP88
75CF:                293 ; UUT passed self test so put out 1Khz
75CF:                294 ; tone, print message & stop
75CF:                295 ;
75CF:                296 ; MEMX, INDX
75CF:                297 ;.............................................................
75CF:          75CF  298 PASS.SELF.STOP88 EQU *
75CF:20 35 76        299           JSR   ROM.HOME.86  ;Routine sets 86
75D2:                300           DATA8
75D2:E2 30             1+          SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
75D4:                  2+          MEMORY8
75D4:                  3+          INDEX8
75D4:                  4+          MEND
75D4:9C 58 C0        301           STZ   PASS.LED     ;Tell game IO we've passed
75D7:9C E3 C0        302           STZ   PASS.LEDD    ;And disk port
75DA:9C 5A C0        303           STZ   POWER.OFF    ;Tell game IO test finished
75DD:9C E5 C0        304           STZ   POWER.OFFD   ;And disk port
75E0:A2 00           305           LDX   #0
75E2:9C 27 03        306           STZ   CUR
75E5:          00FF  307           SEG   RUNBANK      ;Tell assembler where code resides
75E5:BF F0 75 FF     308 NXT.PASS.CHAR LDA =SELF.PASS.MSG,X
75E9:F0 15     7600  309           BEQ   PASS.SELF.TONE  ;Check for end of msg
75EB:20 56 76        310           JSR   ROM.COUT88   ;Print "System Good message"
75EE:80 F5     75E5  311           BRA   NXT.PASS.CHAR
75F0:          75F0  312 SELF.PASS.MSG EQU *
75F0:A0 D3 F9 F3     313           ASC   "          System Good"
75FC:00              314           DFB   $00
75FD:CB D2 C7        315 VALID.MASK ASC  'KRG'        ;Do not change - used in NVR by tst s/w
7600:          0000  316           SEG   $00
7600:          7600  317 PASS.SELF.TONE EQU *
7600:                318           IND16
7600:C2 10             1+          REP   #%00010000   ;X=0 ( 16 bit index )
7602:                  2+          INDEX16
7602:                  3+          MEND
7602:A2 E8 03        319           LDX   #1000        ;Number of speaker clicks
7605:A9 11           320           LDA   #17          ;Period for 1KHz

7607:                322 ;.............................................................
7607:                323 ; TONE86
7607:                324 ; Generate tone Period: ACC (8)
7607:                325 ;               Cycles: X (16)
7607:                326 ;
7607:                327 ; Enter MEM8,IND16
7607:                328 ; Ret :Does not return
7607:                329 ;.............................................................
7607:                330 TONE86    SLOW
7607:9C 36 C0          1+          STZ   CYAREG       ;Go into slow mode
760A:                  2+          MEND
760A:48              331           PHA                ;Save period
```

```
760B:68            332 NXT.KICK  PLA
760C:48            333           PHA
760D:38            334 KG.WAIT   SEC               ;Copy of monitor wait routine $FCA8
760E:48            335 KG.WAIT2  PHA
760F:E9 01         336 KG.WAIT3  SBC   #$01
7611:D0 FC   760F  337           BNE   KG.WAIT3
7613:68            338           PLA
7614:E9 01         339           SBC   #$01
7616:D0 F6   760E  340           BNE   KG.WAIT2
7618:2C 30 C0      341           BIT   SPKR          ;Give speaker coil a kick
761B:CA            342           DEX                 ;No cycles
761C:D0 ED   760B  343           BNE   NXT.KICK
761E:68            344           PLA                 ;
761F:AD 1D 03      345           LDA   ST.FAIL       ;Has self test failed?
7622:F0 01   7625  346           BEQ   END.SELF      ;No so go wait for OA pressed to repeat Self Test
7624:DB            347           STP                 ;Die here leaving Self Test error message on screen.
7625:        7625  348 END.SELF  EQU   *
7625:0E 46 C0      349           ASL   DIAGTYPE      ;Test if in burn-in or user diags
7628:AD 61 C0      350           LDA   SELF          ;Get Open-Apple status
762B:B0 03   7630  351           BCS   CHKOAONLY     ;In burn-In check Open-Apple only
762D:2D 62 C0      352           AND   BUTN1         ;Else both buttons pressed to continue?
7630:        7630  353 CHKOAONLY EQU   *
7630:10 F3   7625  354           BPL   END.SELF      ;If '+' in any case then keep looping
7632:82 E6 FD 741B 355           BRL   ST.SEQ.B      ;Go to very beginning of diags again.

7635:              357 ;.....................................................
7635:              358 ;
7635:              359 ;ROM.HOME.86
7635:              360 ;Clear the text page & set to 40 text
7635:              361 ;
7635:              362 ; Enter/RET Native
7635:              363 ; Exit Mem8 Ind16
7635:              364 ;.....................................................
7635:        7635  365 ROM.HOME.86 EQU *
7635:              366           IND16             ;Index regs 16 bit
7635:C2 10          1+           REP   #%00010000   ;X=0 ( 16 bit index )
7637:               2+           INDEX16
7637:               3+           MEND
7637:              367           MEM8              ;For soft switch toggles
7637:E2 20          1+           SEP   #%00100000   ;Set M=1 ( 8 bit memory )
7639:               2+           MEMORY8
7639:               3+           MEND
7639:9C 00 C0      368           STZ   CLR80COL
763C:9C 0C C0      369           STZ   CLR80VID      ;40 COL
763F:9C 51 C0      370           STZ   TXTSET        ;Get out of graphics
7642:9C 54 C0      371           STZ   TXTPAGE1
7645:A9 01         372           LDA   #1
7647:8D 29 C0      373           STA   NEWVIDEO      ;Make sure not in new video mode
764A:A0 00 04      374           LDY   #LINE1        ;The text screen
764D:A9 A0         375           LDA   #'            '  ;Space
764F:        764F  376 ROM.NXT.SP EQU   *             ;Write another space
764F:99 00 04      377           STA   LINE1,Y       ;The text screen
7652:88            378           DEY
7653:10 FA   764F  379           BPL   ROM.NXT.SP
7655:60            380           RTS
```

```
7656:                382 ****************************************************************
7656:                383 *
7656:                384 * All following subroutines are MEMORY8 & INDEX8
7656:                385 * (subname)88
7656:                386 *
7656:                387 ****************************************************************

7656:                389 ;.............................................................
7656:                390 ;
7656:                391 ; ROM.COUT88
7656:                392 ; Print char on last line of screen. 40 col only.
7656:                393 ; offset by CUR. (CUR <40 !)
7656:                394 ; The caller must make sure the 'curser' does not
7656:                395 ; end up outside the text window!
7656:                396 ;
7656:                397 ; If ST.FAIL is true then print staggered fail status on scrn
7656:                398 ; If PORT.ON is true then send fail status to port 1
7656:                399 ;
7656:                400 ; Enter/RET: DATA8
7656:                401 ;.............................................................
7656:                402 ROM.COUT88 DATA8
7656:E2 30             1+          SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
7658:                  2+          MEMORY8
7658:                  3+          INDEX8
7658:                  4+          MEND
7658:AE 27 03        403           LDX   CUR         ;Get curser
765B:9D D0 07        404           STA   LINE1+$3D0,X
765E:AE 1D 03        405           LDX   ST.FAIL     ;For self test fail only
7661:E0 00           406           CPX   #0
7663:F0 18    767D   407           BEQ   NOT.FAIL
7665:                408 ;Fail status so print on screen in different posn
7665:                409 ;and send to port 1
7665:AE 27 03        410           LDX   CUR         ;Restore curser
7668:9D F1 03        411           STA   LINE1-$0F,X  ;Top lhs of scrn
766B:9D 72 04        412           STA   LINE1+$81-$0F,X  ;Next line shift one to right
766E:9D F3 04        413           STA   LINE1+$102-$0F,X  ;Nxt line, one to right
7671:AE 29 03        414           LDX   PORT.ON     ;Check do we write to port1 also ?
7674:E0 00           415           CPX   #0
7676:F0 05    767D   416           BEQ   NOT.FAIL    ;No so don' write to port 1
7678:20 49 77        417           JSR   COUT.PORT1  ;Send Acc to port
767B:                418           NTV               ;Make sure we're still in native mode
767B:18                1+          CLC
767C:FB                2+          XCE                ;Get into native mode
767D:                  3+          mend
767D:         767D   419 NOT.FAIL  EQU   *
767D:AE 27 03        420           LDX   CUR         ;
7680:EE 27 03        421           INC   CUR         ;Increment curser
7683:60              422           RTS

7684:                424 ;.............................................................
7684:                425 ; ROM.PRBYTE88    ( Accumulator )
7684:                426 ; Print a byte on screen at CUR
7684:                427 ; Enter/Ret DATA8
7684:                428 ;.............................................................
7684:         7684   429 ROM.PRBYTE88 EQU *
7684:48              430           PHA               ;Save byte to prt
7685:6A              431           ROR   A
7686:6A              432           ROR   A
```

```
7687:6A              433          ROR   A
7688:6A              434          ROR   A           ;Get hi nybble into low nyb posn
7689:20 8D 76        435          JSR   PRNYBBLE     ;go prt nybble
768C:68              436          PLA               ;Now work on low nybble
768D:29 0F           437 PRNYBBLE AND   #$0F         ;Strip out hi nybble
768F:C9 0A           438          CMP   #$0A
7691:90 02    7695   439          BLT   PR.NOADD
7693:69 06           440          ADC   #$06
7695:         7695   441 PR.NOADD EQU   *
7695:69 B0           442          ADC   #$B0
7697:80 BD    7656   443          BRA   ROM.COUT88

7699:                445 ;.....................................................
7699:                446 ; PRINT.STATUS88
7699:                447 ; Print 6 byte status on text scrn.
7699:                448 ;
7699:                449 ; Enter/Ret DATA8
7699:                450 ;.....................................................
7699:         7699   451 PRINT.STATUS88 EQU *
7699:A9 0F           452          LDA   #15
769B:8D 27 03        453          STA   CUR           ;Set up curser
769E:A0 00           454          LDY   #0
76A0:5A              455          PHY               ;Save Y when using ROM.PRBYTE
76A1:         76A1   456 NXT.STAT  EQU   *
76A1:B9 15 03        457          LDA   TST.STATUS,Y  ;Get the byte
76A4:20 B4 76        458          JSR   ROM.PRBYTE88  ;Y gets trashed
76A7:7A              459          PLY               ;Restore byte count
76A8:C8              460          INY               ;Nxt byte
76A9:5A              461          PHY
76AA:C0 04           462          CPY   #4
76AC:D0 F3    76A1   463          BNE   NXT.STAT      ;Next status byte
76AE:7A              464          PLY
76AF:60              465          RTS
```

```
76B0:                467 ;
76B0:                468 ;.....................................................................
76B0:                469 ; RD.BI.NVR.88.66
76B0:                470 ; Read NVR counters to ram for BI cycling
76B0:                471 ;
76B0:                472 ; NTV DATA 8
76B0:                473 ;.....................................................................
76B0:          76B0  474 RD.BI.NVR.88.66 EQU *
76B0:          0361  475 MVN.NVR   EQU   NVR.FREE     ;Lower 2 bytes of BRAMBUFFER adr (free bat space)
76B0:                476           DATA8
76B0:E2 30            1+          SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
76B2:                 2+          MEMORY8
76B2:                 3+          INDEX8
76B2:                 4+          MEND
76B2:22 84 00 E1     477          JSL   =TOREADBR    ;Read NVR to bank ^BRAMBUFFER (=$E1)
76B6:                478          DATA16
76B6:C2 30            1+          REP   #%00110000  ;SET M=0, X=0 (16 BIT)
76B8:                 2+          MEMORY16
76B8:                 3+          INDEX16
76B8:                 4+          MEND
76B8:A2 61 03        479          LDX   #MVN.NVR     ;Source addess
76BB:A0 00 03        480          LDY   #BI.STATUS   ;Destination address
76BE:A9 15 00        481          LDA   #NVR.END-NVR.BEG  ;Number of bytes to move
76C1:54 00 E1        482          MVN   ^BRAMBUFFER,$000000
76C4:60              483          RTS                ;

76C5:                485 ;.....................................................................
76C5:                486 ; WR.BI.NVR.66.88
76C5:                487 ; Write BRAMBUFFER to NVR
76C5:                488 ;
76C5:                489 ; NTV DATA 8
76C5:                490 ;.....................................................................
76C5:          76C5  491 WR.BI.NVR.66.88 EQU *
76C5:A2 00 03        492          LDX   #BI.STATUS   ;From (Bank $00)
76C8:A0 61 03        493          LDY   #MVN.NVR     ;To   (Bank $E1)
76CB:A9 15 00        494          LDA   #NVR.END-NVR.BEG  ;Number of bytes to move
76CE:54 E1 00        495          MVN   $000000,^BRAMBUFFER  ;Move status bytes to bank $E1
76D1:                496          DATA8
76D1:E2 30            1+          SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
76D3:                 2+          MEMORY8
76D3:                 3+          INDEX8
76D3:                 4+          MEND
76D3:AD 61 C0        497          LDA   SELF         ;Are we in BI but running Self Diags?
76D6:30 04    76DC   498          BMI   BI.SLF.NVR   ;Yes so don't write to NVR
76D8:22 80 00 E1     499          JSL   =TOWRITEBR   ;Write stuff permanently to BRAM
76DC:                500 BI.SLF.NVR DATA8
76DC:E2 30            1+          SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
76DE:                 2+          MEMORY8
76DE:                 3+          INDEX8
76DE:                 4+          MEND
76DE:60              501          RTS                ;

76DF:                503 ;.....................................................................
76DF:                504 ; BANK.0   XX/XX
76DF:                505 ; Set Data bank register to zero
76DF:                506 ;
76DF:                507 ;.....................................................................
```

```
76DF:          76DF  509 BANK.0    EQU   *
76DF:F4 00 00        510           PEA   0
76E2:AB              511           PLB
76E3:AB              512           PLB               ;Set it
76E4:60              513           RTS
76E5:                514 ;.....................................................
76E5:                515 ; BANK.RUN XX/XX
76E5:                516 ; Set Data bank register to bank code is in RUNBANK
76E5:                517 ;
76E5:                518 ;.....................................................

76E5:          76E5  520 BANK.RUN  EQU   *
76E5:F4 FF FF        521           PEA   PEA.BANK
76E8:AB              522           PLB
76E9:AB              523           PLB               ;Set normally to bank $FF
76EA:60              524           RTS
76EB:                525 ;.....................................................
76EB:                526 ; TST.DEATH.MGR
76EB:                527 ; This routine replaces the system death manager during tests.
76EB:                528 ; SYSDMGRV vector points to here.
76EB:                529 ;
76EB:                530 ;.....................................................

76EB:          00FF  532           SEG   RUNBANK     ;Bank this code is in
76EB:          76EB  533 TST.DEATH.MGR EQU *
76EB:                534           NTV16
76EB:18               1+           CLC
76EC:FB               2+           XCE               ;SET E = 0 ( Native mode )
76ED:C2 30            3+           REP   #%00110000
76EF:                 4+           MEMORY16
76EF:                 5+           INDEX16
76EF:                 6+           MEND
76EF:A2 FF 01        535           LDX   #$01FF
76F2:9A              536           TXS               ;Reset stack pointer
76F3:20 DF 76        537           JSR   BANK.0      ;Make sure Data Bank is zero
76F6:A9 00 00        538           LDA   #0
76F9:5B              539           TCD               ;Direct register
76FA:1A              540           INC   A           ;Errcode 01
76FB:8F 18 03 00     541           STA   TST.STATUS+3  ;DD=01: Death manager was called during test
76FF:82 E7 FD 74E9   542           BRL   TST.FAIL    ;Print/save test that was running
7702:                 59           INCLUDE COUT.PORT1
```

```
08 COUT.PORT1      Setup & write to Port 1            23-JUL-87  13:17 PAGE 47

7702:                3 ;.............................................................
7702:                4 ; COUT.PORT1  8/8
7702:                5 ; Output Fail status registers to serial device
7702:                6 ; using Pascal 1.1 entry points.
7702:                7 ;
7702:                8 ; Two entry Points:
7702:                9 ; INIT.PORT1 -> Initialise port for writing
7702:               10 ; COUT.PORT1 -> Send 8 bit acc to port1
7702:               11 ;
7702:               12 :
7702:               13 ; 22 Jan 86:  KG  Created
7702:               14 ; 20 Mar 86:  KG  Port to Bank $FF ROM
7702:               15 ;.............................................................

7702:         00FF  17           SEG   RUNBANK      :Bank code resides in
7702:         7702  18 INIT.PORT1 EQU  *
7702:               19           MEMORY8
7702:               20           INDEX8
7702:A9 04          21           LDA   #$04
7704:8D 68 C0       22           STA   :STATEREG    ;Correct language card setup

7707:               24 ;
7707:               25 ; Set up Port 1 Parameters
7707:               26 :
7707:               27 ; Copy PORT.PARMS to BRAMBUFFER ( in bank $E1 )
7707:               28 ; Then a Pascal Init call will use these parms
7707:               29 ; instead of those in the Control Panel.
7707:               30 :

7707:A2 0C          32           LDX   #END.PORT.PARMS-PORT.PARMS  ;All of port 1 parameters
7709:         7709  33 STO.NXT.PARM EQU *           ;Store next parameter
7709:BF 52 77 FF    34           LDA   =PORT.PARMS,X  ;Get it from table
770D:9F C0 02 E1    35           STA   =BRAMBUFFER,X  ;Save it in BRAMBUFFER
7711:CA             36           DEX
7712:10 F5    7709  37           BPL   STO.NXT.PARM  ;Do next parameter

7714:               39 ; Release $C800 space
7714:A9 C1          40           LDA   #<SLOT.ADDR
7716:8D F8 07       41           STA   :$07F8       ;For interrupts
7719:AD FF CF       42           LDA   :CLRROM      :Disable C800

771C:               44 ; Setup Init pointer
771C:AD 0D C1       45           LDA   :INIT.OFF    ;Offset byte for Pascal Init
771F:85 10          46           STA   INIT.JMP     ;Points to actual init address
7721:A9 C1          47           LDA   #<INIT.OFF
7723:85 11          48           STA   INIT.JMP+1   ;Contains actual Init address

7725:               50 ; Setup write pointer
7725:AD 0F C1       51           LDA   :WRITE.OFF   ;Offset byte for Pascal Write
7728:85 12          52           STA   WRITE.JMP
772A:A9 C1          53           LDA   #<WRITE.OFF
772C:85 13          54           STA   WRITE.JMP+1  ;Contains actual Write address

772E:               56 ; Relocate section of Bank $FF code to bank zero
772E:A9 0E          57           LDA   #BANK.ZERO.END-BANK.ZERO.BEG  ;No of bytes to transfer
7730:AA             58           TAX                ;Use y as byte loop counter
7731:         7731  59 NXT.ZBYTE EQU   *
7731:BF 5E 77 FF    60           LDA   =BANK.ZERO.BEG,X  ;Load it from bank $ff rom
```

```
08 COUT.PORT1      Setup & write to Port 1            23-JUL-87  13:17 PAGE 48

7735:9D 00 20        61           STA   :SER.ZBANK,X  ;Save it in zero bank
7738:CA              62           DEX                 ;Move down a byte
7739:10 F6    7731   63           BPL   NXT.ZBYTE     ;Repeat
773B:A2 C1           64           LDX   #<SLOT.ADDR   ;X & X must have slot related values
773D:A0 10           65           LDY   #SLOT*16

773F:                67 ; Initialise Port 1. Use Pascal 1.1 protocol
773F:         773F   68 DO.INIT   EQU   *
773F:22 00 20 00     69           JSL   =SER.ZBANK    ;Go initialise device for writing

7743:A9 AA           71           LDA   #'*'
7745:20 49 77        72           JSR   COUT.PORT1    ;Tell tester failcode is sent next
7748:60              73           RTS                 ;To sequencer

7749:         7749   75 COUT.PORT1 EQU  *            ; send acc to port
7749:A2 C1           76           LDX   #<SLOT.ADDR
774B:A0 10           77           LDY   #SLOT*16
774D:         774D   78 DO.WRITE  EQU   *            ;Write Acc to slot
774D:22 07 20 00     79           JSL   =SER.ZBANK+ZERO.WRITE-ZERO.INIT  ;To bank zero routine
7751:60              80           RTS                 ;


7752:         7752   82 PORT.PARMS EQU  *            ;Port Parms (default values except baud)

7752:00              84           DFB   $00          ;This port is a: Printer
7753:00              85           DFB   $00          ;Line Length: Unlimited
7754:00              86           DFB   $00          ;Delete First LF after CR: No
7755:01              87           DFB   $01          ;Add LF after CR: Yes
7756:00              88           DFB   $00          ;Echo: No
7757:00              89           DFB   $00          ;Buffering: No
7758:07              90           DFB   $07          ;Baud: 1200  ** Non Default **
7759:06              91           DFB   $06          ;Data/Stop Bits: 8/1
775A:02              92           DFB   $02          ;Parity: None
775B:01              93           DFB   $01          ;DCD Handshake: Yes
775C:01              94           DFB   $01          ;DSR Handshake: Yes
775D:00              95           DFB   $00          ;XON/XOFF Handshake: No
775E:         775E   96 END.PORT.PARMS EQU *
```

```
775E:                99 ;
775E:               100 ; The next section of code is downloaded to bank zero RAM
775E:               101 ; and run from there. (Because we cannot call the Port
775E:               102 ; firmware with a JSL)
775E:               103 :

775E:         00FF  105           SEG   RUNBANK
775E:         775E  106 BANK.ZERO.BEG EQU *          ;Beginning of Bank zero image
775E:         0000  107           SEG   $00
775E:         775E  108 ZERO.INIT EQU   *            ;Initialise from bank zero
775E:20 04 20       109           JSR   SER.ZBANK+IJ-ZERO.INIT  ;Go through INIT.JMP vector in bank $00
7761:6B             110           RTL                 ;Return to bank sequencer RUNBANK
7762:6C 10 00       111 IJ        JMP   (INIT.JMP)    ;Jump to $00/C1XX space

7765:         7765  113 ZERO.WRITE EQU  *           ;Write Acc form bank zero
7765:20 0B 20       114           JSR   SER.ZBANK+WJ-ZERO.INIT  ;Go through WRITE.JMP vector
7768:6B             115           RTL                 ;Return to sequencer RUNBANK
7769:6C 12 00       116 WJ        JMP   (WRITE.JMP)   ;Jump to $00/C1XX space
776C:         776C  117 BANK.ZERO.END EQU *          ;End of Bank zero image
776C:         0000   60           DO    SEQ.DEBUG
 S                   61           INCLUDE ROM.SEQ.DEBUG
776C:               62           FIN
776C:               63           INCLUDE MOVIRAM
```

```
776C:                3 * file: moviram
776C:                4 *
776C:                5 * 16 bit version of popular //e Ramtest of the 80's
776C:                6 *
776C:                7 * Revision History:
776C:                8 *
776C:                9 * 11 sep 85    RCarr    New Today
776C:               10 * 16 sep 85    RCarr    Converted to use indirect long addressing
776C:               11 * 05 dec 85    RCarr    Added to DVT test, make seg unrelated
776C:               12 * XX Jan 86    KG       Added to internal diags...
776C:               13 * 20 mar 86    RCarr    Improve Error Reporting and code krunch 1
776C:               14 *
776C:               15 * Data 8 , Data bank =0
776C:         0000  16          SEG   $00
776C:         776C  17 MOVIRAM  equ   *
776C:2C 50 C0       18          BIT   TXTCLR
776F:2C 52 C0       19          BIT   MIXCLR
7772:2C 57 C0       20          BIT   HIRES
7775:20 6B 78       21          jsr   DoZpage00
7778:B0 52    77CC  22          bcs   RamError2
777A:               23          ind16
777A:C2 10           1+         REP   #%00010000   ;X=0 ( 16 bit index )
777C:                2+         INDEX16
777C:                3+         MEND
777C:               24          mem8
777C:E2 20           1+         SEP   #%00100000   ;Set M=1 ( 8 bit memory )
777E:                2+         MEMORY8
777E:                3+         MEND
777E:EE 34 C0       25          INC   CLOCKCTL
7781:A9 00          26          lda   #0
7783:20 CD 77       27          jsr   MoviRamTest
7786:B0 3A    77C2  28          bcs   MoviRamExit
7788:EE 34 C0       29          INC   CLOCKCTL
778B:A9 C1          30          LDA   #$C1
778D:8D 29 C0       31          STA   NEWVIDEO
7790:A9 01          32          lda   #01
7792:20 CD 77       33          jsr   MoviRamTest
7795:B0 2B    77C2  34          bcs   MoviRamExit
7797:20 C7 78       35          jsr   DoZpageInd
779A:B0 26    77C2  36          bcs   MoviRamExit
779C:EE 34 C0       37          INC   CLOCKCTL
779F:A9 01          38          LDA   #1
77A1:8D 29 C0       39          STA   NEWVIDEO
77A4:A9 E0          40          lda   #$E0
77A6:20 CD 77       41          jsr   MoviRamTest
77A9:B0 17    77C2  42          bcs   MoviRamExit
77AB:20 C7 78       43          jsr   DoZpageInd
77AE:B0 12    77C2  44          bcs   MoviRamExit
77B0:EE 34 C0       45          INC   CLOCKCTL
77B3:A9 C1          46          LDA   #$C1
77B5:8D 29 C0       47          STA   NEWVIDEO
77B8:A9 E1          48          lda   #$E1
77BA:20 CD 77       49          jsr   MoviRamTest
77BD:B0 03    77C2  50          bcs   MoviRamExit
77BF:20 C7 78       51          jsr   DoZpageInd
77C2:               52 MoviRamExit data8
77C2:E2 30           1+         SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
77C4:                2+         MEMORY8
```

```
77C4:                3+          INDEX8
77C4:                4+          MEND
77C4:B0 01    77C7   53          bcs   RamError
77C6:6B              54          RTL
77C7:         77C7   55 RamError  EQU   *
77C7:A5 F0           56          lda   RamBank
77C9:8D 16 03        57          sta   ErrCode1
77CC:6B              58 RamError2 RTL
77CD:                59          memory8
77CD:                60          index16
77CD:                61 *
77CD:         77CD   62 MoviRamTest equ *
77CD:85 F0           63          sta   RamBank
77CF:                64          ind16
77CF:C2 10            1+         REP   #%00010000   :X=0 ( 16 bit index )
77D1:                 2+         INDEX16
77D1:                 3+         MEND
77D1:                65          mem8
77D1:E2 20            1+         SEP   #%00100000   :Set M=1 ( 8 bit memory )
77D3:                 2+         MEMORY8
77D3:                 3+         MEND
77D3:A2 00 04        66          ldx   #$0400
77D6:A0 FF BF        67          ldy   #$BFFF
77D9:20 FF 77        68          jsr   DoMoviTest
77DC:B0 20    77FE   69          bcs   FoundMoviErr
77DE:AD 8B C0        70          lda   lcbank1
77E1:AD 8B C0        71          lda   lcbank1
77E4:A2 00 D0        72          ldx   #$d000
77E7:A0 FF DF        73          ldy   #$dfff
77EA:20 FF 77        74          jsr   DoMoviTest
77ED:B0 0F    77FE   75          bcs   FoundMoviErr
77EF:AD 83 C0        76          lda   lcbank2
77F2:AD 83 C0        77          lda   lcbank2
77F5:A2 00 D0        78          ldx   #$d000
77F8:A0 FF FF        79          ldy   #$ffff
77FB:20 FF 77        80          jsr   DoMoviTest
77FE:60              81 FoundMoviErr rts
77FF:                82 *
77FF:                83          memory8
77FF:                84          index16
77FF:         77FF   85 DoMoviTest equ  *
77FF:84 F2           86          sty   RamEndAddr
7801:86 F4           87          stx   RamBegAddr
7803:A2 00 00        88          ldx   #$0000
7806:86 04           89          stx   loc
7808:A5 F0           90          lda   RamBank
780A:85 06           91          sta   loc+2
780C:A9 00           92          lda   #0
780E:85 07           93 outer16   sta   thedata
7810:A2 02 00        94          ldx   #$0002
7813:86 09           95          stx   mod
7815:A4 F4           96 stepp16   ldy   RamBegAddr
7817:A5 07           97          lda   thedata
7819:49 FF           98          eor   #$ff
781B:85 0B           99          sta   flip
781D:97 04          100 stuff16   sta   [loc],y
781F:C8             101          iny
7820:C4 F2          102          cpy   RamEndAddr
```

```
7822:90 F9    781D  103           bcc   stuff16
7824:A6 09          104           ldx   mod
7826:A4 F4          105           ldy   RamBegAddr
7828:A5 07          106           lda   thedata
782A:CA             107 alt16     dex
782B:10 05    7832  108           bpl   skip16
782D:A2 02 00       109           ldx   #$0002
7830:97 04          110           sta   [loc],y
7832:C8             111 skip16    iny
7833:C4 F2          112           cpy   RamEndAddr
7835:90 F3    782A  113           bcc   alt16
7837:A5 09          114           lda   mod
7839:D0 08    7843  115           bne   test16
783B:A5 07          116           lda   thedata
783D:C9 FF          117           cmp   #$FF
783F:D0 02    7843  118           bne   test16
7841:18             119           clc
7842:60             120           rts
7843:               121 *
7843:A6 09          122 test16    ldx   mod
7845:A4 F4          123           ldy   RamBegAddr
7847:A5 0B          124 inv16     lda   flip
7849:CA             125           dex
784A:10 05    7851  126           bpl   jmp16
784C:A2 02 00       127           ldx   #$02
784F:A5 07          128           lda   thedata
7851:D7 04          129 jmp16     cmp   [loc],y
7853:D0 0F    7864  130           bne   error16
7855:C8             131           iny
7856:C4 F2          132           cpy   RamEndAddr
7858:90 ED    7847  133           bcc   inv16
785A:C6 09          134           dec   mod
785C:10 B7    7815  135           bpl   stepp16
785E:A5 07          136           lda   thedata
7860:49 FF          137           eor   #$ff
7862:B0 AA    780E  138           bra   outer16
7864:               139 *
7864:57 04          140 error16   eor   [loc],y
7866:8D 17 03       141           sta   ErrCode2
7869:38             142           sec
786A:60             143           rts
786B:               144 *
786B:               145 *
786B:9C 16 03       146 doZPage00 stz   ErrCode1
786E:9C 17 03       147           stz   ErrCode2
7871:               148           ind16
7871:C2 10            1+          REP   #%00010000   ;X=0 ( 16 bit index )
7873:                 2+          INDEX16
7873:                 3+          MEND
7873:A0 00 00       149           ldy   #0
7876:B9 00 00       150 ZRam1     lda   $0000,y
7879:48             151           pha
787A:A9 FE          152           lda   #$FE
787C:A2 00 00       153           ldx   #$0
787F:99 00 00       154 ZRam2     sta   $0000,y
7882:D9 00 00       155           cmp   $0000,y
7885:D0 25    78AC  156           bne   ZError
7887:38             157           sec
```

```
7888:2A           158          rol   a
7889:E8           159          inx
788A:E0 08 00     160          cpx   #$08
788D:D0 F0   787F 161          bne   ZRam2
788F:CA           162          dex
7890:A9 80        163          lda   #$80
7892:99 00 00     164 ZRam4    sta   $0000,y
7895:D9 00 00     165          cmp   $0000,y
7898:D0 12   78AC 166          bne   ZError
789A:4A           167          lsr   a
789B:CA           168          dex
789C:10 F4   7892 169          bpl   ZRam4
789E:68           170          pla
789F:99 00 00     171          sta   $0000,y
78A2:C8           172          iny
78A3:C0 00 04     173          cpy   #$400
78A6:D0 CE   7876 174          bne   ZRam1
78A8:             175          ind8
78A8:E2 10         1+          SEP   #%00010000   ;Set X=1 ( 8bit index )
78AA:              2+          INDEX8
78AA:              3+          MEND
78AA:18           176          clc
78AB:60           177          rts
78AC:             178 *
78AC:             179 ZError   ind8
78AC:E2 10         1+          SEP   #%00010000   ;Set X=1 ( 8bit index )
78AE:              2+          INDEX8
78AE:              3+          MEND
78AE:59 00 00     180          eor   $0000,y
78B1:8D 17 03     181          sta   ErrCode2
78B4:9C 16 03     182          stz   ErrCode1
78B7:38           183          sec
78B8:60           184          rts
78B9:             185 ZErrInd  ind8
78B9:E2 10         1+          SEP   #%00010000   ;Set X=1 ( 8bit index )
78BB:              2+          INDEX8
78BB:              3+          MEND
78BB:57 04        186          eor   [loc],y
78BD:8D 17 03     187          sta   ErrCode2
78C0:A5 F0        188          lda   RamBank
78C2:8D 16 03     189          sta   ErrCode1
78C5:38           190          sec
78C6:60           191          rts
78C7:             192 *
78C7:        78C7 193 DoZPageInd equ  *          ;tests zero pages 1, E0, E1
78C7:             194          ind16
78C7:C2 10         1+          REP   #%00010000   ;X=0 ( 16 bit index )
78C9:              2+          INDEX16
78C9:              3+          MEND
78C9:A0 00 00     195          ldy   #0
78CC:86 04        196          stx   loc
78CE:A5 F0        197          lda   RamBank
78D0:85 06        198          sta   loc+2
78D2:B7 04        199 ZPInd1   lda   [loc],y
78D4:48           200          pha
78D5:A9 FE        201          lda   #$fe
78D7:A2 00 00     202          ldx   #$0000
78DA:97 04        203 ZPInd2   sta   [loc],y
```

```
78DC:D7 04          204          cmp   [loc],y
78DE:D0 D9    78B9  205          bne   ZErrIND
78E0:38             206          sec
78E1:2A             207          rol   a
78E2:EB             208          inx
78E3:E0 08 00       209          cpx   #$0008
78E6:D0 F2    78DA  210          bne   ZPInd2
78E8:CA             211          dex
78E9:A9 80          212          lda   #$80
78EB:97 04          213 ZPInd4   sta   [loc],y
78ED:D7 04          214          cmp   [loc],y
78EF:D0 C8    78B9  215          bne   ZErrIND
78F1:4A             216          lsr   a
78F2:CA             217          dex
78F3:10 F6    78EB  218          bpl   ZPInd4
78F5:68             219          pla
78F6:97 04          220          sta   [loc],y
78F8:C8             221          iny
78F9:C0 00 04       222          cpy   #$0400
78FC:D0 D4    78D2  223          bne   ZPInd1
78FE:               224          ind8
78FE:E2 10            1+         SEP   #%00010000   ;Set X=1 ( 8bit index )
7900:                 2+         INDEX8
7900:                 3+         MEND
7900:18             225          clc
7901:60             226          rts
7902:                64          INCLUDE SER.TST
```

```
7902:                3 ****************************************************************
7902:                4 *
7902:                5 *    File   :  SER.TST
7902:                6 *
7902:                7 *
7902:                8 *   Serial tests for the SCC (8530)
7902:                9 *
7902:               10 *    14-Nov-85   :  JR    : New Today!
7902:               11 *    23-Jan-86   :  JR    : Modified to run in ROM
7902:               12 *
7902:               13 *    See Diagnostics ERS for error codes
7902:               14 *
7902:               15 *    Data8, Data Bank = 0
7902:               16 ****************************************************************

7902:               18 *
7902:               19 *   Calls to the three test routines
7902:               20 *

7902:        7902   22 SER.TST   EQU   *
7902:        0000   23           SEG   $00          ;Should really be $FF, but program was written for bnk 0
7902:20 0B 79       24           JSR   SCCRegRW
7905:B0 03   790A   25           BCS   SCCEnd
7907:        0000   26           DO    BAUD.SWITCH  ;If baud tests included
 S                  27           JSR   BaudIRQ
 S                  28           BCS   SCCEnd
7907:               29           FIN
7907:20 8A 79       30           JSR   IntLoop
790A:6B             31 SCCEnd    RTL
```

```
790B:                33 ********************************************************************
790B:                34 *
790B:                35 *  SCC register R/W test
790B:                36 *
790B:                37 ********************************************************************

790B:          790B  39 SCCRegRW  EQU   *

790B:78              41           SEI                ;Disable interupts
790C:9C 0C 08        42           STZ   DATA
790F:9C 16 03        43           STZ   ErrCode1
7912:20 3D 7A        44           JSR   ResetSCC     ;SCC reset & point to Chn A
7915:A9 FF           45           LDA   #$FF
7917:8D 0E 08        46           STA   SerMask      ;BIT mask,for WR15 only
791A:A9 02           47           LDA   #2
791C:8D 0D 08        48           STA   RegNo        ;Write register to test
791F:20 55 79        49           JSR   RWChA        ;Test Wr 2 for ChA only
7922:30 23    7947   50           BMI   FailRW
7924:A9 0C           51           LDA   #12
7926:8D 0D 08        52           STA   RegNo        ;Test Wr reg 12
7929:20 4E 79        53           JSR   TstRW
792C:30 19    7947   54           BMI   FailRW
792E:EE 0D 08        55           INC   RegNo        ;Test Wr reg 13
7931:20 4E 79        56           JSR   TstRW
7934:30 11    7947   57           BMI   FailRW
7936:A9 FA           58           LDA   #$FA
7938:8D 0E 08        59           STA   SerMask      ;Mask non R/W bits in Wr 15
793B:A9 0F           60           LDA   #15
793D:8D 0D 08        61           STA   RegNo        ;Test Wr reg 15
7940:20 4E 79        62           JSR   TstRW
7943:30 02    7947   63           BMI   FailRW
7945:1B              64           CLC
7946:60              65           RTS
7947:A9 01           66 FailRW    LDA   #01
7949:8D 16 03        67           STA   ErrCode1
794C:3B              68           SEC
794D:60              69           RTS

794E:A0 3B           71 TstRW     LDY   #ChBIndex    ;Test Channel A & B
7950:20 5F 79        72           JSR   RegRW
7953:30 09    795E   73           BMI   EndRWtst
7955:A0 3C           74 RWChA     LDY   #ChAIndex
7957:20 5F 79        75           JSR   RegRW
795A:30 02    795E   76           BMI   EndRWtst
795C:A9 00           77           LDA   #0
795E:60              78 EndRWtst  RTS

795F:AD 0D 08        80 RegRW     LDA   RegNo        ;Actual R/W test
7962:99 FD BF        81           STA   SCCCmd,Y     ;Point to the Wr reg
7965:AD 0C 08        82           LDA   DATA         ;Initially = 0
7968:2D 0E 08        83           AND   SerMask      ;Mask out non R/W bits
796B:8D 0F 08        84           STA   MaskData     ;Store for later comparison
796E:99 FD BF        85           STA   SCCCmd,Y     ;Write the register
7971:AD 0D 08        86           LDA   RegNo
7974:99 FD BF        87           STA   SCCCmd,Y
7977:B9 FD BF        88           LDA   SCCCmd,Y     ;Read the register
797A:CD 0F 08        89           CMP   MaskData     ;Compare
797D:D0 08    7987   90           BNE   BadData
```

```
797F:CE 0C 08        91          DEC   DATA
7982:D0 DB    795F   92          BNE   RegRW       ;Try $0-FF
7984:A9 00           93          LDA   #$00
7986:60              94          RTS
7987:A9 80           95 BadData  LDA   #$80
7989:60              96          RTS
```

```
798A:  0000   98          DO    BAUD.SWITCH  :Leave out next section
S              99 ****************************************************************
S             100 *
S             101 *        Test  for         interupt on zero count of the baud rate generator
S             102 *
S             103 ****************************************************************
S             104          Skp   2
S             105 BaudIRQ  EQU   *
S             106          SEI                :Disable interupts
S             107          STZ   CNT1
S             108          STZ   CNT2
S             109          STZ   CNT3
S             110          STZ   CNT4
S             111          STZ   SerIRQCnt
S             112          JSR   ResetSCC
S             113          Skp   1
S             114 *
S             115 *             Local        IRQ vector setup for Zero count interupt.
S             116 *
S             117          Skp   1
S             118          LDA   =INTMGRV+1   :Interupt manager vector
S             119          STA   SerIRQtemp1
S             120          LDA   =INTMGRV+2
S             121          STA   SerIRQtemp2
S             122          LDA   =INTMGRV+3
S             123          STA   SerIRQtemp3
S             124          LDA   #>ZerIRQHnd
S             125          STA   =INTMGRV+1
S             126          LDA   #<ZerIRQHnd
S             127          STA   =INTMGRV+2
S             128          LDA   #^ZerIRQHnd
S             129          STA   =INTMGRV+3
S             130          LDX   #$00
S             131 SetEnv   EQU   *
S             132          LDA   =TBL1,X      :Set up the SCC regs
S             133          STA   SCCCmd,Y
S             134          INX
S             135          LDA   =TBL1,X
S             136          STA   SCCCmd,Y
S             137          INX
S             138          CPX   #14
S             139          BNE   SetEnv
S             140          CLI                :Enable interupts
S             141          Skp   1
S             142 Ze1      EQU   *
S             143          LDA   SerIRQCnt    :Wait for 1st interupt
S             144          CMP   #1
S             145          BEQ   Ze2
S             146          DEC   CNT1
S             147          BNE   Ze1
S             148          DEC   CNT2
S             149          BNE   Ze1
S             150 TIMOUT   EQU   *
S             151          LDA   #02          :Timout fail on IRQ one
S             152          STA   ErrCode1
S             153          BRA   ErrRestore
S             154          Skp   1
S             155 Ze2      EQU   *
```

```
S              156           LDA   SerIRQCnt   ;Wait for 2nd interupt
S              157           CMP   #2
S              158           BEQ   CheckTime
S              159           DEC   CNT3
S              160           BNE   Ze2
S              161           DEC   CNT4
S              162           BNE   Ze2
S              163           LDA   #03         ;Timout fail on IRQ two
S              164           STA   ErrCode1
S              165           BRA   ErrRestore
S              166           Skp   1
S              167 CheckTime EQU   *
S              168           LDA   CNT2
S              169           CMP   CNT4          ;Upper byte compare
S              170           BNE   BadXtal
S              171           LDA   CNT1
S              172           CMP   #Zer1+Limit   ;Low byte 1st IRQ
S              173           BGE   BadXtal
S              174           CMP   #Zer1-Limit
S              175           BLT   BadXtal
S              176           LDA   CNT3          ;Low byte 2nd IRQ
S              177           CMP   #Zer2+Limit
S              178           BGE   BadXtal
S              179           CMP   #Zer2-Limit
S              180           BLT   BadXtal
S              181           CLC
S              182           BRA   IRQRestore
S              183           Skp   1
S              184 BadXtal   EQU   *
S              185           LDA   #04
S              186           STA   ErrCode1
S              187           BRA   ErrRestore
S              188           Skp   1
S              189 ErrRestore EQU  *
S              190           JSR   IRQRestore
S              191           SEC
S              192           RTS
S              193           Skp   1
S              194 *
S              195 *               Local       interupt handler for IRQ on zero count
S              196 *
S              197           Skp   1
S              198           SEG   RUNBANK      ;Tell assembler next entry in bank $ff
S              199 ZerIRQHnd EQU   *
S              200           SEG   $00
S              201           INC   SerIRQCnt
S              202           LDA   SerIRQCnt   ;1st or 2nd time here ?
S              203           CMP   #2
S              204           BEQ   SCNDIRQ
S              205           LDA   CNT1
S              206           STA   STCNT1
S              207           LDA   CNT2
S              208           STA   STCNT2
S              209           BRA   Rtn
S              210 SCNDIRQ   LDA   CNT3
S              211           STA   STCNT3
S              212           LDA   CNT4
S              213           STA   STCNT4
```

```
S                214 Rtn       LDA   #0
S                215           STA   SCCCmd,Y
S                216           LDA   #$10          ;Reset ext int sts
S                217           STA   SCCCmd,Y
S                218           RTI
S                219           Skp   1
S                220 *
S                221 *               Intrupt     manager vector restore
S                222 *
S                223           Skp   1
S                224 IRQRestore EQU  *
S                225           LDY   #ChAIndex
S                226           LDA   #9
S                227           STA   SCCCmd,Y
S                228           LDA   #$00
S                229           STA   SCCCmd,Y
S                230           LDA   SerIRQtemp1
S                231           STA   =INTMGRV+1
S                232           LDA   SerIRQTemp2
S                233           STA   =INTMGRV+2
S                234           LDA   SerIRQtemp3
S                235           STA   =INTMGRV+3
S                236           CLC
S                237           RTS
798A:            238           FIN
```

```
798A:                240 ****************************************************************
798A:                241 *
798A:                242 *   Internal loopback test
798A:                243 *
798A:                244 ****************************************************************

798A:          798A  246 IntLoop   EQU   *

798A:78              248           SEI
798B:20 3D 7A        249           JSR   ResetSCC    :Reset SCC & point to Ch A
798E:A0 3C           250           LDY   #ChAIndex   ;
7990:20 26 7A        251           JSR   SCCRegSet
7993:20 A9 79        252           JSR   TstCh
7996:D0 0C    79A4   253           BNE   IntLoopErr

7998:A0 3B           255           LDY   #ChBIndex
799A:20 26 7A        256           JSR   SCCRegSet
799D:20 A9 79        257           JSR   TstCh
79A0:D0 02    79A4   258           BNE   IntLoopErr
79A2:18              259           CLC
79A3:60              260           RTS

79A4:8D 16 03        262 IntLoopErr STA  ErrCode1
79A7:38              263           SEC
79A8:60              264           RTS

79A9:9C 0C 08        266 TstCh     STZ   DATA
79AC:9C 00 08        267 Chn1      STZ   CNT1
79AF:9C 01 08        268           STZ   CNT2
79B2:A9 00           269 TxClr     LDA   #0
79B4:99 FD BF        270           STA   SCCCmd,Y
79B7:B9 FD BF        271           LDA   SCCCmd,Y
79BA:29 04           272           AND   #$04        ;Tx Buffer empty status
79BC:D0 08    79C6   273           BNE   SendData
79BE:CE 00 08        274           DEC   CNT1
79C1:D0 EF    79B2   275           BNE   TxClr
79C3:A9 05           276           LDA   #05         :Tx Buffer empty failure
79C5:60              277           RTS

79C6:AD 0C 08        279 SendData  LDA   DATA
79C9:99 FF BF        280           STA   SCCData,Y
79CC:9C 00 08        281           STZ   CNT1
79CF:A9 01           282 AllSent   LDA   #1
79D1:99 FD BF        283           STA   SCCCmd,Y
79D4:B9 FD BF        284           LDA   SCCCmd,Y
79D7:29 01           285           AND   #$01        :All Sent status
79D9:D0 0D    79E8   286           BNE   ChkRead
79DB:CE 00 08        287           DEC   CNT1
79DE:D0 EF    79CF   288           BNE   AllSent
79E0:CE 01 08        289           DEC   CNT2
79E3:D0 EA    79CF   290           BNE   ALLSENT
79E5:A9 06           291           LDA   #06         :All Sent status fail
79E7:60              292           RTS
79E8:9C 00 08        293 ChkRead   STZ   CNT1
79EB:9C 01 08        294           STZ   CNT2
79EE:A9 00           295 ChkRd     LDA   #0
79F0:99 FD BF        296           STA   SCCCmd,Y
79F3:B9 FD BF        297           LDA   SCCCmd,Y
```

```
79F6:29 01          298          AND   #$01        :Rx character available
79F8:D0 08    7A02  299          BNE   ChkData
79FA:CE 00 08       300          DEC   CNT1
79FD:D0 EF    79EE  301          BNE   ChkRd
79FF:A9 07          302          LDA   #07         :Rx char available error
7A01:60             303          RTS
7A02:B9 FF BF       304 ChkData  LDA   SCCData,Y
7A05:CD 0C 08       305          CMP   DATA
7A08:F0 03    7A0D  306          BEQ   TxChk
7A0A:A9 08          307          LDA   #08         :Bad Data
7A0C:60             308          RTS

7A0D:A9 01          310 TxChk    LDA   #1
7A0F:99 FD BF       311          STA   SCCCmd,Y
7A12:B9 FD BF       312          LDA   SCCCmd,Y
7A15:29 70          313          AND   #$70        :Framing, Overrun & Parity errors
7A17:F0 03    7A1C  314          BEQ   AGAIN
7A19:A9 09          315          LDA   #09         :Frame, Overrun or Parirty error
7A1B:60             316          RTS

7A1C:CE 0C 08       318 AGAIN    DEC   DATA
7A1F:F0 02    7A23  319          BEQ   Retrn
7A21:B0 B9    79AC  320          BRA   Chn1

7A23:A9 00          322 Retrn    LDA   #0
7A25:60             323          RTS

7A26:A2 00          325 SCCRegSet LDX  #0
7A28:BF 4A 7A FF    326 SetEnv2  LDA   =TBL2,X     :Set up the SCC registers
7A2C:99 FD BF       327          STA   SCCCmd,Y    ;  for internal loopback.
7A2F:E8             328          INX
7A30:BF 4A 7A FF    329          LDA   =TBL2,X
7A34:99 FD BF       330          STA   SCCCmd,Y
7A37:E8             331          INX
7A38:E0 10          332          CPX   #16
7A3A:D0 EC    7A28  333          BNE   SetEnv2
7A3C:60             334          RTS

7A3D:               336 *
7A3D:               337 *   Reset the SCC
7A3D:               338 *

7A3D:A0 3C          340 ResetSCC LDY   #ChAIndex   :Channel A
7A3F:A9 09          341          LDA   #$9         ;Reg 9
7A41:99 FD BF       342          STA   SCCCmd,Y
7A44:A9 C0          343          LDA   #$C0        ;Force an SCC reset
7A46:99 FD BF       344          STA   SCCCmd,Y
7A49:60             345          RTS

7A4A:         00FF  347          SEG   RUNBANK     :Normally bank $FF
7A4A:         7A4A  348 TBL2     EQU   *
7A4A:09 00          349          DFB   9,$00       :Disable interupts
7A4C:04 4C          350          DFB   4,$4C       :X16 clk 2 Stp bits
7A4E:0B D0          351          DFB   11,$D0      :Xtal RTxC
7A50:0C 5E          352          DFB   12,$5E      :Low Byte Time const
7A52:0D 00          353          DFB   13,$00      :Hi Byte Time const
7A54:0E 13          354          DFB   14,$13      :Loopback BR enable
7A56:03 C1          355          DFB   3,$C1       :Rx 8bits
```

```
7A58:05 6A          356          DFB   5,$6A       ;Tx 8bits Tx enable RTS

7A5A:       0000    358          DO    BAUD.SWITCH  ;Only if baud test assembled
 S                  359 TBL1     DFB   9,$0A       ;Enable interupts no vectors
 S                  360          DFB   12,$FF      ;Low byte of time constant
 S                  361          DFB   13,$FF      ;Hi byte       "     "
 S                  362          DFB   15,$02      ;Int on zero count enable
 S                  363          DFB   0,$10       ;Reset ext status interupt
 S                  364          DFB   1,$01       ;Ext int enable
 S                  365          DFB   14,$03      ;BR gen enable & BR source
7A5A:               366          FIN
7A5A:                65          INCLUDE SOFT.SW
```

```
7A5A:                3 ************************************************************
7A5A:                4 *
7A5A:                5 * MAIN PROG TO test cortland softswitches
7A5A:                6 * 8 BIT ROUTINE
7A5A:                7 * Enter with Data Bank = $00
7A5A:                8 * Returns with RTL and in 8 bit EML mode.
7A5A:                9 *
7A5A:               10 * Test returns with carry set if fail, else carry clear
7A5A:               11 * kg 01 OCT 85: Created
7A5A:               12 * KG 03 JAN 86: Modified to run in Bank $FF rom
7A5A:               13 *
7A5A:               14 *
7A5A:               15 * Karl Grabe, Apple Computer Inc., Corcaig
7A5A:               16 ************************************************************
7A5A:               17 *
7A5A:               18 * Now test each softswitch by setting & checking if set,
7A5A:               19 * then clearing and checking if clear.
7A5A:               20 *
7A5A:               21 * 8 of the softswitches have equivalent bits in the STATEREG
7A5A:               22 * register and these are tested/set/reset and verified
7A5A:               23 * for correct value and function.
7A5A:               24 *
7A5A:               25 * If a fail occurs then:
7A5A:               26 *       1.   The carry flag is set
7A5A:               27 *       2.   A contains the failed STATEREG bit ( if any )
7A5A:               28 *       3.   TST.STATUS+2 +$C000 gives the read softswitch location
7A5A:               29 *            that the fail occurred on
7A5A:               30 *       4.   TST.STATUS+1 contains STATEREG bit in error
7A5A:               31 *
7A5A:               32 * NOTE: Call in NTV DATA8, Data Bank =$00
7A5A:               33 *
7A5A:               34 *
7A5A:               35 ************************************************************

7A5A:        7A5A   37 SOFT.SW   EQU   *
7A5A:        00FF   38           SEG   RUNBANK
7A5A:20 E5 76       39           JSR   BANK.RUN     ;Set data bank reg to bank code is running in
7A5D:        00FF   40 NOT.SW    EQU   $FF            ;Flags non standard soft switch
7A5D:A9 01          41           LDA   #$01           ;STATEREG bit mask
7A5F:A0 29          42           LDY   #SOFT.TAB.SIZE ;For all softswitches
7A61:BE CE 7A       43 NEXT.SW   LDX   SOFT.TAB+1,Y   ;Get ready to set switch
7A64:E0 FF          44           CPX   #NOT.SW        ;Test for skip or not
7A66:F0 4C   7AB4   45           BEQ   INC.SW.1       ;Skip this soft switch

7A68:               47 * Set switch ...
7A68:9F 00 C0 00    48           STA   =IOADR,X       ;Set it now
7A6C:BE CF 7A       49           LDX   SOFT.TAB+2,Y   ;Get ready to read the switch
7A6F:20 DF 76       50           JSR   BANK.0       ;Set data bank reg to zero
7A72:1E 00 C0       51           ASL   :IOADR,X       ;Put bit 7 into carry
7A75:90 49   7AC0   52           BCC   MEGA.SOFT.ER   ;Mega softswitch error

7A77:               54 * STATEREG ...
7A77:C9 00          55           CMP   #$00           ;More STATEREG bits to test ?
7A79:F0 20   7A9B   56           BEQ   CLR.SW         ;No ... skip STATEREG test
7A7B:89 0C          57           BIT   #%00001100     ;Testing bit 2 ?
7A7D:D0 19   7A98   58           BNE   STATE.INV      ;Yes ... so test inverted.
7A7F:20 DF 76       59           JSR   BANK.0       ;Set data bank reg to zero
7A82:1C 68 C0       60           TRB   :STATEREG      ;Test/reset using A mask
```

```
7A85:F0 39    7AC0   61          BEQ   MEGA.SOFT.ER    ;State reg does'nt match
7A87:1E 00 C0        62          ASL   :IOADR,X        ;Does STATEREG clr softsw?
7A8A:B0 34    7AC0   63          BCS   MEGA.SOFT.ER    ;No so err
7A8C:0C 68 C0        64          TSB   :STATEREG       ;Test/set using A mask
7A8F:D0 2F    7AC0   65          BNE   MEGA.SOFT.ER    ;STATEREG did not set, err
7A91:1E 00 C0        66          ASL   :IOADR,X        ;Does STATEREG set softssw ?
7A94:90 2A    7AC0   67          BCC   MEGA.SOFT.ER    ;No .. er
7A96:80 03    7A9B   68          BRA   CLR.SW          ;Jump over inverted bits test

7A98:         7A98   70 STATE.INV EQU  *             ;Not branched to on new production Mega //
7A98:0C 68 C0        71          TSB   :STATEREG       ;Test/reset using A mask

7A9B:                73 * Clear switch ...
7A9B:         7A9B   74 CLR.SW    EQU  *
7A9B:20 E5 76        75          JSR   BANK.RUN      ;Set data bank reg to bank code is running in
7A9E:BE CD 7A        76          LDX   SOFT.TAB,Y      ;Get ready to clear
7AA1:9F 00 C0 00     77          STA   =IOADR,X        ;Clear it
7AA5:BE CF 7A        78          LDX   SOFT.TAB+2,Y    ;Get ready to read softswitch
7AA8:20 DF 76        79          JSR   BANK.0        ;Set data bank reg to zero
7AAB:1E 00 C0        80          ASL   :IOADR,X        ;Put bit 7 into carry
7AAE:B0 10    7AC0   81          BCS   MEGA.SOFT.ER    ;Mega softswitch error
7AB0:20 E5 76        82          JSR   BANK.RUN      ;Set data bank reg to bank code is running in
7AB3:88              83 INC.SW    DEY                   ;Inc softsw cntr by 3
7AB4:88              84 INC.SW.1  DEY                   ;Inc softsw cntr by 2
7AB5:0A              85          ASL   A               ;Move to next STATEREG bit
7AB6:88              86          DEY
7AB7:10 A8    7A61   87          BPL   NEXT.SW         ;Do next soft switch

7AB9:A9 0C           89          LDA   #$0C
7ABB:8D 68 C0        90          STA   :STATEREG     ;Restore to 'normal' settings
7ABE:18              91          CLC                   ;Clear carry for test pass
7ABF:6B              92          RTL

7AC0:         7AC0   94 MEGA.SOFT.ER EQU *
7AC0:8D 16 03        95          STA   :TST.STATUS+1   ;State register Bit at fault (if any)
7AC3:8E 17 03        96          STX   :TST.STATUS+2   ;>of soft switch at fault
7AC6:A9 0C           97          LDA   #$0C
7AC8:8D 68 C0        98          STA   :STATEREG     ;Restore softswitches to normal
7ACB:38              99          SEC                   ;Set carry as test fails
7ACC:6B             100          RTL                   ;Determine faulty softwsitch from Y
```

```
7ACD:                102 ************************************************************
7ACD:                103 *
7ACD:                104 * Softswitches table:
7ACD:                105 * Format: Clear softswitch, Set softswitch, Read softswitch
7ACD:                106 * The last 8 softswitches have a STATEREG bit.
7ACD:                107 *
7ACD:                108 ************************************************************

7ACD:          7ACD  110 SOFT.TAB       EQU       *
7ACD:0A 0B 17        111               DFB   >SETINTC3ROM,>SETSLOTC3ROM,>RDC3ROM
7AD0:00 01 18        112               DFB   >CLR80COL,>SET80COL,>RD80COL
7AD3:50 51 1A        113               DFB   >TXTCLR,>TXTSET,>RDTEXT
7AD6:52 53 1B        114               DFB   >MIXCLR,>MIXSET,>RDMIX
7AD9:56 57 1D        115               DFB   >LORES,>HIRES,>RDHIRES
7ADC:0E 0F 1E        116               DFB   >CLRALTCHAR,>SETALTCHAR,>ALTCHARSET
7ADF:0C 0D 1F        117               DFB   >CLR80VID,>SET80VID,>RD80VID
7AE2:                118 *
7AE2:                119 * The following softswitches have an equivalent bit
7AE2:                120 * in the Mega // STATEREG register.
7AE2:                121 *
7AE2:08 09 16        122 STATE.7   DFB   >SETSTDZP,>SETALTZP,>RDALTZP   ;Bit 7
7AE5:54 55 1C        123 STATE.6   DFB   >TXTPAGE1,>TXTPAGE2,>RDPAGE2   ;Bit 6
7AE8:02 03 13        124 STATE.5   DFB   >RDMAINRAM,>RDCARDRAM,>RDRAMRD ;Bit 5
7AEB:04 05 14        125 STATE.4   DFB   >WRMAINRAM,>WRCARDRAM,>RDRAMWRT Bit 4
7AEE:8B 83 11        126 STATE.3   DFB   >LCBANK1,>LCBANK2,>RDLCBNK2    ;Bit 3
7AF1:81 8B 12        127 STATE.2   DFB   >ROMIN,>LCBANK1,>RDLCRAM       ;Bit 2
7AF4:FF 28           128 STATE.1   DFB   NOT.SW,>ROMBANK          ;Not soft switch. Bit 1 STATEREG
7AF6:06 07 15        129 STATE.0   DFB   >SETSLOTCXROM,>SETINTCXROM,>RDCXROM ;Bit 0 STATEREG
7AF9:          0029  130 SOFT.TAB.SIZE EQU *-SOFT.TAB-3
7AF9:          7AF9  131 SOFT.END  EQU   *
7AF9:          0000  132           SEG   $00
7AF9:                 66           INCLUDE FPI.SPEED
```

```
7AF9:                3 ;.....................................................
7AF9:                4 ; FPI.SPEED
7AF9:                5 ;
7AF9:                6 ;
7AF9:                7 ; MAIN PROG TO CHECK FAST/SLOW SPEED RATIO OF RAMBO
7AF9:                8 ;
7AF9:                9 ; K.G. 17 SEP 85
7AF9:               10 ;
7AF9:               11 ; 17 SEP 85  File Created
7AF9:               12 ; 14 FEB 86  Modified to run in Bank $FF rom
7AF9:               13 ;
7AF9:               14 ; Call in Native mode 8, Data Bank = $00
7AF9:               15 ; Returns with RTL in Native Mode, Data8, Data Bank = $00
7AF9:               16 ;
7AF9:               17 ;  Errors: TST.STATUS+1: 1=>stuck fast, 2=>stuck slow
7AF9:               18 ;          TST.STATUS+2: Bad ratio measured
7AF9:               19 ;
7AF9:               20 ; By Karl Grabe  Apple Computer Ltd. Cork
7AF9:               21 ;.....................................................

7AF9:        7AF9   23 FPI.SPEED EQU   *           Sequencer calls with JSL
7AF9:        00FF   24           SEG   RUNBANK     ;Bank code is running in
7AF9:        191A   25 FAST.LIM  EQU   $191A       ;Lower and upper fast speed limits, ER 1
7AF9:        0C0D   26 SLOW.LIM  EQU   $0C0D       ;Lwr and upr slow speed limits for ROM, ER 2
7AF9:        7AF9   27 LBL00     EQU   *           ;Check correct Video counter freq in Fast mode
7AF9:               28           FAST
7AF9:A9 80           1+          LDA   #%10000000
7AFB:8F 36 C0 00     2+          STA   CYAREG      ;Set fpi fast
7AFF:                3+          MEND
7AFF:20 26 7B        29          JSR   FIND.SPEED  ;Leave result in X
7B02:A0 01           30          LDY   #1          ;Section of the test if it fails =1
7B04:E0 19           31          CPX   #<FAST.LIM  ;Test low value first
7B06:F0 04   7B0C    32          BEQ   NOW.SLOW    ;Is good so check slow mode
7B08:E0 1A           33          CPX   #>FAST.LIM  ;Try high value
7B0A:D0 12   7B1E    34          BNE   BAD.SPEED   ;Bad so error

7B0C:        7B0C   36 NOW.SLOW  EQU   *           ;Check correct Video counter freq in Slow mode
7B0C:9C 36 C0        37          STZ   :CYAREG     ;slow mode
7B0F:20 26 7B        38          JSR   FIND.SPEED
7B12:A0 02           39          LDY   #2          ;Section of the test if it fails =2
7B14:E0 0C           40          CPX   #<SLOW.LIM  ;Test low value
7B16:F0 04   7B1C    41          BEQ   GOOD.SPEED
7B18:E0 0D           42          CPX   #>SLOW.LIM  ;Try high value
7B1A:D0 02   7B1E    43          BNE   BAD.SPEED

7B1C:        7B1C   45 GOOD.SPEED EQU  *           ;All is well
7B1C:18              46          CLC               ;No error status
7B1D:6B              47          RTL

7B1E:        7B1E   49 BAD.SPEED EQU   *           ;Failed speed test if here
7B1E:8C 16 03        50          STY   :TST.STATUS+1  ;Section of test that failed (1..2)
7B21:8E 17 03        51          STX   :TST.STATUS+2  ;Wrong value we found
7B24:38              52          SEC               ;Bad status
7B25:6B              53          RTL
```

```
7B26:                55 ****************************************************************
7B26:                56 *
7B26:                57 * Find speed of 816 relative to the Vertical video counter
7B26:                58 * The X register ends up with a no which is proportional
7B26:                59 * to the speed of the 816 with respect to the Vertical video
7B26:                60 * counter which is running at a constant rate.
7B26:                61 *
7B26:                62 * Call/Ret in NTV8 mode
7B26:                63 *
7B26:                64 * Return with speed relative speed in X register.
7B26:                65 *
7B26:                66 *
7B26:                67 * Note that this test also takes into account the
7B26:                68 * fact that the FPI should slow down when accessing
7B26:                69 * the Mega // video counters.
7B26:                70 *
7B26:                71 ****************************************************************

7B26:A2 00           73 FIND.SPEED LDX  #0          ;Zero counter.
7B28:AF 2E C0 00     74            LDA  VERTCNT     ;Get vertical video counter
7B2C:CF 2E C0 00     75 LBL01      CMP  VERTCNT     :Wait till it changes
7B30:F0 FA    7B2C   76            BEQ  LBL01       ;Loop till new line
7B32:AF 2E C0 00     77            LDA  VERTCNT     ;Just begun new value so get it
7B36:E8              78 LBL02      INX              ;X counts occurences of same VERTCNT
7B37:CF 2E C0 00     79            CMP  VERTCNT
7B3B:F0 F9    7B36   80            BEQ  LBL02       ;Repeate till it changes again
7B3D:60              81            RTS
7B3E:                67            INCLUDE CUSTOM.IRQ
```

```
7B3E:                 3 ****************************************************************
7B3E:                 4 *
7B3E:                 5 *    File : Custom.IRQ
7B3E:                 6 *
7B3E:                 7 ****************************************************************
7B3E:                 8 *
7B3E:                 9 *   MEGA and VGC interupts test.
7B3E:                10 *
7B3E:                11 *  Modification history:
7B3E:                12 *
7B3E:                13 *  12th-Nov-85   :  JR   : Created
7B3E:                14 *  17th-Feb-86   :  JR   : Modified for Rom
7B3E:                15 *
7B3E:                16 * Sequencer calls in NTV8 mode, Data Bank = $00 with JSL.
7B3E:                17 *
7B3E:                18 ****************************************************************

7B3E:          0000  20           SEG   $00          ;Code was written for bank $00
7B3E:          7000  21 CusIRQCnt EQU   $7000
7B3E:          7002  22 CusIRQsts EQU   $7002
7B3E:          7004  23 CusLineNo EQU   $7004
7B3E:          7006  24 CusIRQSrce EQU  $7006
7B3E:          7008  25 IRQTemp1  EQU   $7008
7B3E:          700A  26 IRQTemp2  EQU   $700A
7B3E:          700C  27 IRQTemp3  EQU   $700C

7B3E:          7B3E  29 CUSTOM.IRQ EQU  *           ;Sequencer entry point
7B3E:20 47 7B        30           JSR   MEGATstIRQ   ;Test Mega // interupts
7B41:B0 03    7B46   31           BCS   CUS.IRQ.RTL  ;Found error so RTL
7B43:20 FE 7B        32           JSR   VGCTstIRQ    ;Now VGC
7B46:6B              33 CUS.IRQ.RTL RTL

7B47:                35 *
7B47:                36 *   VBL interupt
7B47:                37 *

7B47:          7B47  39 MEGATstIRQ EQU  *
7B47:78              40           SEI
7B48:AF 11 00 E1     41           LDA   =INTMGRV+1
7B4C:8D 0C 70        42           STA   IRQTEMP3
7B4F:AF 12 00 E1     43           LDA   =INTMGRV+2
7B53:8D 08 70        44           STA   IRQTemp1
7B56:AF 13 00 E1     45           LDA   =INTMGRV+3
7B5A:8D 0A 70        46           STA   IRQTemp2
7B5D:A9 F1           47           LDA   #>MEGAIntHnd
7B5F:8F 11 00 E1     48           STA   =INTMGRV+1
7B63:A9 7B           49           LDA   #<MEGAIntHnd
7B65:8F 12 00 E1     50           STA   =INTMGRV+2
7B69:A9 FF           51           LDA   #^MEGAIntHnd
7B6B:8F 13 00 E1     52           STA   =INTMGRV+3

7B6F:9C 41 C0        54           STZ   INTEN
7B72:9C 00 08        55           STZ   CNT1
7B75:9C 01 08        56           STZ   CNT2
7B78:9C 00 70        57           STZ   CusIRQCnt
7B7B:58              58           CLI

7B7C:9C 47 C0        60           STZ   CLRVBLINT
```

```
7B7F:A9 08          61          LDA  #%00001000
7B81:8D 41 C0       62          STA  INTEN
7B84:AD 00 70       63 Loop001  LDA  CusIRQCnt
7B87:C9 01          64          CMP  #1
7B89:F0 0E    7B99  65          BEQ  VBLIRQTst
7B8B:CE 00 08       66          DEC  CNT1
7B8E:D0 F4    7B84  67          BNE  Loop001
7B90:CE 01 08       68          DEC  CNT2
7B93:D0 EF    7B84  69          BNE  Loop001
7B95:A9 01          70          LDA  #$01
7B97:80 37    7BD0  71          BRA  MegaErr

7B99:AD 02 70       73 VBLIRQTst LDA  CusIRQsts
7B9C:29 08          74          AND  #%00001000
7B9E:D0 04    7BA4  75          BNE  QtrIRQ
7BA0:A9 02          76          LDA  #$02
7BA2:80 2C    7BD0  77          BRA  MegaErr

7BA4:               79 *
7BA4:               80 *  1/4 secound interupt
7BA4:               81 *

7BA4:9C 00 08       83 QtrIRQ   STZ  CNT1
7BA7:9C 01 08       84          STZ  CNT2
7BAA:9C 47 C0       85          STZ  CLRVBLINT
7BAD:A9 10          86          LDA  #%00010000
7BAF:8D 41 C0       87          STA  INTEN
7BB2:AD 00 70       88 Loop002  LDA  CusIRQCNT
7BB5:C9 02          89          CMP  #2
7BB7:F0 0E    7BC7  90          BEQ  TstQtrIRQ
7BB9:CE 00 08       91          DEC  CNT1
7BBC:D0 F4    7BB2  92          BNE  Loop002
7BBE:CE 01 08       93          DEC  CNT2
7BC1:D0 EF    7BB2  94          BNE  Loop002
7BC3:A9 03          95          LDA  #$03
7BC5:80 09    7BD0  96          BRA  MegaErr

7BC7:AD 02 70       98 TstQtrIRQ LDA  CusIRQsts
7BCA:29 10          99          AND  #%00010000
7BCC:D0 05    7BD3 100          BNE  MegRstre
7BCE:A9 04         101          LDA  #$04

7BD0:8D 16 03      103 MegaErr  STA  ErrCode1

7BD3:AD 0C 70      105 MegRstre LDA  IRQTEMP3
7BD6:8F 11 00 E1   106          STA  =INTMGRV+1
7BDA:AD 08 70      107          LDA  IRQTemp1
7BDD:8F 12 00 E1   108          STA  =INTMGRV+2
7BE1:AD 0A 70      109          LDA  IRQTemp2
7BE4:8F 13 00 E1   110          STA  =INTMGRV+3
7BE8:AD 16 03      111          LDA  ErrCode1
7BEB:F0 02    7BEF 112          BEQ  MegaPass
7BED:38            113          SEC
7BEE:60            114          RTS
7BEF:18            115 MegaPass CLC
7BF0:60            116          RTS

7BF1:         00FF 118          SEG  RUNBANK      ;Rom code bank
```

```
7BF1:          7BF1  119 MEGAIntHnd EQU  *
7BF1:9C 41 C0        120           STZ   :INTEN
7BF4:AD 46 C0        121           LDA   :INTFLAG
7BF7:8D 02 70        122           STA   :CusIRQsts
7BFA:EE 00 70        123           INC   :CusIRQCnt
7BFD:40              124           RTI
7BFE:          0000  125           SEG   $00
```

```
7BFE:                128 ********************************************************************
7BFE:                129 *
7BFE:                130 *   This is a test of the VGC 1 sec interrupt , and of each of
7BFE:                131 *   the possable scan line interupts .
7BFE:                132 *
7BFE:                133 *     13.Nov.85
7BFE:                134 *
7BFE:                135 ********************************************************************

7BFE:          7BFE  137 VGCTstIRQ EQU   *
7BFE:                138           DATA8
7BFE:E2 30            1+           SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
7C00:                 2+           MEMORY8
7C00:                 3+           INDEX8
7C00:                 4+           MEND
7C00:78              139           SEI
7C01:AF 11 00 E1     140           LDA   =INTMGRV+1
7C05:8D 0C 70        141           STA   IRQTEMP3
7C08:AF 12 00 E1     142           LDA   =INTMGRV+2
7C0C:8D 08 70        143           STA   IRQTemp1
7C0F:AF 13 00 E1     144           LDA   =INTMGRV+3
7C13:8D 0A 70        145           STA   IRQTemp2
7C16:A9 8F           146           LDA   #>VGCIntHnd
7C18:8F 11 00 E1     147           STA   =INTMGRV+1
7C1C:A9 7C           148           LDA   #<VGCIntHnd
7C1E:8F 12 00 E1     149           STA   =INTMGRV+2
7C22:A9 FF           150           LDA   #^VGCIntHnd
7C24:8F 13 00 E1     151           STA   =INTMGRV+3

7C28:9C 23 C0        153           STZ   VGCINT      ;VGC Interupts off
7C2B:9C 32 C0        154           STZ   SCANINT     ;Clear IRQ flags
7C2E:9C 06 70        155           STZ   CusIRQSrce  ;Point to 1 sec
7C31:9C 00 70        156           STZ   CusIRQCnt
7C34:9C 01 08        157           STZ   CNT2
7C37:A9 10           158           LDA   #$10         :About  4 seconds
7C39:8D 02 08        159           STA   CNT3        ;Timeout limit for 1 sec IRQ

7C3C:58              161           CLI
7C3D:A9 04           162           LDA   #%00000100
7C3F:8D 23 C0        163           STA   VGCINT
7C42:AD 00 70        164 Loop003   LDA   CusIRQCNT
7C45:C9 01           165           CMP   #1
7C47:F0 13    7C5C   166           BEQ   TstSecIRQ
7C49:CE 00 08        167           DEC   CNT1
7C4C:D0 F4    7C42   168           BNE   Loop003
7C4E:CE 01 08        169           DEC   CNT2
7C51:D0 EF    7C42   170           BNE   Loop003
7C53:CE 02 08        171           DEC   CNT3
7C56:D0 EA    7C42   172           BNE   Loop003
7C58:A9 05           173           LDA   #$05
7C5A:80 09    7C65   174           BRA   ERROR2
```

```
13 CUSTOM.IRQ      VGC IRQ Test                        23-JUL-87  13:17 PAGE 73

7C5C:AD 02 70       176 TstSecIRQ LDA   CusIRQsts   ;Test the IRQ flag
7C5F:C9 D4          177           CMP   #$D4        ;For New VGC's  - 31 MAR 86
7C61:         0000  178           DO    VGC.SWITCH  ;For VGC's after about 31 Mar
 S                  179           BEQ   ScanTst     ;Scan Line interrupt
7C61:               180           ELSE
7C61:F0 05    7C68  181           BEQ   VGCRstre    ;( BEQ ScanTst for production VGC's)
7C63:               182           FIN
7C63:A9 06          183           LDA   #$06
7C65:8D 16 03       184 ERROR2    STA   ErrCode1

7C68:78             186 VGCRstre  SEI
7C69:9C 23 C0       187           STZ   VGCINT
7C6C:A9 01          188           LDA   #$01
7C6E:8D 29 C0       189           STA   NEWVIDEO
7C71:AD 0C 70       190           LDA   IRQTEMP3
7C74:8F 11 00 E1    191           STA   =INTMGRV+1
7C78:AD 0B 70       192           LDA   IRQTemp1
7C7B:8F 12 00 E1    193           STA   =INTMGRV+2
7C7F:AD 0A 70       194           LDA   IRQTemp2
7C82:8F 13 00 E1    195           STA   =INTMGRV+3
7C86:AD 16 03       196           LDA   ErrCode1
7C89:F0 02    7C8D  197           BEQ   VGCPass
7C8B:38             198           SEC
7C8C:60             199           RTS
7C8D:18             200 VGCPass   CLC
7C8E:60             201           RTS

7C8F:               203 *
7C8F:               204 *  Scan line test
7C8F:               205 *

7C8F:         0000  207           DO    VGC.SWITCH  ;For new VGC's/room in rom
 S                  208 ScanTst   SEI
 S                  209           LDA   #$FF
 S                  210           STA   CusIRQCnt
 S                  211           STA   CusIRQSrce
 S                  212           STZ   CusLineNo
 S                  213           STZ   VGCINT
 S                  214           STZ   SCANINT
 S                  215           STZ   CNT1
 S                  216           STZ   CNT2
 S                  217           LDA   #$C1
 S                  218           STA   NEWVIDEO    ;Enable hires
 S                  219 ScnLine   LDX   CusLineNo   ;One line at a time
 S                  220 SetIRQ    LDA   #%01000000
 S                  221           ORA   $019D00,X
 S                  222           STA   $019D00,X   ;Set the IRQ bit
 S                  223           CLI
 S                  224           LDA   #%00000010
 S                  225           STA   VGCINT
 S                  226 WaitFIN   LDA   CusIRQCnt
 S                  227           CMP   CusLineNo
 S                  228           BEQ   ScnIRQTst
 S                  229           DEC   CNT1
 S                  230           BNE   WaitFIN
 S                  231           DEC   CNT2
 S                  232           BNE   WaitFIN
 S                  233           LDA   #$07
```

```
 S                234          BRA   ERROR2
 S                235          SKP   2
 S                236 ScnIRQTst LDA   CusIRQsts
 S                237          CMP   #$B2
 S                238          BNE   FailSts
 S                239          INC   CusLineNo
 S                240          LDA   CusLineNo
 S                241          CMP   #0          ;Wrap a round
 S                242          BNE   ScnLine
 S                243          BRA   VGCRstre
 S                244 FailSts   LDA   #$08
 S                245          BRL   ERROR2
7C8F:             246          FIN

7C8F:        00FF 248          SEG   RUNBANK
7C8F:        7C8F 249 VGCIntHnd EQU   *           ;VGC interupt handler
7C8F:        0000 250          SEG   $00
7C8F:AD 23 C0     251          LDA   VGCINT
7C92:8D 02 70     252          STA   CusIRQsts
7C95:AD 06 70     253          LDA   CusIRQSrce
7C98:F0 0D   7CA7 254          BEQ   IncCnt
7C9A:AE 04 70     255          LDX   CusLineNo
7C9D:A9 BF        256          LDA   #%10111111
7C9F:3F 00 9D 01  257          AND   $019D00,X
7CA3:9F 00 9D 01  258          STA   $019D00,X
7CA7:9C 23 C0     259 IncCnt   STZ   VGCINT
7CAA:EE 00 70     260          INC   CusIRQCnt
7CAD:40           261          RTI
7CAE:              68          INCLUDE ROM.CHECKSUM
```

```
7CAE:                3 ;.....................................................................
7CAE:                4 ; ROM.CHECKSUM
7CAE:                5 ;
7CAE:                6 ; Compute Checksum of rom banks $FE, $FF.
7CAE:                7 ; and compare with expected value
7CAE:                8 ;
7CAE:                9 ; Call  :   In Full native mode, Data Bank 0, use JSL
7CAE:               10 ; Return:   In Full native mode
7CAE:               11 ;
7CAE:               12 ; 20 Feb 86: Created
7CAE:               13 ; 09 Apr 86: Add register based ram test
7CAE:               14 ;
7CAE:               15 ; By Karl Grabe, Apple Computer INC., Cork
7CAE:               16 ;.....................................................................

7CAE:         00FF  18          SEG   RUNBANK
7CAE:         7CAE  19 ROM.CHECKSUM EQU *          :Called from  test sequencer
7CAE:20 6B 78       20          JSR   DoZpage00    ;Register based ram test first
7CB1:               21          DATA16
7CB1:C2 30           1+         REP   #%00110000  ;SET M=0, X=0 (16 BIT)
7CB3:                2+         MEMORY16
7CB3:                3+         INDEX16
7CB3:                4+         MEND
7CB3:B0 2A    7CDF  22          BCS   BAD.RAM      ;Ram is bad so report and stop
7CB5:64 FD          23          STZ   CHK.BEG      :Starting address
7CB7:F4 FE FE       24          PEA   :$FEFE       ;Firts bank $FE
7CBA:AB             25          PLB
7CBB:AB             26          PLB                ;Set data bank register
7CBC:A0 FF FF       27          LDY   #$FFFF       :Entire 64k of rom
7CBF:20 EA 7C       28          JSR   CAL.CHECKSUM  ;Go get checksum
7CC2:20 E5 76       29          JSR   BANK.RUN     ;Set data bank to bank code is in
7CC5:A0 FF FF       30          LDY   #$FFFF       ;Entire 64k of rom
7CC8:20 EE 7C       31          JSR   CAL.CHECKSUM2  :Get checksum
7CCB:               32 ;
7CCB:               33 ; Note that the rom checksum must be adjusted
7CCB:               34 ; to equal $1234
7CCB:               35 ;
7CCB:C9 34 12       36          CMP   #$1234       ;Compare with a fixed value
7CCE:               37 ; Change CHECKSUM condition in file: MAIN
7CCE:         0000  38          DO    CHECKSUM     ;Normally do next line
 S                  39          BNE   BAD.ROM      ;Bad checksum so fail
7CCE:               40          ELSE              ;Do next line for debugging only
7CCE:80 00    7CD0  41          BRA   GOOD.ROM     ;Don't fail ROM checkum
7CD0:               42          FIN
7CD0:         7CD0  43 GOOD.ROM  EQU   *           ;Rom ok
7CD0:18             44          CLC
7CD1:6B             45          RTL
7CD2:         7CD2  46 BAD.ROM   EQU   *           ;Rom does'nt check out
7CD2:8F 16 03 00    47          STA   TST.STATUS+1  :Save actual checksum for display
7CD6:A9 D2 CD       48          LDA   #$CDD2       ;"RM" = RoM fail
7CD9:8F 00 05 00    49          STA   LINE1+$100   ;Put "RM" msg on screen in case system hangs
7CDD:38             50          SEC
7CDE:6B             51          RTL                ;Attempt to return to sequencer to put bad message on screen
7CDF:         7CDF  52 BAD.RAM   EQU   *           ;Encountered bad ram during rom test
7CDF:A9 01 00       53          LDA   #$01         ;Bad ram
7CE2:8D 18 03       54          STA   :TST.STATUS+3  :01-> can't do rom tst because bad ram
7CE5:8D 16 03       55          STA   :TST.STATUS+1  ;Error that occurred (See Test code)
7CE8:38             56          SEC
```

7CE9:6B 57 RTL

```
7CEA:                59 ;.....................................................................
7CEA:                60 ;
7CEA:                61 ; Calculate Checksum. 16 Bit ADC
7CEA:                62 ;
7CEA:                63 ; Call  :   In Full native mode, use JSR
7CEA:                64 ; Return:   In Full native mode
7CEA:                65 ;
7CEA:                66 ; Inputs:
7CEA:                67 ;       C: Partially computed checksum, or else 0
7CEA:                68 ;  CHK.BEG: Start address to checksum (Zero Page var)
7CEA:                69 ;       Y: No Bytes  (odd) to checksum
7CEA:                70 ;       B: Set data bank reg to required bank.
7CEA:                71 ;
7CEA:                72 ; Output:
7CEA:                73 ;       C: Checksum of C to C+Y inclusive
7CEA:                74 ;
7CEA:                75 ;
7CEA:                76 ; Note:     Have partially computed checksum in C when
7CEA:                77 ;           dealing with more than one bank.
7CEA:                78 ;
7CEA:                79 ; kg nov85
7CEA:                80 ;.....................................................................
7CEA:          7CEA  81 CAL.CHECKSUM EQU *          ;Enter here first bank with C=0, else enter next line+1
7CEA:A9 00 00        82           LDA   #$0000       ;Clear checksum to zero
7CED:18              83           CLC
7CEE:          7CEE  84 CAL.CHECKSUM2 EQU *         ;Enter here for remaining banks.
7CEE:88              85           DEY                ;Drop to low byte of top address
7CEF:          7CEF  86 N.NXT.CHK EQU   *
7CEF:71 FD           87           ADC   (CHK.BEG),Y  ;Add to previous value
7CF1:88              88           DEY                 ;Move to next byte
7CF2:F0 03     7CF7  89           BEQ   CHK.END      ;Fin except for last 2
7CF4:88              90           DEY                 ;Move to next byte
7CF5:D0 F8     7CEF  91           BNE   N.NXT.CHK    ;do next two bytes
7CF7:          7CF7  92 CHK.END   EQU   *
7CF7:71 FD           93           ADC   (CHK.BEG),Y  ;Add last 2 bytes
7CF9:60              94           RTS                ;Return in Native 16 mode
7CFA:                69           INCLUDE CLOCK
```

```
7CFA:                3 ;.....................................................................
7CFA:                4 ; CLOCK
7CFA:                5 ; Test each bit in the clock register for R/W
7CFA:                6 ; Restore correct time after test.
7CFA:                7 ;
7CFA:                8 ;
7CFA:                9 ; Created :5 Mar 86, KG
7CFA:               10 ;
7CFA:               11 ; By Karl Grabe, Corcaig.  Apple Computer Ltd.
7CFA:               12 ;.....................................................................

7CFA:         00FF  14           SEG   RUNBANK
7CFA:         7CFA  15 CLOCK     EQU   *
7CFA:F4 E1 E1       16           PEA   :$E1E1
7CFD:AB             17           PLB
7CFE:AB             18           PLB                 ;Bank $E1
7CFF:               19 ;
7CFF:               20 ; Save time
7CFF:               21 ;
7CFF:22 8C 00 E1    22           JSL   =TOREADTIME   ;Cad e an tam?
7D03:               23           DATA16
7D03:C2 30           1+          REP   #%00110000   ;SET M=0, X=0 (16 BIT)
7D05:                2+          MEMORY16
7D05:                3+          INDEX16
7D05:                4+          MEND
7D05:AD E1 03       24           LDA   :CLKRDATA     ;Get low binary time word
7D08:8D 00 09       25           STA   :BRAM.WORK    ;Save it here for retrieval after test
7D0B:AD E3 03       26           LDA   :CLKRDATA+2   ;Get hi time word
7D0E:8D 02 09       27           STA   :BRAM.WORK+2  ;Save it for after test
7D11:               28 ;
7D11:               29 ; Now setup CLKWDATA with 00000000 00000001
7D11:               30 ; and shift thro' each bit
7D11:               31 :
7D11:A9 01 00       32           LDA   #$0001        ;LSB
7D14:8D E5 03       33           STA   :CLKWDATA     ;Lo word
7D17:9C E7 03       34           STZ   :CLKWDATA+2   ;Hi word
7D1A:         7D1A  35 NXT.CLK.BIT EQU *             ;Check next clock bit
7D1A:20 4C 7D       36           JSR   :WR.RD.CMP.TIME  ;Write and see if correctly written
7D1D:F0 05    7D24  37           BEQ   CLK.CONT      ;Yes, so now check hi word
7D1F:               38 ;Doesn't match, so repeat as a second may have incremented ...
7D1F:20 4C 7D       39           JSR   :WR.RD.CMP.TIME  ;Write and see if correctly written
7D22:D0 26    7D4A  40           BNE   CLK.ERR       ;Still doesn't match so we have an error
7D24:AF E3 03 E1    41 CLK.CONT  LDA   CLKRDATA+2    ;Now check hi word
7D28:CD E7 03       42           CMP   :CLKWDATA+2
7D2B:D0 1D    7D4A  43           BNE   CLK.ERR       ;No so there's an error
7D2D:18             44           CLC
7D2E:2E E5 03       45           ROL   :CLKWDATA     ;If carry set then bit moves into hi word
7D31:2E E7 03       46           ROL   :CLKWDATA+2   ;
7D34:90 E4    7D1A  47           BCC   NXT.CLK.BIT   ;Not finished with 32 bits so continue
7D36:AD 00 09       48           LDA   :BRAM.WORK    ;Now restore original time
7D39:8D E5 03       49           STA   :CLKWDATA
7D3C:AD 02 09       50           LDA   :BRAM.WORK+2
7D3F:8D E7 03       51           STA   :CLKWDATA+2   ;Restore hi word
7D42:               52           DATA8
7D42:E2 30           1+          SEP   #%00110000   ;SET M=1, X=1 (8 BIT )
7D44:                2+          MEMORY8
7D44:                3+          INDEX8
7D44:                4+          MEND
```

```
7D44:22 88 00 E1     53          JSL   =TOWRITETIME
7D48:18              54          CLC
7D49:6B              55          RTL
7D4A:          7D4A  56 CLK.ERR   EQU   *           ;Fail handler
7D4A:38              57          SEC
7D4B:6B              58          RTL
```

```
7D4C:                60 ;.....................................................
7D4C:                61 ; WR.RD.CMP.TIME  16/16
7D4C:                62 ;
7D4C:                63 ; Write, read back and compare write to read data
7D4C:                64 ; Z=1 if results equal ( ie good )
7D4C:                65 ;.....................................................

7D4C:         7D4C   67 WR.RD.CMP.TIME EQU *
7D4C:                68           DATA8
7D4C:E2 30            1+          SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
7D4E:                 2+          MEMORY8
7D4E:                 3+          INDEX8
7D4E:                 4+          MEND
7D4E:22 88 00 E1      69          JSL   =TOWRITETIME  ;Write to clock
7D52:22 8C 00 E1      70          JSL   =TOREADTIME  ;Now read back
7D56:                 71          DATA16
7D56:C2 30            1+          REP   #%00110000  ;SET M=0, X=0 (16 BIT)
7D58:                 2+          MEMORY16
7D58:                 3+          INDEX16
7D58:                 4+          MEND
7D58:AD E1 03         72          LDA   :CLKRDATA    ;Clock data read back
7D5B:CD E5 03         73          CMP   :CLKWDATA    :Does it match what was written ?
7D5E:60               74          RTS
7D5F:                 70          INCLUDE SHADOW
```

```
7D5F:                3 ****************************************************************
7D5F:                4 *
7D5F:                5 *  File  :  SHADOW
7D5F:                6 *
7D5F:                7 ****************************************************************
7D5F:                8 *
7D5F:                9 *
7D5F:               10 *  This test checks shadow reg bit 0 (text page 1) by writing
7D5F:               11 *  a known pattern , then  checking for the presence of this
7D5F:               12 *  pattern depending on the state of the shadow register.
7D5F:               13 *
7D5F:               14 *
7D5F:               15 *  Modification history  :
7D5F:               16 *
7D5F:               17 *  10-Mar-86   :   JR  : Today
7D5F:               18 *
7D5F:               19 ****************************************************************
7D5F:        0000   20          SEG   $00          ;Reflect data bank

7D5F:        5A5A   22 ShadPatt1 EQU   $5A5A
7D5F:        A5A5   23 ShadPatt2 EQU   $A5A5
7D5F:        0400   24 ShdLenght EQU   $0400

7D5F:        000A   26 ShdStrt   EQU   $000A         ;Must be 4 bytes in Pg 0
7D5F:        0800   27 ShdData   EQU   $0800

7D5F:        7D5F   29 SHADOW.TST EQU  *
7D5F:9C 35 C0       30          STZ   SHADOW        ;Shadow all
7D62:               31          DATA16
7D62:C2 30           1+         REP   #%00110000  ;SET M=0, X=0 (16 BIT)
7D64:                2+         MEMORY16
7D64:                3+         INDEX16
7D64:                4+         MEND
7D64:A9 00 04       32          LDA   #$0400
7D67:85 0A          33          STA   ShdStrt       ;Txt page one
7D69:64 0C          34          STZ   ShdStrt+2
7D6B:A9 5A 5A       35          LDA   #ShadPatt1
7D6E:20 B1 7D       36          JSR   ShdWrite      ;Write patt # 1 to bank $00
7D71:E6 0C          37          INC   ShdStrt+2
7D73:20 B1 7D       38          JSR   ShdWrite      ;Write patt # 1 to bank $01
7D76:               39          MEM8
7D76:E2 20           1+         SEP   #%00100000   ;Set M=1 ( 8 bit memory )
7D78:                2+         MEMORY8
7D78:                3+         MEND
7D78:A9 01          40          LDA   #%00000001
7D7A:8D 35 C0       41          STA   SHADOW        ;Inhibit text shadowing
7D7D:               42          IND16
7D7D:C2 10           1+         REP   #%00010000   ;X=0 ( 16 bit index )
7D7F:                2+         INDEX16
7D7F:                3+         MEND
7D7F:A9 A5          43          LDA   #ShadPatt2
7D81:8D 00 08       44          STA   ShdData
7D84:20 B1 7D       45          JSR   ShdWrite      ;Write patt # 2 to bank $00
7D87:C6 0C          46          DEC   ShdStrt+2
7D89:20 B1 7D       47          JSR   ShdWrite      ;Write patt # 2 to bank $01

7D8C:20 BE 7D       49          JSR   ShdVerfy      ;Check bank $01
7D8F:B0 1A   7DAB   50          BCS   C035.Bad
```

```
7D91:E6 0C          51          INC   ShdStrt+2     ;Check bank $00
7D93:20 BE 7D       52          JSR   ShdVerfy
7D96:B0 13   7DAB   53          BCS   C035.Bad

7D98:A9 5A          55          LDA   #ShadPatt1
7D9A:8D 00 08       56          STA   ShdData
7D9D:A9 E0          57          LDA   #$00E0
7D9F:85 0C          58          STA   ShdStrt+2
7DA1:20 BE 7D       59          JSR   ShdVerfy      ;Check bank $E0
7DA4:B0 05   7DAB   60          BCS   C035.Bad
7DA6:E6 0C          61          INC   ShdStrt+2
7DA8:20 BE 7D       62          JSR   ShdVerfy      ;Check bank $E1
7DAB:        7DAB   63 C035.Bad EQU   *
7DAB:               64          DATA8
7DAB:E2 30           1+         SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
7DAD:                2+         MEMORY8
7DAD:                3+         INDEX8
7DAD:                4+         MEND
7DAD:9C 35 C0       65          STZ   SHADOW
7DB0:60             66          RTS

7DB1:               68 *
7DB1:               69 *   'Write' a pattern routine
7DB1:               70 *

7DB1:               72          Memory16
7DB1:               73          Index16
7DB1:A0 00 00       74 ShdWrite LDY   #$0000
7DB4:97 0A          75 ShdStore STA   [ShdStrt],Y ;Write directly to the bank
7DB6:C8             76          INY
7DB7:C8             77          INY
7DB8:C0 00 04       78          CPY   #ShdLenght  ;End of page ?
7DBB:D0 F7   7DB4   79          BNE   ShdStore
7DBD:60             80          RTS

7DBE:               82 *
7DBE:               83 *  'Verify' the data in any Bank
7DBE:               84 *

7DBE:A0 00 00       86 ShdVerfy LDY   #$0000
7DC1:B7 0A          87 ShdFetch LDA   [ShdStrt],Y ;Load directly from the bank
7DC3:CD 00 08       88          CMP   ShdDATA
7DC6:D0 09   7DD1   89          BNE   ShdFail
7DC8:C8             90          INY
7DC9:C8             91          INY
7DCA:C0 00 04       92          CPY   #ShdLENGHT
7DCD:D0 F2   7DC1   93          BNE   ShdFetch
7DCF:18             94          CLC
7DD0:60             95          RTS
7DD1:38             96 ShdFail  SEC
7DD2:60             97          RTS
7DD3:               71          INCLUDE BAT.RAM
```

```
7DD3:                3 ;.....................................................
7DD3:                4 ; BAT.RAM  (JSL)
7DD3:                5 ; ROM based Battery Ram Tests using Cortland tools
7DD3:                6 ;
7DD3:                7 ;
7DD3:                8 ; Errors: Carry Set,
7DD3:                9 ;         TST.STATUS+1 :  1=ADDRESS, TST.STATUS+2 = Failed address.
7DD3:               10 ;
7DD3:               11 ;         TST.STATUS+1 :  2=PATTTERN, TST.STATUS+2 = PATTERN
7DD3:               12 :                                     TST.STATUS+3 = FAILED ADDRESS
7DD3:               13 ;
7DD3:               14 ;  Created: 04 Mar 86, KG
7DD3:               15 ;
7DD3:               16 ;  Karl Grabe. Apple Computer Ltd.  Cork.
7DD3:               17 ;.....................................................

7DD3:         00FF  19           SEG   RUNBANK
7DD3:         7DD3  20 BAT.RAM   EQU   *
7DD3:               21           DATA16
7DD3:C2 30           1+           REP   #%00110000  ;SET M=0, X=0 (16 BIT)
7DD5:                2+           MEMORY16
7DD5:                3+           INDEX16
7DD5:                4+           MEND
7DD5:A9 00 08        22           LDA   #BRAM.TEMP   ;Used to hold NVR contents suring test
7DD8:85 10           23           STA   BAT.RAM.PTR  ;READ.NVR routine uses this
7DDA:64 12           24           STZ   BAT.RAM.PTR+2  ;Upper two bytes of pointer
7DDC:20 61 7E        25           JSR   READ.NVR     ;Copy it to BRAM.TEMP

7DDF:                27 ;Now do address uniqueness test
7DDF:                28           INDEX8
7DDF:                29           MEMORY8
7DDF:A0 FF           30           LDY   #255         :Entire buffer
7DE1:         7DE1   31 NXT.BAT.ADR EQU *
7DE1:98              32           TYA
7DE2:99 FF 08        33           STA   :BRAM.WORK-1,Y  ;Store unique value in each addr
7DE5:88              34           DEY
7DE6:D0 F9    7DE1   35           BNE   NXT.BAT.ADR  ;Do next bat ram address
7DE8:20 25 7E        36           JSR   WR.RD.CMP    ;Write to Bram, read and compare
7DEB:B0 23    7E10   37           BCS   BAT.RAM.ER2  ;Error so exit
7DED:A2 05           38           LDX   #5           :6 Patterns
7DEF:8E 27 03        39           STX   :CUR         ;Use as temp storage
7DF2:         7DF2   40 NXT.BAT.PAT EQU *           ;Do next battery pattern
7DF2:AE 27 03        41           LDX   :CUR         ;Get counter back
7DF5:BF 46 7E FF     42           LDA   =BAT.PAT,X
7DF9:                43 ;
7DF9:                44 ; FILL.BRAM.WORK
7DF9:                45 ; Fill BRAM.WORK buffer with accumulator
7DF9:                46 ;
7DF9:         7DF9   47 FILL.BRAM.WORK EQU *
7DF9:A0 FF           48           LDY   #255         :Entire buffer
7DFB:         7DFB   49 NXT.FILL.WORK EQU *
7DFB:99 FF 08        50           STA   :BRAM.WORK-1,Y  ;Fill byte
7DFE:88              51           DEY
7DFF:D0 FA    7DFB   52           BNE   NXT.FILL.WORK
7E01:20 25 7E        53           JSR   WR.RD.CMP    ;Write to Bram, read then compare.
7E04:B0 07    7E0D   54           BCS   BAT.RAM.ER1  ;Er so exit
7E06:CE 27 03        55           DEC   :CUR         ;Another Pattern ?
7E09:10 E7    7DF2   56           BPL   NXT.BAT.PAT  ;Try another pattern
```

```
7E0B:80 0E    7E1B   57          BRA   GOOD.BAT.RAM  :Restore Bram and exit

7E0D:EE 16 03        59 BAT.RAM.ER1 INC :TST.STATUS+1  ;Record error code
7E10:EE 16 03        60 BAT.RAM.ER2 INC :TST.STATUS+1  ;Rec er cd
7E13:8D 17 03        61          STA   :TST.STATUS+2  ;Address or pattern that failed
7E16:8C 18 03        62          STY   :TST.STATUS+3  ;Address of fail if pattern test
7E19:38              63          SEC                  ;Error flag
7E1A:6B              64          RTL                  ;Back to sequencer

7E1B:         7E1B   66 GOOD.BAT.RAM EQU *            ;Battery ram ok
7E1B:                67          DATA16
7E1B:C2 30            1+         REP   #%00110000  ;SET M=0, X=0 (16 BIT)
7E1D:                 2+         MEMORY16
7E1D:                 3+         INDEX16
7E1D:                 4+         MEND
7E1D:A2 00 08        68          LDX   #BRAM.TEMP   ;Where we saved NVR initially
7E20:20 4C 7E        69          JSR   WRITE.NVR    ;Go restore NVR to what was there before test
7E23:18              70          CLC                ;Clear error status
7E24:6B              71          RTL                ;..to sequencer
7E25:                72 ;.....................................................................
7E25:                73 ; WR.RD.CMP  8/8
7E25:                74 ; Write BRAM.WORK to Bram, read Bram to BRAM.WORK1
7E25:                75 ; then copmare BRAM.WORK to BRAM.WORK1
7E25:                76 ;
7E25:                77 ; Carry set if error,
7E25:                78 ; A = Pattern that failed
7E25:                79 ; Y = Address fail occurred
7E25:                80 ;.....................................................................
7E25:         7E25   81 WR.RD.CMP EQU   *
7E25:                82 ;
7E25:                83 ; WR.BRAM.WORK
7E25:                84 ;
7E25:                85 ; Write BRAM.WORK to Bram
7E25:                86 ;
7E25:                87          DATA16
7E25:C2 30            1+         REP   #%00110000  ;SET M=0, X=0 (16 BIT)
7E27:                 2+         MEMORY16
7E27:                 3+         INDEX16
7E27:                 4+         MEND
7E27:A2 00 09        88          LDX   #BRAM.WORK   ;Save this buffer to bram
7E2A:20 4C 7E        89          JSR   WRITE.NVR    ;Go save it
7E2D:                90 ;
7E2D:                91 ; RD.BRAM.WORK1
7E2D:                92 ;
7E2D:                93 ; Copy Bram into BRAM.WORK1
7E2D:                94 ;
7E2D:         7E2D   95 RD.BRAM.WORK1 EQU *
7E2D:A9 00 0A        96          LDA   #BRAM.WORK1  ;Now read back NVR into this buffer
7E30:85 10           97          STA   BAT.RAM.PTR  ;Tell READ.NVR routine where to read to
7E32:20 61 7E        98          JSR   READ.NVR

7E35:                100 ;
7E35:                101 ; Now compare BRAM.WORK1 to BRAM.WORK
7E35:                102 ;
7E35:         7E35   103 CMP.BRAM.WORK EQU *
7E35:                104         MEMORY8
7E35:                105         INDEX8
7E35:A0 FC           106         LDY   #252         ;But not checksum bytes (4)
```

```
7E37:          7E37  107 NXT.CMP.WORK EQU *
7E37:B9 FF 08        108          LDA   :BRAM.WORK-1,Y  ;Bram
7E3A:D9 FF 09        109          CMP   :BRAM.WORK1-1,Y  ;Cortalnd ram
7E3D:D0 05     7E44  110          BNE   BRAM.CMP.ER  ;No match so error
7E3F:88              111          DEY
7E40:D0 F5     7E37  112          BNE   NXT.CMP.WORK
7E42:18              113          CLC                ;Good status
7E43:60              114          RTS
7E44:          7E44  115 BRAM.CMP.ER EQU *           ;Bad status
7E44:38              116          SEC
7E45:60              117          RTS
7E46:AA 55 00 FF     118 BAT.PAT  DFB   $AA,$55,$00,$FF,$CC,$33
```

```
7E4C:                120 ;.....................................................................
7E4C:                121 ; WRITE.NVR 16/16
7E4C:                122 ; Write buffer pointed to by X to NVR
7E4C:                123 ;
7E4C:                124 ;.....................................................................
7E4C:                125           MEMORY16
7E4C:                126           INDEX16
7E4C:         7E4C   127 WRITE.NVR EQU   *
7E4C:                128 ;X already set up
7E4C:A0 C0 02        129           LDY   #BRAMBUFFER ;Normally in bank $e1
7E4F:A9 FF 00        130           LDA   #255        ;All of buffer
7E52:54 E1 00        131           MVN   $000000,^BRAMBUFFER  ;Move it into bank $e1
7E55:                132           DATA8
7E55:E2 30             1+          SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
7E57:                  2+          MEMORY8
7E57:                  3+          INDEX8
7E57:                  4+          MEND
7E57:22 80 00 E1     133           JSL   =TOWRITEBR  ;Go write it
7E5B:20 DF 76        134           JSR   BANK.0      ;Data bank = 0
7E5E:                135           DATA16
7E5E:C2 30             1+          REP   #%00110000  ;SET M=0, X=0 (16 BIT)
7E60:                  2+          MEMORY16
7E60:                  3+          INDEX16
7E60:                  4+          MEND
7E60:60              136           RTS
7E61:                137 ;.....................................................................
7E61:                138 ; READ.NVR XX/8
7E61:                139 ; Copy NVR into a 4 byte buffer ppointed to by BAT.RAM.PTR
7E61:                140 ;
7E61:                141 ;.....................................................................

7E61:         7E61   143 READ.NVR  EQU   *
7E61:                144           DATA8
7E61:E2 30             1+          SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
7E63:                  2+          MEMORY8
7E63:                  3+          INDEX8
7E63:                  4+          MEND
7E63:8B              145           PHB               ;Save current data bank reg
7E64:08              146           PHP               ;DISABLE INTERRUPTS DURING WRITE
7E65:78              147           SEI
7E66:A2 00           148           LDX   #$00        ;Start with 1st byte
7E68:9B              149           TXY               ;Start with address 00
7E69:         7E69   150 READBRLP  EQU   *
7E69:E2 40           151           SEP   #$40        ;Set v for read
7E6B:20 7D 7E        152           JSR   XTNDBRAM    ;Read a byte
7E6E:5A              153           PHY               ;Save y
7E6F:9B              154           TXY
7E70:97 10           155           STA   [BAT.RAM.PTR],Y  ;Save for later
7E72:7A              156           PLY               ;Now restore y

7E73:E8              158           INX               ;Next
7E74:C8              159           INY               ;Next address
7E75:D0 F2    7E69   160           BNE   READBRLP

7E77:28              162           PLP               ;Restore interrupt status
7E78:18              163           CLC               ;Data intact indicator
7E79:E2 30           164           SEP   #$30        ;M,X back to 8 bit mode
7E7B:AB              165           PLB               ;Restore data bank register
```

```
7E7C:60              166          RTS

7E7D:          7E7D  168 XTNDBRAM  EQU  *          ;Set up command and extended address
7E7D:48              169          PHA             ;Save data to wr to clock register
7E7E:98              170          TYA             ;Create extended address
7E7F:48              171          PHA             ;Save for later
7E80:29 E0           172          AND  #$E0       ;Get sector data
7E82:4A              173          LSR  A          ;Move to bits 012
7E83:4A              174          LSR  A
7E84:4A              175          LSR  A
7E85:4A              176          LSR  A
7E86:4A              177          LSR  A
7E87:09 38           178          ORA  #EXMEMCMD  ;Sets command corerectly ($38)
7E89:50 02     7E8D  179          BVC  XTNDBRAM4  ;V=0 then writing
7E8B:09 80           180          ORA  #$80       ;Modify command for reading

7E8D:          7E8D  182 XTNDBRAM4 EQU  *
7E8D:EB              183          XBA             ;Move in hi byte of a
7E8E:68              184          PLA             ;Recall adr
7E8F:29 1F           185          AND  #$1F       ;Strip off sector bits
7E91:0A              186          ASL            ;Shift bits 2 to 6
7E92:0A              187          ASL  A
7E93:EB              188          XBA             ;Save address/do command
7E94:08              189          PHP             ;Save user read/write command
7E95:20 AC 7E        190          JSR  ACCCLKCLV  ;Transfer command to clock chip
7E98:EB              191          XBA             ;Send address to write to, to clock chip
7E99:          7E99  192 XTNDBRAM6 EQU  *
7E99:20 AC 7E        193          JSR  ACCCLKCLV  ;Transfer extended adr to clk chip
7E9C:28              194          PLP             ;Recall what test sequencer wants
7E9D:68              195          PLA             ;Recall sequencer data if writing
7E9E:20 AD 7E        196          JSR  ACCCLOCK   ;Read/write byte of data
7EA1:48              197          PHA
7EA2:AD 34 C0        198          LDA  :CLOCKCTL  ;Reset bit
7EA5:29 DF           199          AND  #$DF
7EA7:8D 34 C0        200          STA  :CLOCKCTL
7EAA:68              201          PLA
7EAB:60              202          RTS

7EAC:          7EAC  204 ACCCLKCLV EQU  *          ;Write a byte of data to clock chip
7EAC:B8              205          CLV             ;For write data
7EAD:          7EAD  206 ACCCLOCK  EQU  *          ;Send a byte of data to the clock chip
7EAD:8D 33 C0        207          STA  :CLOCKDATA ;Data/address/cmd to clock data register
7EB0:AD 34 C0        208          LDA  :CLOCKCTL  ;Reset read/write bit
7EB3:29 3F           209          AND  #$3F       ;Bit 7/6 to 0
7EB5:70 04     7EBB  210          BVS  ACCCLOCK0  ;V=1 IF READING
7EB7:09 A0           211          ORA  #$A0       ;Bit 7/5 to 1 to write transfer
7EB9:80 02     7EBD  212          BRA  ACCCLOCK1

7EBB:          7EBB  214 ACCCLOCK0 EQU  *          ;Doing a read of battery ram if here
7EBB:09 E0           215          ORA  #$E0       ;Enable transfer for reading
7EBD:          7EBD  216 ACCCLOCK1 EQU  *          ;
7EBD:8D 34 C0        217          STA  :CLOCKCTL  ;
7EC0:          7EC0  218 ACCCLOCK2 EQU  *          ;Wait till transaction is done
7EC0:AD 34 C0        219          LDA  :CLOCKCTL  ;Wait for transfer to complete
7EC3:30 FB     7EC0  220          BMI  ACCCLOCK2  ;Loop till done
7EC5:AD 33 C0        221          LDA  :CLOCKDATA ;Data is in a on exit
7EC8:60              222          RTS
7EC9:                 72          INCLUDE FDB.EQU
```

```
7EC9:                3 ;.....................................................................
7EC9:                4 ; FDB.EQU
7EC9:                5 ; Front Desk Bus Test Equates
7EC9:                6 ; KG
7EC9:                7 ;.....................................................................

7EC9:          00FF   9           SEG   RUNBANK
7EC9:          1400  10 FDB.ROM.B EQU   $1400        ;Start of rom address
7EC9:          1FFF  11 FDB.ROM.E EQU   $1FFF        :End

7EC9:                13 ; FDB rom checksums:
7EC9:          3DE6  14 FDB.REV1  EQU   $3DE6        :No longer supported kg/apr/86
7EC9:          F772  15 FDB.REV2  EQU   $F772
7EC9:          2672  16 FDB.REV3  EQU   $2672        :Checksum for all new rom releases

7EC9:                18 ;
7EC9:                19 ;FDB Tool table for RDMEM
7EC9:                20 ;
0000:                21           DSECT
0000:          00FF  22           SEG   RUNBANK
802C:          802C  23           ORG   11*4+$8000
802C:          802C  24 FDB.TOOL.ADDR EQU *
7EC9:                25           DEND
7EC9:                73           INCLUDE FDB
```

```
7EC9:                3 ;.....................................................
7EC9:                4 ; FDB
7EC9:                5 ; Test the FDB processor rom
7EC9:                6 ;
7EC9:                7 ;
7EC9:                8 ; Created : 27 Feb 86
7EC9:                9 ; Modified : 19 Mar 86 for rom
7EC9:               10 ; Modified : 10 Apr 86 use FDB tool call
7EC9:               11 ;
7EC9:               12 ; Routine to read FDB rom into main ram
7EC9:               13 ; starting at FDB.BUF and then do a checksum.
7EC9:               14 ;
7EC9:               15 ; Call/RET: NTV16
7EC9:               16 ;
7EC9:               17 ; Output: Carry set if error, TST.STATUS+1,+2 CONTAINS BAD CHECKSUM
7EC9:               18 ;
7EC9:               19 ; By Karl Grabe Apple Computer Ltd., Cork by the Lee
7EC9:               20 ;.....................................................

7EC9:        7EC9   22 FDB       EQU   *
7EC9:        0000   23           SEG   $00
7EC9:               24           DATA16
7EC9:C2 30           1+          REP   #%00110000  ;SET M=0, X=0 (16 BIT)
7ECB:                2+          MEMORY16
7ECB:                3+          INDEX16
7ECB:                4+          MEND
7ECB:A9 00 14       25           LDA   #FDB.ROM.B   ;Start address
7ECE:8D 00 0B       26           STA   POINTER
7ED1:A9 04 0B       27           LDA   #FDB.BUF     ;Where to stash in main mem
7ED4:85 10          28           STA   BAT.RAM.PTR
7ED6:               29 NXT.FDB.BYTE DATA8
7ED6:E2 30           1+          SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
7ED8:                2+          MEMORY8
7ED8:                3+          INDEX8
7ED8:                4+          MEND
7ED8:20 15 7F       30           JSR   R.FDB        ;Read a byte
7EDB:B0 2D   7F0A   31           BCS   FDB.TOOL.ER  ;Tool had a problem reading up
7EDD:92 10          32           STA   (BAT.RAM.PTR)  ;Save it
7EDF:               33           DATA16
7EDF:C2 30           1+          REP   #%00110000  ;SET M=0, X=0 (16 BIT)
7EE1:                2+          MEMORY16
7EE1:                3+          INDEX16
7EE1:                4+          MEND
7EE1:E6 10          34           INC   BAT.RAM.PTR
7EE3:EE 00 0B       35           INC   POINTER      ;FDB rom address
7EE6:AD 00 0B       36           LDA   POINTER
7EE9:C9 00 20       37           CMP   #FDB.ROM.E+1  ;Are we at end of rom yet ?
7EEC:30 E8   7ED6   38           BMI   NXT.FDB.BYTE
7EEE:        7EEE   39 FDB.CHECK.TST EQU *          Debug entry point (NTV16)
7EEE:A9 04 0B       40           LDA   #FDB.BUF     ;Start of address where FDB rom code is
7EF1:85 FD          41           STA   CHK.BEG
7EF3:A0 FF 0B       42           LDY   #FDB.ROM.E-FDB.ROM.B  ;End address
7EF6:20 EA 7C       43           JSR   CAL.CHECKSUM  ;Go do checksum (data bank is zero)
7EF9:C9 72 F7       44           CMP   #FDB.REV2
7EFC:F0 0A   7F08   45           BEQ   FDB.GOOD.CHK  ;It's rev 2
7EFE:C9 72 26       46           CMP   #FDB.REV3    ;All other revs must equal this
7F01:F0 05   7F08   47           BEQ   FDB.GOOD.CHK
7F03:8D 16 03       48           STA   TST.STATUS+1  ;Record the bad checksum for display
```

```
7F06:38             49          SEC                 ;Error
7F07:6B             50          RTL                 ;Callable from disk diagnostics also
7F08:         7F08  51 FDB.GOOD.CHK EQU *
7F08:18             52          CLC
7F09:6B             53          RTL                 ;Callable from disk diagnostics also
7F0A:         7F0A  54 FDB.TOOL.ER EQU *            ;Tool had problem reading uC
7F0A:A9 01 00       55          LDA   #1            ;Tool problem
7F0D:8D 18 03       56          STA   TST.STATUS+3  ;Last byte of status
7F10:8D 16 03       57          STA   TST.STATUS+1  ;The error tool encountered
7F13:38             58          SEC                 ;Bad status
7F14:6B             59          RTL
```

```
7F15:        0001   61          DO   USE.FDB.TOOL  ;Use following code if FDB tool code is in rom
7F15:               62 ********************************************************************
7F15:               63 *
7F15:               64 * Read a byte from the FDB Processor Memory (RAM/ROM)
7F15:               65 *                      .... Using Vegas Tool code
7F15:               66 *
7F15:               67 * Call: in 8 bit mode
7F15:               68 * RTN : in 8 bit mode
7F15:               69 *
7F15:               70 * Inputs:
7F15:               71 *       (POINTER) is byte to read
7F15:               72 *       ROM in range $1400-1FFF
7F15:               73 *       RAM in range $0000-0050
7F15:               74 *
7F15:               75 * Output:
7F15:               76 *       ACC containes byte
7F15:               77 *
7F15:               78 ********************************************************************

7F15:        7F15   80 R.FDB    EQU  *
7F15:               81         MEM16            ;Into 16 bit memory
7F15:C2 20           1+         REP  #%00100000  :M=0 (16 bit memory)
7F17:                2+         MEMORY16
7F17:                3+         MEND
7F17:               82         INDEX8
7F17:A9 00 00       83         LDA  #0          ;STZ has no long address mode
7F1A:8F D6 0F E1    84         STA  =FLAGS      ;Set these to zero for tool
7F1E:8F D8 0F E1    85         STA  =FLAGS+2
7F22:8F DA 0F E1    86         STA  =FLAGS+4
7F26:F4 00 00       87         PEA  ^RESULT
7F29:F4 02 0B       88         PEA  :RESULT     ;Where to stash result
7F2C:F4 00 00       89         PEA  ^POINTER    ;And uppr
7F2F:F4 00 0B       90         PEA  :POINTER    ;Tell tool wher i/p is
7F32:F4 09 00       91         PEA  9           ;The command no
7F35:48             92         PHA              ;Push 3 dummy bytes
7F36:8B             93         PHB              ;3rd byte
7F37:               94 ;
7F37:               95 ; Set up stack to RTL from tool
7F37:A2 FF          96         LDX  #^FDB.TOOL.RTN  ;So we get good rtl from tool
7F39:DA             97         PHX              ;Bank address
7F3A:F4 4A 7F       98         PEA  :FDB.TOOL.RTN-1
7F3D:               99 ;
7F3D:              100 ; Set up stack to RTL to tool instead of JSL
7F3D:        0000  101         SEG  $00
7F3D:AF 2E 80 FF   102         LDA  =FDB.TOOL.ADDR+2  ;Push 3 bytes RTL address
7F41:AA            103         TAX
7F42:DA            104         PHX
7F43:AF 2C 80 FF   105         LDA  =FDB.TOOL.ADDR
7F47:48            106         PHA              ;Get ready to RTL to tool entry point
7F48:              107         DATA16           ;Back to index 16 (already in mem16)
7F48:C2 30           1+        REP  #%00110000  ;SET M=0, X=0 (16 BIT)
7F4A:                2+        MEMORY16
7F4A:                3+        INDEX16
7F4A:                4+        MEND
7F4A:6B            108         RTL              ;To Front Desk Bus tool
7F4B:        00FF  109         SEG  RUNBANK
7F4B:        7F4B  110 FDB.TOOL.RTN EQU *        ;Tool will RTL here
7F4B:              111 :Now pull off garbage bytes
```

```
7F4B:                112          DATA8
7F4B:E2 30             1+         SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
7F4D:                  2+         MEMORY8
7F4D:                  3+         INDEX8
7F4D:                  4+         MEND
7F4D:68              113          PLA
7F4E:68              114          PLA
7F4F:68              115          PLA
7F50:AD 02 0B        116          LDA   :RESULT      :Get rom contents
7F53:60              117          RTS
```

```
119           ELSE
120           REPT  40
121 *
122 *         Read  a           byte from the FDB Processor Memory (RAM/ROM)
123 *                                        ... Don't use FDB Tool code
124 *
125 *         Call: in          8 bit mode
126 *         RTN   :           in 8 bit mode
127 *
128 *         Inputs:
129 *                                (POINTER) is byte to read
130 *                               ROM in range $1400-1FFF
131 *                               RAM in range $0000-0050
132 *
133 *         Output:
134 *                               ACC containes byte
135 *
136           REPT  40
137           SKP   1
138           DATA8
139 R.FDB     LDA   #%00001001   ;Command byte to read UP memory
140           JSR   FDB.CMD      ;Do a store to FDB micro, rtn when ready
141           LDA   POINTER      ;the low byte of rom start address
142           JSR   FDB.CMD      ;Store lo address
143           LDA   POINTER+1    ;Hi address
144           JSR   FDB.CMD      ;Do hi
145           JSR   FDB.RDATA    ;Read data & wait till rdy
146           CLC               ;Good status
147           RTS
148           SKP   2
149 ;.....................................................................
150 ;         FDB.CMD
151 ;         Write a           byte to FDB processor
152 ;
153 ;         Store acc         in FDB command register and wait
154 ;         'till ready       for next command/data
155 ;
156 ;         Input Accumulator
157 ;
158 ;         8     bit         mode
159 ;.....................................................................
160           SKP   1
161 FDB.CMD   EQU   *
162           PHP               ;Save interrupt status
163           SEI               ;No interrupts
164           PHA               ;Save the command/data byte
165 FDB.WT.1  LDA   KMSTATUS     ;Get status for command accept
166           ROR   A           ;Put lsb in carry
167           BCS   FDB.WT.1
168           PLA
169           STA   DATAREG     ;Command register for write
170 FDB.WT    LDA   KMSTATUS     ;Get status for command accept
171           ROR   A           ;Put lsb in carry
172           BCS   FDB.WT      ;Processor hasn't accepted command yet
173           PLP               ;Restore interrupt status
174           RTS
175           PAGE
```

```
S          177 ;        FDB.RDATA
S          178 ;        Read  a          byte from FDB processor
S          179 ;
S          180 ;        Read  DATAREG    register ( Wait till data valid )
S          181 ;
S          182 ;        Output :         Accumulator
S          183 ;
S          184 ;        8     bit        mode
S          185 ;
S          186 ;.....................................................
S          187          SKP   1
S          188 FDB.RDATA equ  *
S          189          PHP              ;Save interrupt status
S          190          SEI              ;No interupts allowed
S          191          LDA   DATAREG     ;Clear out any previous data
S          192 FDB.NVR  LDA   #%00100000  ;This bit 1 when data valid
S          193          AND   KMSTATUS    ;Is data valid yet
S          194          BEQ   FDB.NVR     ;Not valid if bit zero
S          195          LDA   DATAREG     ;Get the FDB DATA
S          196          PLP
S          197          RTS
7F54:      198          FIN
```

```
7F54:                74          INCLUDE RAM.ADDR
```

```
7F54:                    3 *
7F54:                    4 * File: RAM.ADDR
7F54:                    5 *
7F54:                    6 * Routines to test the addressing of all onboard RAM
7F54:                    7 *  taking care to notice the Bank Address in the test.
7F54:                    8 *
7F54:                    9 *
7F54:                   10 *  20 Mar 86    RCarr    New Today!
7F54:                   11 *
7F54:                   12 *
7F54:         000A      13 xtemp     equ   10
7F54:                   14 *
7F54:         00FF      15           SEG   RUNBANK
7F54:         7F54      16 RAM.ADDR  EQU   *
7F54:         0000      17           SEG   $00
7F54:         7F54      18 DoNewAddr equ   *
7F54:A9 5F              19           LDA   #%01011111
7F56:8D 35 C0           20           STA   SHADOW       ;All shadowing off
7F59:A2 03              21           ldx   #3
7F5B:BF FC 7F FF        22 Wlp1      lda   =Banktbl,x
7F5F:20 77 7F           23           jsr   WriteAddr
7F62:CA                 24           dex
7F63:10 F6    7F5B      25           bpl   Wlp1
7F65:                   26 *
7F65:A2 03              27           ldx   #3
7F67:BF FC 7F FF        28 Clp1      lda   =Banktbl,x
7F6B:20 A2 7F           29           jsr   CmpAddr
7F6E:B0 03    7F73      30           bcs   NewAddrErr  ;fall thru w/ carry set
7F70:CA                 31           dex
7F71:10 F4    7F67      32           bpl   Clp1
7F73:2C 81 C0           33 NewAddrErr bit  $c081       ;off LCard memory
7F76:6B                 34           rtl
7F77:                   35 *
7F77:         7F77      36 WriteAddr equ   *
7F77:85 F0              37           sta   RamBank
7F79:85 06              38           sta   loc+2
7F7B:86 0A              39           stx   xtemp
7F7D:                   40           data16
7F7D:C2 30               1+          REP   #%00110000  ;SET M=0, X=0 (16 BIT)
7F7F:                    2+          MEMORY16
7F7F:                    3+          INDEX16
7F7F:                    4+          MEND
7F7F:64 04              41           stz   loc         ;sets up the indirection
7F81:A0 00 04           42           ldy   #$400       ;start addr
7F84:A2 FE BF           43           ldx   #$bffe      ;end addr
7F87:20 CF 7F           44           jsr   writeit
7F8A:                   45           mem8
7F8A:E2 20               1+          SEP   #%00100000   ;Set M=1 ( 8 bit memory )
7F8C:                    2+          MEMORY8
7F8C:                    3+          MEND
7F8C:AD 8B C0           46           lda   lcbank1     ;do lc bank 1 only
7F8F:AD 8B C0           47           lda   lcbank1
7F92:                   48           mem16
7F92:C2 20               1+          REP   #%00100000   ;M=0 (16 bit memory)
7F94:                    2+          MEMORY16
7F94:                    3+          MEND
7F94:A0 00 D0           49           ldy   #$d000
7F97:A2 FE FF           50           ldx   #$fffe
```

```
7F9A:20 CF 7F          51          jsr   writeit
7F9D:                  52          data8
7F9D:E2 30              1+         SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
7F9F:                   2+         MEMORY8
7F9F:                   3+         INDEX8
7F9F:                   4+         MEND
7F9F:A6 0A             53          ldx   xtemp
7FA1:60                54          rts
7FA2:                  55 *
7FA2:          7FA2    56 CmpAddr  equ   *
7FA2:85 F0             57          sta   Rambank
7FA4:85 06             58          sta   loc+2
7FA6:86 0A             59          stx   xtemp
7FA8:                  60          data16
7FA8:C2 30              1+         REP   #%00110000  ;SET M=0, X=0 (16 BIT)
7FAA:                   2+         MEMORY16
7FAA:                   3+         INDEX16
7FAA:                   4+         MEND
7FAA:64 04             61          stz   loc         ;to be sure
7FAC:A0 00 04          62          ldy   #$400       :start of addr check
7FAF:A2 FE BF          63          ldx   #$bffe
7FB2:20 E0 7F          64          jsr   cmpit
7FB5:B0 13     7FCA    65          bcs   CmpFail
7FB7:                  66          mem8
7FB7:E2 20              1+         SEP   #%00100000   ;Set M=1 ( 8 bit memory )
7FB9:                   2+         MEMORY8
7FB9:                   3+         MEND
7FB9:AD 8B C0          67          lda   lcbank1
7FBC:AD 8B C0          68          lda   lcbank1
7FBF:                  69          mem16
7FBF:C2 20              1+         REP   #%00100000   ;M=0 (16 bit memory)
7FC1:                   2+         MEMORY16
7FC1:                   3+         MEND
7FC1:A0 00 D0          70          ldy   #$d000
7FC4:A2 FE FF          71          ldx   #$fffe
7FC7:20 E0 7F          72          jsr   cmpit
7FCA:                  73 CmpFail  data8             ;carry is set on error
7FCA:E2 30              1+         SEP   #%00110000  ;SET M=1, X=1 (8 BIT )
7FCC:                   2+         MEMORY8
7FCC:                   3+         INDEX8
7FCC:                   4+         MEND
7FCC:A6 0A             74          ldx   xtemp
7FCE:60                75          rts
7FCF:                  76 *
7FCF:          7FCF    77 writeit  equ   *
7FCF:86 F2             78          stx   RamEndAddr
7FD1:BB                79 writelp1 tyx
7FD2:98                80          tya
7FD3:18                81          clc
7FD4:65 F0             82          adc   RamBank
7FD6:9B                83          txy
7FD7:97 04             84          sta   [loc],y
7FD9:C8                85          iny
7FDA:C8                86          iny
7FDB:C4 F2             87          cpy   RamEndAddr
7FDD:90 F2     7FD1    88          bcc   writelp1
```

```
7FE0:          7FE0   91 cmpit     equ   *
7FE0:86 F2            92           stx   RamEndAddr
7FE2:BB               93 cmplp1    tyx
7FE3:98               94           tya
7FE4:18               95           clc
7FE5:65 F0            96           adc   RamBank
7FE7:9B               97           txy
7FE8:D7 04            98           cmp   [loc],y
7FEA:D0 08     7FF4   99           bne   cmperror
7FEC:C8              100           iny
7FED:C8              101           iny
7FEE:C4 F2           102           cpy   RamEndAddr
7FF0:90 F0     7FE2  103           bcc   cmplp1
7FF2:18              104           clc
7FF3:60              105           rts
7FF4:                106 *
7FF4:          7FF4  107 cmperror  equ   *
7FF4:8D 16 03        108           sta   ErrCode1    ;16 bit write is ok!
7FF7:8C 17 03        109           sty   ErrCode2    ; as this covers it
7FFA:38              110           sec
7FFB:60              111           rts
7FFC:                112 *
7FFC:                113 *
7FFC:          00FF  114           SEG   RUNBANK     ;Table is in ROM
7FFC:00 01 E0 E1     115 Banktbl   dfb   $0,$1,$e0,$e1 :table of banks to check
8000:          0000   75           DS    $8000-*,$00
```

```
?   3D A1H        ?   3C A1L        ?   3F A2H        ?   3E A2L        ?   41 A3H        ?   40 A3L
?   43 A4H        ?   42 A4L        ?   45 A5H        ?E10018 ABORTMGRV  FF7EAC ACCCLKCLV  FF7EBB ACCCLOCK0
FF7EBD ACCCLOCK1  FF7EC0 ACCCLOCK2  FF7EAD ACCCLOCK    7A1C AGAIN       ?010101 ALEMSTKPTR   79CF ALLSENT
  782A ALT16        C01E ALTCHARSET ?E10340 APTALKNODE ?E10133 ASCFILTER ?E10108 ASCIITIME  ?E10120 AVAL
  747E BAD.NVR      FF7CDF BAD.RAM   ?FF7CD2 BAD.ROM    FF7B1E BAD.SPEED    7987 BADDATA      76DF BANK.0
  76E5 BANK.RUN     FF775E BANK.ZERO.BE  776C BANK.ZERO.EN?E1013F BANKD   ?E1013E BANKS      FF7FFC BANKTBL
?   2B BAS2H      ?   2A BAS2L      ?   29 BASH       ?  E000 BASIC     ?  E003 BASIC2    ?   28 BASL
FF7E46 BAT.PAT    FF7DD3 BAT.RAM    FF7E0D BAT.RAM.ER1 FF7E10 BAT.RAM.ER2    10 BAT.RAM.PTR    00 BAUD.SWITCH
  0313 BI.ALT.MASK  0306 BI.COUNTER   75A4 BI.CYCLE.FAI  031B BI.FAIL      030A BI.FAILS     030C BI.LASTF
  FFFF BI.MASK      0308 BI.PASSES  ?  7489 BI         ?  C063 BI.FLAG     FF3F BI.SELF.MASK  7490 BI.SELF.TST
  76DC BI.SLF.NVR   0300 BI.STATUS    FF7E44 BRAM.CMP.ER  0800 BRAM.TEMP    0A00 BRAM.WORK1   0900 BRAM.WORK
E102C0 BRAMBUFFER ?E103BC BRAMCHKSUM1 ?E103BE BRAMCHKSUM2 ?  03F0 BRKV     ?E100FF BUSYFLAG    C061 BUTN0
  C062 BUTN1        C063 BUTN2      ?  C060 BUTN3     ?E1012B BVAL      7DAB C035.BAD    FF7CEA CAL.CHECKSUM
FF7CEE CAL.CHECKSUM   3C CHAINDEX   ?  C02C CHARROM   ?  067B CHAR        3B CHBINDEX   ?  080B CHDTR
    00 CHECKSUM   ?E10136 CHGCURFLG ?  080A CHHSK        FD CHK.BEG     FF7CF7 CHK.END     7A02 CHKDATA
  7630 CHKOAONLY    79EE CHKRD        79E8 CHKREAD   ?   24 CH        ?  AAAA CHKSMCNST   79AC CHN1
FF7D24 CLK.CONT   FF7D4A CLK.ERR    E103E1 CLKRDATA   E103E5 CLKWDATA    C034 CLOCKCTL     C033 CLOCKDATA
FF7CFA CLOCK      7F67 CLP1       FF7A9B CLR.SW      C000 CLR80COL    C00C CLR80VID     C00E CLRALTCHAR
  C059 CLRAN0       C05B CLRAN1   ?  C05D CLRAN2   ?  C05F CLRAN3     CFFF CLRROM    ?  C06E CLRTM
  C047 CLRVBLINT  ?  C048 CLRXYINT ?FF7E35 CMP.BRAM.WOR  7FA2 CMPADDR    7FF4 CMPERROR     7FCA CMPFAIL
  7FE0 CMPIT        7FE2 CMPLP1     0800 CNT1         0801 CNT2        0802 CNT3      ?  0803 CNT4
?   30 COLOR      758F CONT.CYC   ?E10014 COPMGRV   FF7749 COUT.PORT1 ?   37 CSWH    ?   36 CSWL
?E10060 CUPDATE     0327 CUR      ?E10134 CURSOR     7B46 CUS.IRQ.RTL   7000 CUSIRQCNT    7006 CUSIRQSRCE
  7002 CUSIRQSTS    7004 CUSLINEND   7B3E CUSTOM.IRQ ?   25 CV        C036 CYAREG       0321 CYCLE.COUNT
  030E CYCLES.MAX M   B5 DATA16   M   F5 DATA8      080C DATA      ?  C026 DATAREG   ?E10068 DECBUSYFLG
?FF7400 DIAGS     C046 DIAGTYPE   ?E10142 DIGITCNT  ?  C031 DISKREG   ?E10000 DISPATCH1  ?E10004 DISPATCH2
?  C037 DMAREG  ?FF773F DO.INIT    FF75B2 DO.TEST   ?FF774D DO.WRITE    77FF DOMOVITEST  ?  7F54 DONEWADDR
  786B DOZPAGE00    78C7 DOZPAGEIND ?E10128 DVAL    ?E1013A E1PCLH   ?   77 EML       FF775E END.PORT.PAR
  7625 END.SELF     757A END.TST.SEQ   795E ENDRWTST  ?E100D6 ENSONIQFLG ?  C06F ENTM    ?E10132 EPVAL
  0316 ERRCODE1     0317 ERRCODE2     7864 ERROR16     7C65 ERROR2    ?E100E1 EVMGRDATA     38 EXMEMCMD
  749F EXT.SEQ  ?E10140 EXTENDBNK ?   06 F8VERSION   7565 FAIL.BI.CONT  C0E2 FAIL.LEDD    C059 FAIL.LED
FF752D FAIL.SELF.TO  7517 FAIL.SELF   FF7558 FAIL.SELF.MS  7947 FAILRW      191A FAST.LIM  M  0236 FAST
  0B04 FDB.BUF      7F08 FDB.GOOD.CHK?  3DE6 FDB.REV1     2672 FDB.REV3   FF7EC9 FDB     ?  7EEE FDB.CHECK.TS
  F772 FDB.REV2     1400 FDB.ROM.B    1FFF FDB.ROM.E   FFB02C FDB.TOOL.ADD  7F0A FDB.TOOL.ER  FF7F4B FDB.TOOL.RTN
?FF7DF9 FILL.BRAM.WO   03 FILL2IOMSK    04 FILL2MONVAR    08 FILL2MSGPTR    10 FILL2MVCTRS   759A FIN.CYCLES
 FF7B26 FIND.SPEED ?E10146 FIXST1    ?E10148 FIXST2   ?E1014A FIXST3    E10FD6 FLAGS       0B FLIP
?   2E FORMAT     77FE FOUNDMOVIERR  7AF9 FPI.SPEED  ?   27 GBASH     ?   26 GBASL     FF7E1B GOOD.BAT.RAM
 FF7CD0 GOOD.ROM   FF7B1C GOOD.SPEED ?  06 GOODF8    ?E1013C GOS      ?   2C H2      ?E100D8 HBPTR
  C057 HIRES      ?E102BC HIXCLAMP  ?E102BE HIYCLAMP  ?  C02F HORIZCNT    7762 IJ        FF7AB4 INC.SW.1
?FF7AB3 INC.SW    ?E10064 INCBUSYFLG   7CA7 INCCNT   M  0133 IND16    M  019F IND8       10 INIT.JMP
  C10D INIT.OFF     FF7702 INIT.PORT1 ?FE0000 INITDISMGR ?E100D0 INMASK    C041 INTEN       798A INTLOOP
  79A4 INTLOOPERR ?  0200 IN          C046 INTFLAG    E10010 INTMGRV    7B47 INV16     ?   32 INVFLG
  C000 IOADR      ?E10054 IRQ.1SEC  ?E100CB IRQ.ACTIVE  E10108 IRQ.A    E10020 IRQ.APTLK  ?E100C9 IRQ.APTLKHI
?E10113 IRQ.DB    ?E10110 IRQ.D     ?E100C4 IRQ.DATAREG ?E10048 IRQ.DSKACC ?E10114 IRQ.E    ?E10058 IRQ.EXT
?E1004C IRQ.FLUSH ?E100C3 IRQ.INTFLAG ?E1003C IRQ.KBD  ?E10115 IRQ.K    ?E100C5 IRQ.KMSTATUS?E10060 IRQ.LEN
?E10050 IRQ.MICRO ?E10034 IRQ.MOUSE ?E100D7 IRQ.MOUSVBL ?E1011B IRQ.MSLOT ?E1005C IRQ.OTHER  ?E10116 IRQ.PC
?E10112 IRQ.P     ?E10038 IRQ.QTR   ?E10040 IRQ.RESPONSE?E10028 IRQ.SCAN  ?E100C6 IRQ.SECFLAG ?E100C7 IRQ.SERIAL1
?E10119 IRQ.SHADOW ?E1010E IRQ.S    ?E100CB IRQ.SERIAL2 ?E10024 IRQ.SERIAL ?E1002C IRQ.SOUND  ?E10044 IRQ.SRQ
?E10118 IRQ.STATE ?E10030 IRQ.VBL   ?E100CA IRQ.VOLUME  ?E1010A IRQ.X     ?E1010C IRQ.Y     ?E1000C IRQLEN
?  03FE IRQLOC      7008 IRQTEMP1     700A IRQTEMP2     700C IRQTEMP3     7851 JMP16     ?  C000 KBD
?  C010 KBDSTRB   ?  C025 KEYMODREG ?  760D KG.WAIT    760E KG.WAIT2    760F KG.WAIT3  ?  C027 KMSTATUS
?   39 KSWH      ?   38 KSWL       ?E1012C KVAL      ?E102F8 LANGS    ?  C02B LANGSEL   ?E10301 LAYOUTS
?FF7AF9 LBL00     FF7B2C LBL01      FF7B36 LBL02      C08B LCBANK1      C083 LCBANK2   ?   2F LENGTH
?   FF LIMIT        0400 LINE1    ?   2C LMNEM    ?   00 LOC0      ?   01 LOC1       04 LOC
   7B84 LOOP001     7BB2 LOOP002      7C42 LOOP003      C056 LORES    ?E102B8 LOWXCLAMP ?E102BA LOWYCLAMP
?E1012F LVAL      ?   02 M.1      ?   04 M.2      ?   10 M.4      ?   40 M.6      ?   08 M.CTL
?   20 M.CTL2     ?   10 M.CURSOR ?   08 M.GOXY   ?   01 M.MOUSE  ?   02 M.PAS1.0 ?   80 M.PASCAL
?   04 M.VMODE    ?   20 M8       ?   2E MASK       080F MASKDATA   FF7AC0 MEBA.SOFT.ER  7BD0 MEGAERR
```

```
FF7BF1 MEGAINTHND     7BEF MEGAPASS       7B47 MEGATSTIRQ     7BD3 MEGRSTRE    M  0169 MEM16       M  01D6 MEMB
  C052 MIXCLR         C053 MIXSET      ?  C044 MMDELTAX    ?  C045 MMDELTAY   ?010100 MNEMSTKPTR  ?  04FB MODE
    09 MOD         ?    31 MONMODE          C021 MONOCOLOR    ?  C024 MOUSEDATA  ?E100E0 MOUSESLOT      77C2 MOVIRAMEXIT
  77CD MOVIRAMTEST    776C MOVIRAM     ?E10130 MPVAL       ?  0267 MSG.LEN     ?E100C0 MSGPOINTER  ?  07F8 MSLOT
?E1012D MVAL          0361 MVN.NVR        FF7CEF N.NXT.CHK       7F73 NEWADDRERR     C029 NEWVIDEO      FF7A61 NEXT.SW
?  03FB NMI           0325 NO.TESTS        767D NOT.FAIL          FF NOT.SW       FF7B0C NOW.SLOW     M    28 NTV16
M    00 NTV           0300 NVR.BEG         0315 NVR.END          0361 NVR.FREE     FF7DE1 NXT.BAT.ADR  FF7DF2 NXT.BAT.PAT
  7462 NXT.BI.CYCLE FF7D1A NXT.CLK.BIT  FF7E37 NXT.CMP.WORK FF7522 NXT.FAIL.CHA    7ED6 NXT.FDB.BYTE FF7DFB NXT.FILL.WOR
  760B NXT.KICK     FF75E5 NXT.PASS.CHA    76A1 NXT.STAT         74B5 NXT.TST      FF7731 NXT.ZBYTE    ?E10135 NXTCUR
?  07FB OLDBASH     ?  077B OLDBASL     ?  047B OLDCH       ?E10139 ORGMEMORY   ?E10341 OSVARIABLES ?E100A8 OSVECTORS
?  03D0 OSWMSTRT    ?  057B OURCH       ?  05FB OURCV          780E OUTER16     ?E100D3 OUTMASK     ?  C064 PADDL0
  C058 PASS.LED       C0E3 PASS.LEDD       75CF PASS.SELF.ST     7600 PASS.SELF.TO?    3B PCH         ?    3A PCL
  FFFF PEA.BANK       0B00 POINTER         0329 PORT.ON       FF7752 PORT.PARMS      C05A POWER.OFF      C0E5 POWER.OFFD
  C05B POWER.ON       C0E4 POWER.OND       7695 PR.NOADD         7699 PRINT.STATUS?    06 PRINTTMP        768D PRNYBBLE
?FF5A00 PRODOSINIT  ?    33 PROMPT      ?  C070 PTRIG       ?E1012A PVAL          03F4 PWREDUP        7BA4 QTRIRQ
?E1012E QVAL          7F15 R.FDB        FF7F54 RAM.ADDR          F0 RAMBANK         F4 RAMBEGADDR       F2 RAMENDADDR
  77CC RAMERROR2      77C7 RAMERROR        76B0 RD.BI.NVR.BB?FF7E2D RD.BRAM.WORK   C018 RDB0COL        C01F RDB0VID
  C016 RDALTZP     ?E1006C RDBUSYFLG       C017 RDC3ROM          C003 RDCARDRAM    C015 RDCXROM        C01D RDHIRES
  C011 RDLCBNK2       C012 RDLCRAM         C002 RDMAINRAM        C01B RDMIX        C01C RDPAGE2        C013 RDRAMRD
  C014 RDRAMWRT       C01A RDTEXT      ?  C019 RDVBLBAR      FF7E61 READ.NVR     FF7E69 READBRLP        0B0D REGNO
  795F REGRW          7A3D RESETSCC        0B02 RESULT           7A23 RETRN     ?    2D RMNEM       ?    4F RNDH
?    4E RNDL        FF7CAE ROM.CHECKSUM    7656 ROM.COUT88       7635 ROM.HOME.86    764F ROM.NXT.SP     7684 ROM.PRBYTE88
  C028 ROMBANK        C081 ROMIN             FF RUNBANK          7955 RWCHA        C032 SCANINT     ?E10105 SCC1
?E10106 SCC2       ?E10107 SCC3        ?  C03B SCCADATA    ?  C039 SCCAREG    ?  C03A SCCBDATA    ?  C038 SCCBREG
  BFFD SCCCMD         BFFF SCCDATA         790A SCCEND        ?E10103 SCCFLAG2  ?E10102 SCCFLAG        790B SCCREGRW
  7A26 SCCREGSET   ?E10101 SCCSTAT        FFFF SELF.MASK     FF75F0 SELF.PASS.MS   749A SELF.TST       C061 SELF
  79C6 SENDDATA         00 SEQ.DEBUG       7902 SER.TST          2000 SER.ZBANK    ?E10104 SERFLAG     ?  0B09 SERIRQCNT
?  0B10 SERIRQTEMP1 ?  0B11 SERIRQTEMP2 ?  0B12 SERIRQTEMP3    0B0E SERMASK         C001 SETB0COL       C00D SETB0VID
  C00F SETALTCHAR     C009 SETALTZP        C05B SETAN0           C05A SETAN1    ?  C05C SETAN2      ?  C05E SETAN3
  7A28 SETENV2        C00A SETINTC3ROM     C007 SETINTCXROM      C00B SETSLOTC3ROM   C006 SETSLOTCXROM   C008 SETSTDZP
  C035 SHADOW         7D5F SHADOW.TST      5A5A SHADPATT1        A5A5 SHADPATT2      0B00 SHDDATA        7DD1 SHDFAIL
  7DC1 SHDFETCH       0400 SHDLENGHT       7DB4 SHDSTORE           0A SHDSTRT        7DBE SHDVERFY       7DB1 SHDWRITE
?E100CF SIOFLAG       7832 SKIP16          C100 SLOT.ADDR          01 SLOT        ?E10138 SLOTSFLG       0C0D SLOW.LIM
M  0210 SLOW          C02D SLTROMSEL    ?FF7AF9 SOFT.END       FF7A5A SOFT.SW       FF0029 SOFT.TAB.SIZ FF7ACD SOFT.TAB
?  03F2 SOFTEV     ?  C03F SOUNDADRH    ?  C03E SOUNDADRL       C03C SOUNDCTL    ?  C03D SOUNDDATA      C030 SPKR
?    49 SPNT       ?E10FD9 SRQCNT       ?E10FDB SRQMRQ       ?E10FD8 SRQNUM       ?E10FDA SRQPTR         031D ST.FAIL
  741B ST.SEQ.B   ?FF7AF6 STATE.0      ?FF7AF4 STATE.1      ?FF7AF1 STATE.2      ?FF7AEE STATE.3     ?FF7AEB STATE.4
?FF7AE8 STATE.5   ?FF7AE5 STATE.6      ?FF7AE2 STATE.7       FF7A98 STATE.INV      C068 STATEREG     ?E10FD7 STATUSTMP
?    48 STATUS    ?  0804 STCNT1       ?  0805 STCNT2     ?  0806 STCNT3     ?  0807 STCNT4         7815 STEPP16
 FF7709 STD.NXT.PARM  781D STUFF16      ?E10126 SVAL          E1001C SYSDMGRV   ?E10137 SYSTEMSPD  ?  C022 TBCOLOR
 FF7A4A TBL2      ?  077B TEMP1           7565 TEST.PASS        7843 TEST16     ?  C06D TESTREG     ?E11E00 THBUF
    07 THEDATA    ?E100DC TICKCNT      ?E103E9 TIMEBUFFER    ?E103E0 TIMEOUT   ?E10094 TOBRAMSETUP ?E10090 TOCTRL.PANEL
  7607 TONE86     ?E1009C TOPRINTMSG16?E10098 TOPRINTMSG8    E10084 TOREADBR     E1008C TOREADTIME    E10080 TOWRITEBR
 E10088 TOWRITETIME ?E10144 TRKEY          0323 TST.COUNT       FF76EB TST.DEATH.MG   74E9 TST.FAIL       031F TST.MASK
    FB TST.PTR        75B1 TST.RTS         0315 TST.STATUS      FF7402 TST.TAB        FF7419 TST.TAB.E      79A9 TSTCH
  7BC7 TST0TRIRQ      794E TSTRW           7C5C TSTSECIRQ    ?E10143 TWKEY           7A0D TXCHK          79B2 TXCLR
  C050 TXTCLR         C054 TXTPAGE1        C055 TXTPAGE2        C051 TXTSET       ?E10145 TYPHED     ?E10008 UDISPATCH1
?E1000C UDISPATCH2     01 USE.FDB.TOOL?E102F5 USERAMPM      ?E102DC USERBACK    ?E102DF USERBELL    ?E102DD USERBORDER
?E102F0 USERCAPS  ?E102D9 USERCOL      ?E102D8 USERCOLOR     ?E102F4 USERDATE    ?E102E9 USERDISP    ?E102F2 USERDUAL
?E102F1 USERFSPDE ?E102DA USERHERTZ    ?E102F3 USERHMR       ?E102EB USERKBUFF   ?E102EA USERLANG    ?E102F6 USERMRAM
?E102C0 USERPORT1 ?E102CC USERPORT2    ?E102ED USERRDLY      ?E102EC USERRSPD    ?E102E1 USERSLT1    ?E102E2 USERSLT2
?E102E3 USERSLT3  ?E102E7 USERSLT7     ?E102E8 USERSLTST     ?E102E0 USERSPEED   ?E102DB USERTEXT    ?E102DE USERVOL
?E102F7 USERXRAM  ?  03FB USRADR     ?    2D V2               0310 VALID.CHK      FF75FD VALID.MASK      7B99 VBLIRQTST
  C02E VERTCNT        00 VGC.SWITCH    FF7C8F VGCINTHND      C023 VGCINT         7C8D VGCPASS          7C68 VGCRSTRE
  7BFE VGCTSTIRQ    7769 WJ              7F5B WLP1        ?    23 WNDBTM     ?    20 WNDLFT      ?    22 WNDTOP
?    21 WNDWDTH       76C5 WR.BI.NVR.66 FF7E25 WR.RD.CMP    FF7D4C WR.RD.CMP.TI   C005 WRCARDRAM        12 WRITE.JMP
 FF7E4C WRITE.NVR     C10F WRITE.OFF       7F77 WRITEADDR        7FDF WRITEIT      7FD1 WRITELP1       C004 WRMAINRAM
?    10 X8        ?  06FB XCOORD       ?E10131 XPVAL      ?    46 XREG             0A XTEMP       FF7E8D XTNDBRAM4
```

```
7FF7E99 XTNDBRAM6   FF7E7D XTNDBRAM   ?   47 YREG      ?   35 YSAV1     ?   34 YSAV      ?   BE ZER1
?   E2 ZER2           775E ZERO.INIT    7765 ZERO.WRITE   78B9 ZERRIND      78AC ZERROR       F9 ZP.PNTR
  78D2 ZPIND1         78DA ZPIND2       78EB ZPIND4       7876 ZRAM1        787F ZRAM2      7892 ZRAM4
```

```
M   00 NTV              00 VGC.SWITCH ?   00 LOC0       ?FE0000 INITDISMGR      00 SEQ.DEBUG  ?E10000 DISPATCH1
    00 BAUD.SWITCH      00 CHECKSUM   ?   01 M.MOUSE          01 USE.FDB.TOOL?  01 LOC1           01 SLOT
?   02 M.1          ?   02 M.PAS1.0       03 FILL2IOMSK ?   04 M.VMODE    ?   04 M.2        ?E10004 DISPATCH2
    04 FILL2MONVAR      04 LOC        ?   06 GOODF8     ?   06 F8VERSION  ?   06 PRINTTMP        07 THEDATA
?E10008 UDISPATCH1  ?   08 M.CTL      ?   08 M.GOXY         08 FILL2MSGPTR    09 MOD            0A XTEMP
    0A SHDSTRT          0B FLIP       ?E1000C IRQLEN    ?E1000C UDISPATCH2      10 INIT.JMP       10 BAT.RAM.PTR
?   10 X8           ?   10 M.4        ?   10 M.CURSOR    E10010 INTMGRV         10 FILL2MVCTRS    12 WRITE.JMP
?E10014 COPMGRV     ?E10018 ABORTMGRV  E1001C SYSDMGRV    E10020 IRQ.APTLK  ?   20 M.CTL2     ?   20 WNDLFT
?   20 M8           ?   21 WNDWDTH    ?   22 WNDTOP     ?   23 WNDBTM     ?   24 CH         ?E10024 IRQ.SERIAL
?   25 CV           ?   26 GBASL      ?   27 GBASH      ?E10028 IRQ.SCAN  M   28 NTV16      ?   28 BASL
 FF0029 SOFT.TAB.SIZ?   29 BASH       ?   2A BAS2L      ?   2B BAS2H      ?   2C H2         ?   2C LMNEM
?E1002C IRQ.SOUND   ?   2D RMNEM      ?   2D V2         ?   2E FORMAT     ?   2E MASK       ?   2F LENGTH
?   30 COLOR        ?E10030 IRQ.VBL   ?   31 MONMODE    ?   32 INVFLG     ?   33 PROMPT     ?E10034 IRQ.MOUSE
?   34 YSAV         ?   35 YSAV1      ?   36 CSWL       ?   37 CSWH       ?E10038 IRQ.QTR       38 EXMEMCMD
?   38 KSWL         ?   39 KSWH       ?   3A PCL            3B CHBINDEX   ?   3B PCH            3C CHAINDEX
?   3C A1L          ?E1003C IRQ.KBD   ?   3D A1H      ?   3E A2L        ?   3F A2H        ?   40 A3L
?E10040 IRQ.RESPONSE?   40 M.6        ?   41 A3H      ?   42 A4L        ?   43 A4H        ?E10044 IRQ.SRQ
?   45 A5H          ?   46 XREG       ?   47 YREG       ?   48 STATUS     ?E10048 IRQ.DSKACC  ?   49 SPNT
?E1004C IRQ.FLUSH   ?   4E RNDL       ?   4F RNDH       ?E10050 IRQ.MICRO ?E10054 IRQ.1SEC  ?E10058 IRQ.EXT
?E1005C IRQ.OTHER   ?E10060 CUPDATE   ?E10060 IRQ.LEN   ?E10064 INCBUSYFLG ?E10068 DECBUSYFLG ?E1006C RDBUSYFLG
?   77 EML          ?   80 M.PASCAL    E10080 TOWRITEBR   E10084 TOREADBR    E10088 TOWRITETIME E1008C TOREADTIME
?E10090 TOCTRL.PANEL?E10094 TOBRAMSETUP ?E10098 TOPRINTMSGB ?E1009C TOPRINTMSG16?E100A8 OSVECTORS  M   B5 DATA16
?   BE ZER1         ?E100C0 MSGPOINTER ?E100C3 IRQ.INTFLAG ?E100C4 IRQ.DATAREG ?E100C5 IRQ.KMSTATUS?E100C6 IRQ.SECFLAG
?E100C7 IRQ.SERIAL1 ?E100C8 IRQ.SERIAL2 ?E100C9 IRQ.APTLKHI ?E100CA IRQ.VOLUME  ?E100CB IRQ.ACTIVE  ?E100CF SIOFLAG
?E100D0 INMASK      ?E100D3 OUTMASK    ?E100D6 ENSONIQFLG ?E100D7 IRQ.MOUSVBL ?E100D8 HBPTR       ?E100DC TICKCNT
?E100E0 MOUSESLOT   ?E100E1 EVMGRDATA  ?   E2 ZER2            F0 RAMBANK        F2 RAMENDADDR     F4 RAMBEGADDR
M   F5 DATA8            F9 ZP.PNTR        FB TST.PTR        FD CHK.BEG    ?   FF LIMIT          FF NOT.SW
    FF RUNBANK      ?E100FF BUSYFLAG  ?010100 MNEMSTKPTR ?E10101 SCCSTAT   ?010101 ALEMSTKPTR ?E10102 SCCFLAG
?E10103 SCCFLAG2    ?E10104 SERFLAG   ?E10105 SCC1     ?E10106 SCC2      ?E10107 SCC3      ?E10108 ASCIITIME
 E10108 IRQ.A       ?E1010A IRQ.X     ?E1010C IRQ.Y     ?E1010E IRQ.S     ?E10110 IRQ.D     ?E10112 IRQ.P
?E10113 IRQ.DB      ?E10114 IRQ.E     ?E10115 IRQ.K     ?E10116 IRQ.PC    ?E10118 IRQ.STATE ?E10119 IRQ.SHADOW
?E1011B IRQ.MSLOT   ?E10120 AVAL      ?E10126 SVAL      ?E10128 DVAL      ?E1012A PVAL      ?E1012B BVAL
?E1012C KVAL        ?E1012D MVAL      ?E1012E QVAL      ?E1012F LVAL      ?E10130 MPVAL     ?E10131 XPVAL
?E10132 EPVAL       M  0133 IND16     ?E10133 ASCFILTER ?E10134 CURSOR    ?E10135 NXTCUR    ?E10136 CHGCURFLG
?E10137 SYSTEMSPD   ?E10138 SLOTSFLG  ?E10139 ORGMEMORY ?E1013A E1PCLH    ?E1013C GOS       ?E1013E BANKS
?E1013F BANKD       ?E10140 EXTENDBNK ?E10142 DIGITCNT  ?E10143 TWKEY     ?E10144 TRKEY     ?E10145 TYPHED
?E10146 FIXST1      ?E10148 FIXST2    ?E1014A FIXST3    M  0169 MEM16     M  019F IND8      M  01D6 MEM8
?  0200 IN          M  0210 SLOW      M  0236 FAST      ?  0267 MSG.LEN   ?E102BB LOWXCLAMP  ?E102BA LOWYCLAMP
?E102BC HIXCLAMP    ?E102BE HIYCLAMP   E102C0 BRAMBUFFER ?E102C0 USERPORT1 ?E102CC USERPORT2 ?E102D8 USERCOLOR
?E102D9 USERCOL     ?E102DA USERHERTZ ?E102DB USERTEXT  ?E102DC USERBACK  ?E102DD USERBORDER ?E102DE USERVOL
?E102DF USERBELL    ?E102E0 USERSPEED ?E102E1 USERSLT1  ?E102E2 USERSLT2  ?E102E3 USERSLT3   ?E102E7 USERSLT7
?E102E8 USERSLTST   ?E102E9 USERDISP  ?E102EA USERLANG  ?E102EB USERKBUFF ?E102EC USERRSPD   ?E102ED USERRDLY
?E102F0 USERCAPS    ?E102F1 USERFSPDE ?E102F2 USERDUAL  ?E102F3 USERHMR   ?E102F4 USERDATE   ?E102F5 USERAMPM
?E102F6 USERMRAM    ?E102F7 USERXRAM  ?E102FB LANGS       0300 NVR.BEG      0300 BI.STATUS  ?E10301 LAYOUTS
   0306 BI.COUNTER     0308 BI.PASSES     030A BI.FAILS     030C BI.LASTF     030E CYCLES.MAX    0310 VALID.CHK
   0313 BI.ALT.MASK    0315 TST.STATUS    0315 NVR.END      0316 ERRCODE1     0317 ERRCODE2      031B BI.FAIL
   031D ST.FAIL        031F TST.MASK      0321 CYCLE.COUNT  0323 TST.COUNT    0325 NO.TESTS      0327 CUR
   0329 PORT.ON     ?E10340 APTALKNODE ?E10341 OSVARIABLES  0361 NVR.FREE     0361 MVN.NVR    ?E103BC BRAMCHKSUM1
?E103BE BRAMCHKSUM2 ?  03D0 OSWMSTRT  ?E103E0 TIMEOUT    E103E1 CLKRDATA   E103E5 CLKWDATA   ?E103E9 TIMEBUFFER
?  03F0 BRKV        ?  03F2 SOFTEV        03F4 PWREDUP   ?  03F8 USRADR    ?  03FB NMI       ?  03FE IRQLOC
   0400 LINE1          0400 SHDLENGHT ?  047B OLDCH     ?  04FB MODE      ?  057B OURCH     ?  05FB OURCV
?  067B CHAR        ?  06FB XCOORD    ?  077B OLDBASL   ?  077B TEMP1     ?  07F8 MSLOT     ?  07FB OLDBASH
   0800 BRAM.TEMP      0800 SHDDATA       0800 CNT1         0801 CNT2         0802 CNT3      ?  0803 CNT4
?  0804 STCNT1      ?  0805 STCNT2    ?  0806 STCNT3    ?  0807 STCNT4    ?  0809 SERIRQCNT ?  080A CHHSK
?  080B CHDTR          080C DATA          080D REGNO        080E SERMASK      080F MASKDATA  ?  0810 SERIRQTEMP1
?  0811 SERIRQTEMP2 ?  0812 SERIRQTEMP3   0900 BRAM.WORK    0A00 BRAM.WORK1   0B00 POINTER      0B02 RESULT
   0B04 FDB.BUF        0C0D SLOW.LIM     E10FD6 FLAGS     ?E10FD7 STATUSTMP ?E10FD8 SRQNUM  ?E10FD9 SRQCNT
 ?E10FDA SRQPTR     ?E10FDB SRQMRQ        1400 FDB.ROM.B    191A FAST.LIM   ?E11E00 THBUF       1FFF FDB.ROM.E
```

```
  2000 SER.ZBANK      2672 FDB.REV3    ? 3DE6 FDB.REV1    ?FF5A00 PROCONINIT     5A5A SHADPATT1      7000 CUSIRQCNT
  7002 CUSIRQSTS      7004 CUSLINENO      7006 CUSIRQSRCE     7008 IRQTEMP1       700A IRQTEMP2       700C IRQTEMP3
?FF7400 DIAGS       FF7402 TST.TAB      FF7419 TST.TAB.E    741B ST.SEQ.B       7462 NXT.BI.CYCLE   747E BAD.NVR
?  7489 BI          7490 BI.SELF.TST    749A SELF.TST       749F EXT.SEQ        74B5 NXT.TST        74E9 TST.FAIL
  7517 FAIL.SELF    FF7522 NXT.FAIL.CHA FF752D FAIL.SELF.TO FF7558 FAIL.SELF.MS 7565 FAIL.BI.CONT  7565 TEST.PASS
  757A END.TST.SEQ    758F CONT.CYC       759A FIN.CYCLES     75A4 BI.CYCLE.FAI   75B1 TST.RTS      FF75B2 DO.TEST
  75CF PASS.SELF.ST FF75E5 NXT.PASS.CHA FF75F0 SELF.PASS.MS FF75FD VALID.MASK     7600 PASS.SELF.TO   7607 TONE86
  760B NXT.KICK    ? 760D KG.WAIT        760E KG.WAIT2       760F KG.WAIT3       7625 END.SELF       7630 CHKOAONLY
  7635 ROM.HOME.86    764F ROM.NXT.SP     7656 ROM.COUT88     767D NOT.FAIL       7684 ROM.PRBYTE88   768D PRNYBBLE
  7695 PR.NOADD       7699 PRINT.STATUS   76A1 NXT.STAT       76B0 RD.BI.NVR.88   76C5 WR.BI.NVR.66   76DC BI.SLF.NVR
  76DF BANK.0         76E5 BANK.RUN     FF76EB TST.DEATH.MG FF7702 INIT.PORT1   FF7709 STO.NXT.PARM FF7731 NXT.ZBYTE
?FF773F DO.INIT     FF7749 COUT.PORT1 ?FF774D DO.WRITE      FF7752 PORT.PARMS     775E ZERO.INIT    FF775E END.PORT.PAR
 FF775E BANK.ZERO.BE  7762 IJ            7765 ZERO.WRITE     7769 WJ             776C MOVIRAM        776C BANK.ZERO.EN
  77C2 MOVIRAMEXIT    77C7 RAMERROR       77CC RAMERROR2      77CD MOVIRAMTEST    77FE FOUNDMOVIERR   77FF DOMOVITEST
  780E OUTER16        7815 STEPP16        781D STUFF16        782A ALT16          7832 SKIP16         7843 TEST16
  7847 INV16          7851 JMP16          7864 ERROR16        786B DOZPAGE00      7876 ZRAM1          787F ZRAM2
  7892 ZRAM4          78AC ZERROR         78B9 ZERRIND        78C7 DOZPAGEIND     78D2 ZPIND1         78DA ZPIND2
  78EB ZPIND4         7902 SER.TST        790A SCCEND         790B SCCREGRW       7947 FAILRW         794E TSTRW
  7955 RWCHA          795E ENDRWTST       795F REGRW          7987 BADDATA        798A INTLOOP        79A4 INTLOOPERR
  79A9 TSTCH          79AC CHN1           79B2 TXCLR          79C6 SENDDATA       79CF ALLSENT        79E8 CHKREAD
  79EE CHKRD          7A02 CHKDATA        7A0D TXCHK          7A1C AGAIN          7A23 RETRN          7A26 SCCREGSET
  7A28 SETENV2        7A3D RESETSCC     FF7A4A TBL2         FF7A5A SOFT.SW      FF7A61 NEXT.SW      FF7A98 STATE.INV
 FF7A9B CLR.SW      ?FF7AB3 INC.SW      FF7AB4 INC.SW.1     FF7AC0 MEGA.SOFT.ER FF7ACD SOFT.TAB    ?FF7AE2 STATE.7
?FF7AE5 STATE.6     ?FF7AE8 STATE.5   ?FF7AEB STATE.4     ?FF7AEE STATE.3     ?FF7AF1 STATE.2     ?FF7AF4 STATE.1
?FF7AF6 STATE.0     ?FF7AF9 SOFT.END      7AF9 FPI.SPEED  ?FF7AF9 LBL00      FF7B0C NOW.SLOW     FF7B1C GOOD.SPEED
 FF7B1E BAD.SPEED   FF7B26 FIND.SPEED   FF7B2C LBL01      FF7B36 LBL02        7B3E CUSTOM.IRQ     7B46 CUS.IRQ.RTL
  7B47 MEGATSTIRQ     7B84 LOOP001        7B99 VBLIRQTST      7BA4 QTRIRQ         7BB2 LOOP002        7BC7 TSTQTRIRQ
  7BD0 MEGAERR        7BD3 MEGRSTRE       7BEF MEGAPASS     FF7BF1 MEGAINTHND     7BFE VGCTSTIRQ      7C42 LOOP003
  7C5C TSTSECIRQ      7C65 ERROR2         7C68 VGCRSTRE       7C8D VGCPASS      FF7C8F VGCINTHND      7CA7 INCCNT
 FF7CAE ROM.CHECKSUM FF7CD0 GOOD.ROM   ?FF7CD2 BAD.ROM      FF7CDF BAD.RAM      FF7CEA CAL.CHECKSUM FF7CEE CAL.CHECKSUM
 FF7CEF N.NXT.CHK    FF7CF7 CHK.END      FF7CFA CLOCK        FF7D1A NXT.CLK.BIT  FF7D24 CLK.CONT     FF7D4A CLK.ERR
 FF7D4C WR.RD.CMP.TI   7D5F SHADOW.TST     7DAB C035.BAD       7DB1 SHDWRITE       7DB4 SHDSTORE       7DBE SHDVERFY
  7DC1 SHDFETCH       7DD1 SHDFAIL      FF7DD3 BAT.RAM      FF7DE1 NXT.BAT.ADR  FF7DF2 NXT.BAT.PAT ?FF7DF9 FILL.BRAM.WD
 FF7DFB NXT.FILL.WOR FF7E0D BAT.RAM.ER1  FF7E10 BAT.RAM.ER2  FF7E1B GOOD.BAT.RAM FF7E25 WR.RD.CMP   ?FF7E2D RD.BRAM.WORK
?FF7E35 CMP.BRAM.WOR FF7E37 NXT.CMP.WORK FF7E44 BRAM.CMP.ER  FF7E46 BAT.PAT      FF7E4C WRITE.NVR    FF7E61 READ.NVR
 FF7E69 READBRLP     FF7E7D XTNDBRAM     FF7E8D XTNDBRAM4  ?FF7E99 XTNDBRAM6   FF7EAC ACCCLKCLV    FF7EAD ACCCLOCK
 FF7EBB ACCCLOCK0    FF7EBD ACCCLOCK1    FF7EC0 ACCCLOCK2    FF7EC9 FDB          7ED6 NXT.FDB.BYTE? 7EEE FDB.CHECK.TS
  7F08 FDB.GOOD.CHK   7F0A FDB.TOOL.ER    7F15 R.FDB        FF7F4B FDB.TOOL.RTN? 7F54 DONEWADDR    FF7F54 RAM.ADDR
  7F5B WLP1           7F67 CLP1           7F73 NEWADDRERR     7F77 WRITEADDR      7FA2 CMPADDR        7FCA CMPFAIL
  7FCF WRITEIT        7FD1 WRITELP1       7FE0 CMPIT          7FE2 CMPLP1         7FF4 CMPERROR     FF7FFC BANKTBL
 FF802C FDB.TOOL.ADD  A5A5 SHADPATT2    ? AAAA CHKSMCNST      BFFD SCCCMD         BFFF SCCDATA        C000 IOADR
  C000 CLR80COL    ? C000 KBD             C001 SET80COL       C002 RDMAINRAM      C003 RDCARDRAM      C004 WRMAINRAM
  C005 WRCARDRAM      C006 SETSLOTCXROM   C007 SETINTCXROM    C008 SETSTDZP       C009 SETALTZP       C00A SETINTC3ROM
  C00B SETSLOTC3ROM   C00C CLR80VID       C00D SET80VID       C00E CLRALTCHAR     C00F SETALTCHAR  ? C010 KBDSTRB
  C011 RDLCBNK2       C012 RDLCRAM        C013 RDRAMRD        C014 RDRAMWRT       C015 RDCXROM        C016 RDALTZP
  C017 RDC3ROM        C018 RD80COL     ? C019 RDVBLBAR        C01A RDTEXT         C01B RDMIX          C01C RDPAGE2
  C01D RDHIRES        C01E ALTCHARSET     C01F RD80VID        C021 MONOCOLOR   ? C022 TBCOLOR        C023 VGCINT
?  C024 MOUSEDATA   ? C025 KEYMODREG    ? C026 DATAREG     ? C027 KMSTATUS        C028 ROMBANK        C029 NEWVIDEO
?  C02B LANGSEL     ? C02C CHARROM         C02D SLTROMSEL      C02E VERTCNT     ? C02F HORIZCNT       C030 SPKR
?  C031 DISKREG        C032 SCANINT        C033 CLOCKDATA      C034 CLOCKCTL       C035 SHADOW         C036 CYAREG
?  C037 DMAREG      ? C038 SCCBREG     ? C039 SCCAREG     ? C03A SCCBDATA    ? C03B SCCADATA       C03C SOUNDCTL
?  C03D SOUNDDATA   ? C03E SOUNDADRL   ? C03F SOUNDADRH      C041 INTEN       ? C044 MMDELTAX    ? C045 MMDELTAY
   C046 INTFLAG        C046 DIAGTYPE       C047 CLRVBLINT  ? C048 CLRXYINT       C050 TXTCLR         C051 TXTSET
   C052 MIXCLR         C053 MIXSET         C054 TXTPAGE1       C055 TXTPAGE2       C056 LORES          C057 HIRES
   C058 SETAN0         C058 PASS.LED       C059 FAIL.LED       C059 CLRAN0         C05A SETAN1         C05A POWER.OFF
   C05B CLRAN1         C05B POWER.ON    ? C05C SETAN2      ? C05D CLRAN2    ? C05E SETAN3      ? C05F CLRAN3
 ?  C060 BUTN3         C061 BUTN0          C061 SELF           C062 BUTN1       ? C063 BI.FLAG        C063 BUTN2
 ?  C064 PADDL0        C068 STATEREG    ? C06D TESTREG     ? C06E CLRTM     ? C06F ENTM        ? C070 PTRIG
```

```
  C081 ROMIN          C083 LCBANK2        C08B LCBANK1        C0E2 FAIL.LEDD      C0E3 PASS.LEDD      C0E4 POWER.ONO
  C0E5 POWER.OFFD     C100 SLOT.ADDR      C10D INIT.OFF       C10F WRITE.OFF      CFFF CLRROM       ? E000 BASIC
? E003 BASIC2         F772 FDB.REV2       FF3F BI.SELF.MASK   FFFF PEA.BANK       FFFF BI.MASK        FFFF SELF.MASK
** SUCCESSFUL ASSEMBLY := NO ERRORS
** ASSEMBLER CREATED ON 07-JAN-86 20:27
** TOTAL LINES ASSEMBLED  4563
** FREE SPACE PAGE COUNT  140
```