```
;*************************************************
;* Lisa 800 -- 800K Drive Patch for MacWorks  3.0 *
;*************************************************
;*
;*        Author:  C. Lukaszewski
;*       Updated:  02/02/88  - Ian H. Abel
;*       Version:  1.0
;*
;*************************************************
;*  Copyright  (C) 1988 / Ian H. Abel  *
;*************************************************

        INCLUDE         Traps.D             ;Include  system & toolbox  traps
        INCLUDE         ToolEqu.D           ;Include  toolbox  equates
        INCLUDE         SysEqu.D            ;Include  system  equates
        INCLUDE         FSEqu.D             ;Include  file system  equates
        INCLUDE         PackMacs.Txt        ;Include  package  macros
        INCLUDE         ISDEqu.Txt          ;Include  PSM file  equates
        STRING_FORMAT   3                   ;Length  precedes  string

        XDEF            Lisa800,Junk        ;Finder  entry  point
        XDEF            RefNum0,RefNum1     ;PSMFile  required  globals
        XREF            InitMgrs,LineOut    ;PSM  routines
        XREF            OpenWndw,OpenDlog
        XREF            NormText

MWBASE  EQU             $1E693A             ;Base  of patch  w/o Lisabug

;* This  code  makes  patches  to MacWorks  v3.0  to allow  the use  of an 800K  drive
;* with  Sun  Remarketing's  HFS  on the Lisa-2/Mac  XL.  The code  is set up  as an
;* executable  program  which  directly  modifies  a 5- or 10-megabyte  widget.  The
;* MacWorks  operating  system  is really  an extensively  modified  Apple  'monitor'
;* operating  system,  whose  filesystem  structure  can be figured  out  from  my
;* notes  or  from  not-too-extensive   hacking.   This patch  makes  modifications  to
;* the  file  'DRIVERS.OBJ'  on  the  MacWorks  volume.   It first  copies  the  files  to
;* a  location  farther  out  so that  there  is  space  to  append  code  to  it.  Then
;* the  patch  is  appended  and  the  monitor  directory  is  updated.   The  patch  is
;* executed  at startup  time  only,  assuming  that  there  is a  $02  value  at  $FCC015,
;* which  indicates  that  there  is  a  two-sided  drive  attached.
;*
;* The  patches  are  as  follows:
;*---------------------------------------------------------------------------*
;*      00xCFF16   6708  ->  6008             The  6504  code  returns  a  '$22'  instead
;*                                            of  a  '$02'  at  $00FCC013  (the  drive
;*                                            type  field).   This  removes  the  error
;*                                            branch  which  sets  up  error  $FFB2  (-78)
;*                                            try  to  read  second  side  on  a  400K  drive
;*---------------------------------------------------------------------------*
;*      00000CFB   xx  -->  FF                Update  drive  queue  to  reflect  two-sided
;*                                            disk  drive
;*---------------------------------------------------------------------------*
;*      MWBASE+HFSCalc                        Routine  for  calculating  side/track/sector
;*                                            for  HFS  volumes:
;*                                              * Routine  must  verify  that  $FCC015  =
;*                                                $02  and  $03BC=$FFFB  otherwise  execute:
;*                                                           MOVE.W    #$0640,D1
;*                                                           LSR.W     #$1,D1
;*                                                           JMP       $00xCFFDE
;*                                              * If  two-side  check  is  successful,  calc-
;*                                                ulate  side/track/sector   with  A0  &  A1
;*                                                preserved  and  absolute  sector  number
;*                                                in  D0.   Patch  routine  ends  with:
;*                                                           D5  =  Side  (0  or  1)
;*                                                           D4  =  Track
;*                                                           D0  =  Sector
;*                                                and  exits  with  a  copy  of  the  code  at
;*                                                   $00xD003E.
;*------------------------------------------------------------------------------*
;*        001CFFD8    323C0640E249    ->
;*                                JMP  HFSCalc         Intercept  normal  calculation  routine
;*------------------------------------------------------------------------------*
;*        MWBASE+InitPtch1                             Patch  that  ignores  control  calls  of value
;*                                                     $15  to  the  Sony  driver  &  forces  a nonzero
;*                                                     return  value  in  D0.
;*------------------------------------------------------------------------------*
;*        00401826    ->          JMP  InitPtch1       Intercept  normal  Control  codeflow
;*------------------------------------------------------------------------------*
;*        MWBASE+InitPtch2                             Patch  that  adjusts  internal  HFS flag
;*                                                     and  drive  queue  volume  size  according
;*                                                     to  the  user's  selection  of single-  or
;*                                                     double-sided  format.   For  single-sided
;*                                                     formats,  the  #sides  byte  at  $FCC015  is  set
;*                                                     to  1  to  get  normal  400K  format  time.
;*------------------------------------------------------------------------------*
;*        00401826    ->          JMP  InitPtch2       Intercept  normal  PACK2  control  flow
;*------------------------------------------------------------------------------*
;*        MWBASE+SidePtch                              Routine  to modify  drive  queue  entry  for
;*                                                     floppy  drive  when  HFS  disk  is  inserted
;*                                                     (byte  just  before  the  queue  entry  should
;*                                                     be  -1  for  HFS,  0  for  MFS)
;*------------------------------------------------------------------------------*
;*        001BF438    CMPI.W  #$4244,8(A2)->
;*                                JMP  SidePtch        Intercept  normal  comparison  routine
;*------------------------------------------------------------------------------*
;*        0000039E  ->  ptr  to  HFSDefaults           Set  up  HFS  Defaults  as  described  in  IM4
;*                                                     for  formatting
;*------------------------------------------------------------------------------*

;***   InitWndw  --  Draw  Banner  Window
;*
;*     Entry         None
;*     Exit          Banner  Window  Drawn  &  Handle  in  TempA(A5)
;*     Uses          A0,A1,A2
;*     Calls         OpenWndw,LineOut
;*     Macros        None
;*
;***

InitWndw  LEA        LogoItms,A0                       ;Store  item  handle
          LEA        LItmHndl,A1
          MOVE.L     A0,(A1)
          LEA        LogoWndw,A2                       ;Get  Dialog  Info.  Block
          SUBA.L     A1,A1                             ;No  title
          LEA        LogoSize,A0
          JSR        OpenDlog                          ;Open  dialog  box
          MOVE.L     A0,TempA(A5)                      ;Save  handle
          MOVE.W     #3,-(SP)                          ;Frame  Begin  Box
          MOVE.W     #3,-(SP)
          _PenSize
          PEA        BeginSiz
          MOVE.W     #-4,-(SP)
          MOVE.W     #-4,-(SP)
          _InsetRect
          PEA        BeginSiz
          MOVE.W     #16,-(SP)
          MOVE.W     #16,-(SP)
          _FrameRoundRect
          LEA        TitleString,A2
          BSR        LineOut
          LEA        MeString,A2
          BSR        LineOut
          LEA        VerString,A2
          BSR       LineOut
          LEA       LString1,A2
          BSR       LineOut
          LEA       LString2,A2
          BSR       LineOut
          LEA       LString3,A2
          BSR       LineOut
          LEA       LString4,A2
          BSR       LineOut
          LEA       LString5,A2
          BSR       LineOut
          LEA       LString6,A2
          BSR       LineOut
          LEA       LString7,A2
          BSR       LineOut
          LEA       LString8,A2
          BSR       LineOut
          LEA       LString9,A2
          BSR       LineOut
          LEA       LStringA,A2
          BSR       LineOut
          LEA       CRString,A2
          BSR       LineOut
          LEA       ISWString,A2                  ;Plot strings
          BSR       LineOut
          JSR       NormText
          RTS

DoTxtWdw  MOVE.L    (SP)+,A3
          LEA       TextWndw,A2                   ;Display building window
          SUBA.L    A1,A1
          LEA       TextSize,A0
          JSR       OpenWndw
          MOVE.L    A0,-(SP)                      ;Push handle on stack
          JMP       (A3)                          ;Back to caller

;* Entry Point *
;***************

Lisa800   SUB.L     A2,A2                         ;No resume routine
          JSR       InitMgrs                      ;Setup managers
;*
;* Put a routine here to kill all low-memory vectors
;*
          MOVE.L    RomBase,A0                    ;Check machine type
          CMPI.B    #$FF,9(A0)                    ;See if ROM version is $FF
          BEQ       Lisa1                         ;Yes, branch
          LEA       LisaErr,A2                    ;Do Lisa error
          BRA       Bad
;***Change to BEQ
Lisa1     CMPI.W    #$FFFF,FSFCBLen               ;See if HFS Present
          BNE       Lisa2                         ;Nope, branch
          LEA       HFSErr,A2                     ;Do HFS present error
          BRA       Bad
;***Change to FFFB
Lisa2     MOVE.L    #$3A4,A0
          CLR.L     12(A0)                        ;Find default volume
          _GetVol
          CMP.W     #$FFFE,24(A0)                 ;Must be floppy drive
          BEQ       Lisa4                         ;OK so branch
          LEA       RunErr,A2                     ;Do no run hard disk error
          BRA       Bad
;***Implement this
Lisa3     CLR.L     18(A0)                        ;Try to mount the HD
          MOVE.W    #$FFFE,22(A0)
          _SetVol
```

```
        TST         16(A0)
        BEQ         Lisa4                    ;Branch if OK
        LEA         HDErr,A2                 ;Do no run hard disk error
        BRA         Bad

Lisa4   BSR         InitWndw                 ;And the banner window
Lisa5   CLR.L       -(SP)                    ;No filter proc
        PEA         TempB(A5)                ;Space for result
        _ModalDialog                         ;Handle events
        CMPI        #1,TempB(A5)             ;Get it
        BEQ         Cancel                   ;Skip bad exit
        CMPI        #2,TempB(A5)
        BNE         Lisa5
        BRA         Lisa6                    ;Asked for Install

Bad     MOVE.L      #$100,D0                 ;Request space for Bad
        _NewPtr
        TST.W       D0                       ;Should work, but check it
        BNE         Bad2
        LEA         Bad1,A1                  ;Load address of this routine
        EXG         A0,A1                    ;Exchange them
        MOVE.L      #Cancel-Bad1,D0          ;Move this routine only
        _BlockMove
        MOVE.L      A1,A4
        MOVE.L      ScrnBase,A0              ;Cover Lisa800 with screen RAM
        LEA         InitWndw,A1
        MOVE.L      #HFSCalcEnd-InitWndw,D0
        MOVE.L      A1,D7                    ;Calculate locations of two routines
        LEA         TextWndw,A3
        MOVE.L      A3,D6
        LEA         TextSize,A3
        MOVE.L      A3,D5
        JMP         (A4)                     ;Go away and kill all this
Bad1    _BlockMove
        MOVE.L      D7,A3
        MOVE.L      A3,A4
        ADD.L       #LineOut-InitWndw,A4
        ADD.L       #OpenWndw-InitWndw,A3
        MOVE.L      A2,-(SP)
        MOVE.L      D6,A2                    ;Display building window
        SUBA.L      A1,A1
        MOVE.L      D5,A0
        JSR         (A3)
        MOVE.L      (SP)+,A2
        MOVE.L      A0,-(SP)                 ;Push handle on stack
        MOVE.W      #15,-(SP)                ;Beep
        _SysBeep
        JSR         (A4)
        MOVE.L      #300,A0                  ;Wait 5 seconds
        _Delay
        _DisposWindow
Bad2    MOVE.L      $0002,A0
        JMP         (A0)                     ;Big Bang Boom Crash
Cancel  MOVEQ       #1,D7                    ;Go to finder
        RTS

Lisa6   MOVE.L      TempA(A5),-(SP)          ;Push handle
        _DisposDialog
        MOVE.L      #$64000,D0               ;Load all of Macworks from widget
        _NewPtr                              ;Allocate space to hold it
        TST.W       D0                       ;Check for error
        BEQ         Lisa7
        LEA         MemErr,A2
        BRA         Bad                      ;Do no memory error here
Lisa7   MOVE.L      A0,A4                    ;Put away a copy
        JSR         DoTxtWdw                 ;Make window
        LEA     InstText,A2
        JSR     LineOut
        MOVE    #$3A4,A0
        CLR.L   12(A0)
        MOVE.W  #4,22(A0)
        MOVE.W  #$FFFE,24(A0)
        MOVE.W  #$0009,26(A0)
        CLR.W   28(A0)
        _Control                                ;Set Macworks up to read itself
        MOVE    #$3A4,A0
        MOVE.L  A4,32(A0)                       ;Where to put it?
        MOVE.L  #$64000,36(A0)                  ;Whole thing
        MOVE.W  #1,44(A0)
        MOVE.L  #$1000,46(A0)
        _Read
        MOVE.L  A4,A0                           ;Copy buffer address
        MOVE.L  A4,A1
        ADD.L   #$33600,A0                      ;Location of DRIVERS.OBJ in buffer
        ADD.L   #$33800,A1                      ;Target location + one sector
        MOVE.L  #$28000,D0                      ;Move rest of disk
        _BlockMove
        MOVE.L  A4,A1
        ADD.L   #$31200,A1
        ADD.W   #$300,$32(A1)                   ;Store new length
        LEA     Init800,A0                      ;Location of patch
        ADD.L   #$22FE,A1                       ;Target of patch
        MOVE.L  #InitEnd-Init800+$20,D0         ;Length of move
        CMPM.L  (A0)+,(A1)+                     ;Check already installed
        BNE     Lisa8
        _DisposWindow
        LEA     TwiceErr,A2
        BRA     Bad
Lisa8   SUBQ    #4,A0
        SUBQ    #4,A1
        _BlockMove
        MOVE.L  #14,D0                          ;Modify directory
        LEA     DirPos,A0                       ;Get offsets to positions
Lisa9   MOVE.W  (A0)+,D1
        ADD.W   #1,(A4,D1)                      ;Update them
        DBF     D0,Lisa9
        MOVE.L  ScrnBase,D2                     ;Fetch screen base
        AND.L   #$00700000,D2                   ;Strip off megabyte count
        MOVE.L  A4,A1
        ADD.L   #$34E6A,A1
        MOVE.W  #$6008,(A1)                     ;Install $22 return patch
        MOVE.L  A4,A1
        ADD.L   #$2C3EC,A1
        MOVE.W  #$4EF9,(A1)+                    ;Install patch #6 (GetNextEvent)
        MOVE.L  #$000E6938,A0                   ;Address of Init800
        ADD.L   D2,A0                           ;Add meg count
        MOVE.L  A0,(A1)                         ;Put address
        MOVE    #$3A4,A0                        ;Load up to restore code
        MOVE.L  A4,32(A0)
        MOVE.L  #$64000,36(A0)
        MOVE.W  #1,44(A0)
        MOVE.L  #$1000,46(A0)
        _Write                                  ;Put the whole thing back
        _DisposWindow                           ;Throw away install window
        MOVE.L  A4,A0
        _DisposPtr                              ;Free up our memory
        LEA     DoneText,A2                     ;Do finished window & exit
        BRA     Bad

ErrIO
ErrNSV  _debugger
```

```
;* Init800  *
;***********

Init800  MOVEM.L   D2/A0/A1,-(SP)              ;Store  registers
         MOVE.L    ScrnBase,D2                 ;Fetch  screen  base
         AND.L     #$00700000,D2               ;Strip  off megabyte  count
         MOVE.L    #$000D3738,A0               ;Install  patch  #3 in MW
         MOVE.L    #$000E6ACE,A1               ;Address  of SidePtch
         ADD.L     D2,A0
         ADD.L     D2,A1
         MOVE.W    #$4EF9,(A0)+                ;Put  JMP.L  instruction
         MOVE.L    A1,(A0)                     ;Put  address

         MOVE.L    #$000E42D8,A0
         MOVE.L    #$000E69C6,A1               ;Address  of HFSCalc
         ADD.L     D2,A0
         ADD.L     D2,A1
         MOVE.W    #$4EF9,(A0)+                ;Put  JMP.L  instruction
         MOVE.L    A1,(A0)                     ;Put  address

         MOVE.L    #$00401826,A0
         MOVE.L    #$000E6A52,A1               ;Address  of InitPtch1
         ADD.L     D2,A1
         MOVE.W    #$4EF9,(A0)+                ;Put  JMP.L  instruction
         MOVE.L    A1,(A0)                     ;Put  address

         MOVE.L    #$00401CEE,A0
         MOVE.L    #$000E6A76,A1               ;Address  of InitPtch2
         ADD.L     D2,A1
         MOVE.W    #$4EF9,(A0)+                ;Put  JMP.L  instruction
         MOVE.L    A1,(A0)                     ;Put  address

         MOVE.L    $030A,A0                    ;Flag  drive  as  double
         MOVE.B    #$FF,-1(A0)
         LEA       HFSDefaults,A0              ;Point  FmtDefaults
         MOVE.L    A0,$039E
         MOVE.L    #$0040B3E4,A0               ;Restore  event  routine
         MOVE.L    #$08F80007,(A0)+
         MOVE.W    #$015D,(A0)
         MOVEM.L   (SP)+,D2/A0/A1
         JMP       $0040B3E4                   ;Back  to event  routine

HFSCalc  MOVE.L    A0,-(SP)                    ;Depose  A0
         CMP.W     #$0C,D0                     ;Check  if first  track
         BMI.S     Calc3                       ;Fall  thru  if true
         TST.B     $0B22                       ;See  if single  or double
         BNE       Calc3                       ;Branch  if double
         MOVE.W    #$FF8F,D1                   ;Else  fall  into  old routine
         LEA       Calc2+2,A0
         AND.W     D1,(A0)
         MOVE.L    ScrnBase,D1                 ;Fetch  screen  base
         AND.L     #$00700000,D1               ;Strip  off megabyte  count
         SWAP      D1                          ;Put  in low word
         OR.W      D1,(A0)                     ;Add  to jump  address
         MOVE.W    #$640,D1                    ;Stuff  we trashed  for  patch
         LSR.W     #$1,D1
         MOVE.L    (SP)+,A0                    ;Restore  A0 to its  throne
;Calc2   JMP       $000CFFDE                   ;Return  to control  flow
Calc2    JMP       $000E42DE

Calc3    MOVE.L    (SP)+,A0
         MOVE.B    #$1,(A0)                              ;For  compatibility
         MOVEM.L   A1-A4/D1-D3/D6/D7,-(SP)               ;Store  registers
         LEA       Sectors+20,A0                         ;Get  addr  after  sector  info
         MOVE.W    #$0640,D1                             ;Set  up max  sector  count
Calc4    MOVE.W    D1,D3                                 ;Dupe  current  sector  count
```

```
        MOVE.L   -(A0),D1                           ;Get next sector info
        CMP.W    D0,D1                              ;See if in right one yet
        BGT.S    Calc4                              ;Nope
        MOVE.W   D3,D1
        MOVEQ    #0,D5                              ;Default to side 0
        MOVEQ    #0,D4
        MOVEQ    #0,D2
        MOVE.B   (A0),D4 *                          ;Get track count
        MOVE.B   1(A0),D2                           ;Get sector per track count
        BRA.S    Calc6
Calc5   SUBQ     #1,D4                              ;Decrement track count
Calc6   SUB.W    D2,D1                              ;Decrement to get right count
        CMP.W    D0,D1
        BGT.S    Calc5
        SUB.W    D1,D0
        LSR.W    #1,D2                              ;Halve sector count and de-
        SUBQ     #1,D2                              ;crement so sides work right.
        CMP.W    D0,D2                              ;See if side one
        BPL      Calc7                              ;Nope
        MOVEQ    #1,D5                              ;Set to side one
        ADDQ     #1,D2
        SUB.W    D2,D0                              ;Fix for side sector size
Calc7   MOVEM.L  (SP)+,A1-A4/D1-D3/D6/D7            ;Restore registers
        MOVE.L   (SP)+,A0                           ;Start pulling off of stack
        MOVE.W   D5,(A0)                            ;Put side away
        MOVE.L   (SP)+,A0
        MOVE.W   D4,(A0)                            ;Put track away
        MOVE.L   (SP)+,A0
        MOVE.W   D0,(A0)                            ;Put sector away
        JMP      (A1)                               ;End of patch

InitPtch1
        CMPI.W   #$15,$1A(A0)                       ;See if HD20 call
        BNE      IP1a                               ;And branch if not
        CMPI.W   #$FFFB,$18(A0)                     ;Make sure it is Sony
        BNE      IP1a
        MOVEM.L  (A7)+,D3-D5/A2/A3
        MOVEQ    #-64,D0                            ;Indicate error
        RTS
IP1a    MOVEQ    #$02,D4                            ;Continue as usual
        JMP      $00401798

InitPtch2
        MOVEM.L  A0-A1,-(A7)                        ;Save some regs
        MOVEA.L  $030A,A1                           ;Load drive queue
        CMPI     #2,$1C(A0)                         ;Check two-sided format
        BNE      IP2a                               ;Branch if not
        MOVE     #$640,$0C(A1)                      ;Set volume size to 800K
        MOVE.B   #$FF,$0B22                         ;Set HFS flag
        MOVEM.L  (A7)+,A0-A1
        MOVEM.L  D1-D6,-(A7)
        JSR      (A3)                               ;Do format
        BRA.S    IP2b
IP2a    MOVE     #$320,$0C(A1)                      ;Set volume size to 400K
        CLR.B    $0B22                              ;Clear HFS flag
        MOVEM.L  (A7)+,A0-A1
        MOVEM.L  D1-D6,-(A7)
        MOVE.B   #$01,$FCC015                       ;Set to single-side format
        JSR      (A3)                               ;Do format
        MOVE.B   #$02,$FCC015                       ;Restore double-side
IP2b    MOVEM.L  (A7)+,D1-D6
        JMP      $401CF8                            ;Back to normal flow

SidePtch CMPI.W  #$4244,$08(A2)                     ;Retain old check
        BEQ.S    HFSCalcEnd
        EXG      A4,D0                              ;Test if A4 is odd
```

```
         BTST.L    #0,D0
         EXG       A4,D0                           ;Put it back
         BNE       OddA4                           ;Go do it
         CMPI.W    #$4244,$1C(A4)                  ;Check HFS
         BNE       MFSCheck                        ;Branch if not
         MOVE.B    #$FF,$0B22                      ;Set HFS flag
         BRA.S     MFSCheckEnd
MFSCheck CMPI.W    #$D2D7,$1C(A4)                  ;If MFS mount clear HFS flag
         BNE       HFSCalcEnd
OddA4    CLR.B     $0B22                           ;Set MFS flag
         CMPI.W    #$4244,$08(A2)                  ;Exit without checking A4
         BRA.S     HFSCalcEnd                      ;One exit only
MFSCheckEnd
         CMPI.W    #$4244,$1C(A4)                  ;Set cond. codes correctly
HFSCalcEnd
         RTS

;* Init800 Constants  *
;*********************

HFSDefaults
         DC.W      $4244                ;sigWord
         DC.L      $00000000            ;abSize
         DC.L      $00000000            ;clpSize
         DC.L      $00000010            ;nxFreeFN
         DC.L      $00000000            ;btClpSize

Sectors  DC.L      $0F180000,$1F160180  ;Track/SectorCount/Sector#
         DC.L      $2F1402E0,$3F120420
         DC.L      $4F100540
InitEnd  DC.W      $0000

;* Lisa800 Constants  *
;*********************

LogoWndw DC.L      0                    ;Where to store (on heap)
         DC.B      1,0                  ;Visible
         DC.W      3                    ;Type
         DC.L      -1                   ;Front window
         DC.B      0,0                  ;No go-away flag
LItmHndl DC.L      LogoItms             ;Item handle
         .ALIGN    2

LogoItms DC.W      1                    ;Number of items
LogoItm1 DC.L      0                    ;Handle space
         DC.W      260,34,290,134       ;Rectangle
         DC.B      4                    ;Normal button
         DC.B      'Cancel'             ;Length + title
LogoItm2 DC.L      0
         DC.W      260,172,290,272
         DC.B      4
         DC.B      'Install'
         DC.B      0
         .ALIGN    2

LString1 DC.W      0,12,21,110,115
         DC.B      'Lisa800 updates the MacWorker operating system '
LString2 DC.W      0,12,21,110,127
         DC.B      'on the hard-drive of your Lisa-2r or Macintosh XLr'
LString3 DC.W      0,12,21,110,139
         DC.B      'computer so that you will be able to use the LISAshop '
LString4 DC.W      0,12,21,110,151
         DC.B      '800K diskette drive to read and write double-sided'
LString5 DC.W      0,12,21,110,163
         DC.B      'diskettes with Apple''sr hierarchical file system! '
LString6 DC.W      0,12,21,110,180
```

```
         DC.B    'To begin installation,  click  on the ''Install''  button  at the '
LString7 DC.W    0,12,21,110,192
         DC.B    'bottom  of this window.  Installation  will  take  about  one'
LString8 DC.W    0,12,21,110,204
         DC.B    'minute.  When  it is finished,  you will  be able  to use  '
LString9 DC.W    0,12,21,110,216
         DC.B    '800K  diskettes  from  any Macintoshr,  as well  as be  '
LStringA DC.W    0,12,21,110,228
         DC.B    'able  to read  and write  and format  your  own!  '
         .ALIGN  2

TitleString
         DC.W    16,18,21,220,60              ;Face,Size,Font,X,Y
         DC.B    'Lisa800 '                   ;Length,Text
CRString DC.W    0,14,5,187,265
         DC.B    'Copyright  (C)  1987/88'
         DC.B    0
ISWString
         DC.W    0,14,21,210,280
         DC.B    'The  LISAshop'
         DC.B    0
VerString
         DC.W    0,10,21,235,90
         DC.B    'Version  1.0'
MeString DC.W    0,12,21,186,79
         DC.B    'by  Charles  T.  Lukaszewski'
         .ALIGN  2
InstText DC.W    0,12,0,125,184
         DC.B    ' Installing  Lisa800  to MacWorksr '
         .ALIGN  2
DoneText DC.W    0,12,0,130,184
         DC.B    'Installation  Complete!  Rebooting '
         .ALIGN  2
ErrText  DC.W    0,12,0,140,184
         DC.B    'Error  while  installing!'
         .ALIGN  2
LisaErr  DC.W    0,12,0,100,184
         DC.B    'Lisa800  requires  a Lisa-2r/Macintosh  XLr!'
         .ALIGN  2
RunErr   DC.W    0,12,0,100,184
         DC.B    'Lisa800  cannot  be run  from  the  hard  disk!'
         .ALIGN  2
TwiceErr DC.W    0,12,0,120,184
         DC.B    'Lisa800  has already  been  installed!'
         .ALIGN  2
HDErr    DC.W    0,12,0,100,184
         DC.B    'Could  not  find  the  hard  disk!'
         .ALIGN  2
HFSErr   DC.W    0,12,0,100,184
         DC.B    'HFS  must  be  present!'
         .ALIGN  2
RAMErr   DC.W    0,12,0,100,184
         DC.B    'Not  enough  memory  for  installation!'
         .ALIGN  2

DirPos   DC.W    $46A,$482,$484,$49C,$49E,$4B6,$4B8,$4D0
         DC.W    $4D2,$4EA,$4EC,$504,$506,$51E,$520,$000

TextWndw DC.L    0
         DC.B    1,0
         DC.W    1
         DC.L    -1
         DC.B    0,0

LogoSize DC.W    35,106,330,406
TextSize DC.W    170,90,190,422
```

```
BeginSiz  DC.W       295,140,325,240
LisaEnd   DC.W       $0000

;*  Lisa800

;*  Lisa800  Globals  *
;********************

TempA     DS.L       1                                   ;Scratch  memory
TempB     DS.L       1
TempC     DS.L       15
RefNum0   DS.W       1                                   ;File  RefNum  storage
RefNum1   DS.W       1
Junk      DS.L       1
Buf       DS.W       $200                                ;Two  sector  buffer
EvtRec    DS.L       5                                   ;Event  record  storage
PBRec     DS.L       20                                  ;Parameter  Block  record
          END
```