```
0000!                                   .PROC   IBMROM
0000!                           ;***************************************************************
0000!                           ;
0000!                           ; Program  :  ROM.IBM
0000!                           ; Author   :  S. G. Schwartz
0000!                           ; Date     :  2-Dec-82
0000!                           ; Purpose  :  This is the ROM on the IBM/PC interface cards
0000!                           ;
0000!                           ;             The i/o services provide for six  calls.
0000!                           ;               (1) identify card type (omniNet or Flat Cable)
0000!                           ;               (2) send/recv cmd to disk drive
0000!                           ;               (3) send message to server
0000!                           ;               (4) send/recv msg from server
0000!                           ;               (5) find any disk server on network
0000!                           ;               (6) send write cmd to disk drive
0000!                           ;
0000!                           ; History   : first Version 2-Dec-82                          SGS
0000!                           ;
0000!                           ;***************************************************************
0000!
0000! 0000                      omniNet         .EQU    0
0000! 0001                      flatCbl         .EQU    1
0000!
0000! 0001                      cardType        .EQU    flatCbl
0000! 0015                      version         .EQU    15H
0000!                                   .MACRO  ifOmniNet
0000!                                   CMPBIM  CS:interFace,omniNet
0000!                                   JE      %1
0000!                                   .ENDM
0000!
0000!                                   .MACRO  ifFlatCable
0000!                                   CMPBIM  CS:interFace,flatCbl
0000!                                   JE      %1
0000!                                   .ENDM
0000!
0000!                           ;***************************************************************
0000!                           ;
0000!                           ;ROM -- This is the jump vector
0000!                           ;
0000!                           ;***************************************************************
0000!
0000! E9 ****                  coldStart       JMP     coldContinue            ;cold start entry
0003! E9 ****                  warmStart       JMP     warmContinue            ;warm start entry
0006! E9 ****                  ioStart         JMP     romIo                   ;i/o entry point
0009! E9 ****                  dummyInterrupt  JMP     dummyIret               ;dummy interrupt
000C!
000C! 28 43 29 20 43 6F 70     helloMsg        .ASCII  "(C) Copyright 1983 by Corvus Systems, Inc."
0036! 0D 0A                                    .BYTE   0DH,0AH
0038! 52 6F 6D 20 76 65 72                     .ASCII  "Rom version 1.5 -- loading boot program..."
0062! 0D 0A 00                                 .BYTE   0DH,0AH,0
0065!
0065!                           warmContinue
0065! E8 ****                                   CALL    omInit                  ;init the xporter interface
```

```
0068: CB                                RETL
0069:                           coldContinue
0069: E8 ****                           CALL    omInit                  ;init the xporter interface
006C: E9 ****                           JMP     loadBoot                ;const ii boot
006F:                           dummyIret
006F: CF                                IRET                            ;dummy IRET because ROM basic sets
0070:                                                                   ;up interupt 1B AND 1C, we must
0070:                                                                   ;dummy them thru here
0070:
```

```
0070!
0070!                    ;*************************************************************************
0070!                    ;
0070!                    ; IBM/PC ROM I/O SERVICES
0070!                    ;
0070!                    ;
0070!                    ; (AH) = 0      Identify interface
0070!                    ;
0070!                    ;               On exit
0070!                    ;               (AL) = 0 if omniNet interface card
0070!                    ;                      1 if flatCable interface
0070!                    ;
0070!                    ;               with omniNet interfaces
0070!                    ;
0070!                    ;               (AH) = omniNet transporter number (workstation number)
0070!                    ;                      If (AH) = 255 then transporter with same number
0070!                    ;                      is on network.
0070!                    ;
0070!                    ; (AH) = 1      Xmit/Recv data to drive
0070!                    ;
0070!                    ;               DS:(SI) address of data to send to drive
0070!                    ;               ES:(DI) address of buffer for data from drive
0070!                    ;               (CX)    number of bytes of data to send to drive (MAX = 530d)
0070!                    ;               (DX)    number of bytes wanted back from drive (do not include
0070!                    ;                       return code - MAX = 530d)
0070!                    ;               with OmniNet interfaces
0070!                    ;
0070!                    ;               (AL)    network address of disk server
0070!                    ;               (BL)    number of timer units to wait for reply from disk server.
0070!                    ;                        0 = do not abort, wait forever.
0070!                    ;                       timer unit is approx .86 seconds
0070!                    ;               (BH)    number of xmit trys. 255 = retry until success. Should
0070!                    ;                       be greater 0. 0 = 255 trys.
0070!                    ;
0070!                    ;
0070!                    ;               with flatCable interfaces
0070!                    ;
0070!                    ;               (BL)    number of timer units to wait for disk drive to come ready
0070!                    ;                       before first byte is sent to drive. This is also the number
0070!                    ;                       of timer units to wait for the interface to turn around
0070!                    ;                       after the command sequence is sent to the drive.
0070!                    ;
0070!                    ;                       0 = do not abort, wait forever.
0070!                    ;                       timer unit is approx .86 seconds
0070!                    ;
0070!                    ;               On exit
0070!                    ;
0070!                    ;               (AL)    return code from drive. 255 = aborted
0070!                    ;               (CX)    number of bytes received from drive. Count includes
0070!                    ;                       the return code;
0070!                    ;
0070!                    ; (AH) = 2      Xmit data to network server
0070!                    ;
```

```
0070!                           ;               ES:(SI) address of data to transmit to server
0070!                           ;               (CX)    number of bytes to transmit
0070!                           ;               (AL)    network address of server. Use 255 to broadcast
0070!                           ;                        to all servers.
0070!                           ;               (BH)    number of times to try to xmit. If this is a broadcast
0070!                           ;                        number of times to xmit data.
0070!                           ;
0070!                           ;               On exit
0070!                           ;
0070!                           ;               (AL)    0 = all ok
0070!                           ;                     255 = transmission aborted
0070!                           ;
0070!                           ; (AH) = 3      Xmit/Recv data to network server
0070!                           ;
0070!                           ;               DS:(SI) address of data to transmit to server
0070!                           ;               ES:(DI) address of buffer for data from server
0070!                           ;               (CX)    number of bytes to transmit (530d max)
0070!                           ;               (DX)    number of bytes to receive (530d max)
0070!                           ;               (AL)    network address of server. Use 255 to broadcast
0070!                           ;                        to all servers.
0070!                           ;               (BL)    number of timer units to wait for reply from server.
0070!                           ;                        0 = do not abort, wait forever.
0070!                           ;                        timer unit = approx .86 seconds
0070!                           ;               (BH)    number of times to try. Should be greater than
0070!                           ;                        zero. Zero = 255 times.
0070!                           ;
0070!                           ;               On exit
0070!                           ;
0070!                           ;               (AL)    0 = all ok
0070!                           ;                     255 = transmission aborted
0070!                           ;
0070!                           ; (AH) = 4      Find any disk server on network. Uses simple broadcast command
0070!                           ;                any version disk server recognizes this command. Requests 2 and 3
0070!                           ;                require a disk server with multiple server support.
0070!                           ;
0070!                           ;               (BL)    number of timer units to wait for reply from server.
0070!                           ;                        0 = do not abort, wait forever.
0070!                           ;                        timer unit = approx .86 seconds
0070!                           ;               (BH)    number of times to try. Should be greater than
0070!                           ;                        zero. Zero = 255 times.
0070!                           ;
0070!                           ;               On exit
0070!                           ;
0070!                           ;               (AL)    0 = all ok
0070!                           ;                     255 = transmission aborted
0070!                           ;
0070!                           ;               (AH)    network address of disk server that responded
0070!                           ;
0070!                           ; (AH) = 5      Send write command to drive
0070!                           ;
0070!                           ;               DS:(SI) address of command to send to drive
0070!                           ;               ES:(DI) address of data to send to drive
0070!                           ;               (CX)    number of bytes of command to send to drive (normally 4)
0070!                           ;               (DX)    number of bytes of data to send to drive (normally 512.)
0070!                           ;
```

```
0070!                   ;               with OmniNet interfaces
0070!                   ;
0070!                   ;               (AL)    network address of disk server
0070!                   ;               (BL)    number of timer units to wait for reply from disk server.
0070!                   ;                        0 = do not abort; wait forever.
0070!                   ;                       timer unit is approx .86 seconds
0070!                   ;               (BH)    number of xmit trys. 255 = retry until success. Should
0070!                   ;                       be greater 0. 0 = 255 trys.
0070!                   ;
0070!                   ;
0070!                   ;               with flatCable interfaces
0070!                   ;
0070!                   ;               (BL)    number of timer units to wait for disk drive to come ready
0070!                   ;                       before first byte is sent to drive. This is also the number
0070!                   ;                       of timer units to wait for the interface to turn around
0070!                   ;                       after the command sequence is sent to the drive.
0070!                   ;
0070!                   ;                       0 = do not abort, wait forever.
0070!                   ;                       timer unit is approx .86 seconds
0070!                   ;
0070!                   ;
0070!                   ;               On exit
0070!                   ;
0070!                   ;               (AL)    return code from drive. 255 = aborted
0070!                   ;               (CX)    number of bytes received from drive. Count includes
0070!                   ;                       the return code;
0070!                   ;
0070!                   ;*******************************************************************
0070!
0070!                   ;*******************************************************************
0070!                   ;
0070!                   ; RECORD DESCRIPTIONS
0070!                   ;
0070!                   ;*******************************************************************
0070!                           .ASECT
0000!                           .ORG    0
0000!                   bootCmd         .BYTE                   ;const ii boot drive cmd
0001!                   bootType        .BYTE                   ;computer number(type)
0002!                   bootBlock       .BYTE                   ;block number to read
0003!
0003!                   bootEnd
0003!
0003! 0003             bootLen         .EQU    bootEnd
0003!                           .PSECT
0070!
0070!                   ;*******************************************************************
0070!                   ;
0070!                   ; EQUATES
0070!                   ;
0070!                   ;*******************************************************************
0070! 00FF             brdCast         .EQU    0FFH            ;xporter brdcast address
0070! 0040             xmitCmd         .EQU    40H             ;xporter send cmd
0070! 0006             ioEntry         .EQU    6               ;io offset into Rom
0070! DF00             romSeg          .EQU    0DF00H          ;seg address for ROM
```

```
0070: 7C00                  loadAdd         .EQU    7C00H
0070: FFFE                  stackAdd        .EQU    0FFFEH
0070: 0009                  ibmType         .EQU    9                       ;computer type number for ibm
0070:
0070: 7000                  bootServer      .EQU    7000H                   ;workSpace for loadBoot
0070: 7001                  bootSource      .EQU    bootServer+1            ;   "       "     "
0070: 7004                  loadAddReg      .EQU    bootSource+bootLen      ;   "       "     "
0070:
0070:                       ;***************************************************************
0070:                       ;
0070:                       ; MESSAGES
0070:                       ;
0070:                       ;***************************************************************
0070: 45 72 72 6F 72 20 6C  bootMsg         .ASCII  "Error loading boot program "
008B: 00                                    .BYTE   0
008C: 54 72 61 6E 73 70 6F  dupXportMsg     .ASCII  "Transporter with same address is on the network "
00BC: 00                                    .BYTE   0
00BD: 54 68 65 72 65 20 61  noSrvMsg        .ASCII  "There are no disk servers on the network "
00E6: 00                                    .BYTE   0
00E7: 36 34 4B 20 6D 65 6D  need64msg       .ASCII  "64K memory required"
00FA: 00                                    .BYTE   0
00FB:
00FB:                       ;***************************************************************
00FB:                       ;
00FB:                       ; OMNINET BUFFERRED TRANSPORTER CONSTANTS
00FB:                       ;
00FB:                       ; The bufferred Omninet transporter has an on-board 4K RAM buffer
00FB:                       ; that is pointed to by a 12-bit counter
00FB:                       ;
00FB:                       ;***************************************************************
00FB:
00FB: 0248                  omStatus        .EQU    0248H   ;input port to read status of transporter
00FB:                                                       ;to wait for transporter ready (bit 7 high)
00FB:                                                       ;when strobing address to transporter
00FB:
00FB: 0249                  omHeader        .EQU    0249H   ;input port to read transporter buffer
00FB:                                                       ;at loc pointed to by xporter counter
00FB:                                                       ;without counter auto-incrementing, used to
00FB:                                                       ;read command header or result vector
00FB:
00FB: 024B                  omData          .EQU    024BH   ;r/w port to read/write transporter buffer
00FB:                                                       ;mem at location pointed to by xporter counter
00FB:                                                       ;and counter is automatically incremented
00FB:
00FB: 0249                  omniPort        .EQU    0249H   ;output port to strobe address to transporter
00FB:
00FB: 0248                  omHiCntr        .EQU    0248H   ;output port to set high byte of counter
00FB: 024A                  omLoCntr        .EQU    024AH   ;output port to set low byte of counter
00FB:
00FB: 0080                  omRdyBit        .EQU    80H     ;mask for transporter ready bit
00FB:                       ;***************************************************************
00FB:                       ;
00FB:                       ; FLATCABLE INTERFACE CONSTANTS
00FB:                       ;
00FB:                       ;***************************************************************
```

```
00FB!
00FB! 02EE                  flatData        .EQU    02EEH                   ;data i/o port number
00FB! 02EF                  flatStatus      .EQU    02EFH                   ;status port number
00FB! 0001                  flatReady       .EQU    1                       ;drive ready bit mask
00FB! 0002                  flatBusDir      .EQU    2                       ;bus direction bit mask
00FB!
00FB!                       ;*****************************************************************
00FB!                       ;
00FB!                       ; OMNINET TRANSPORTER CONSTANTS, EQUATES, and MACROS
00FB!                       ;
00FB!                       ;*****************************************************************
00FB!
00FB! 0002                  goLen           .EQU    2       ; LENGTH OF GO MSG
00FB! 00B0                  rcvSocket       .EQU    0B0H    ; SOCKET TO RECEIVE
00FB! 00B0                  sendSocket      .EQU    0B0H    ; DISK SERVER REQUEST SOCKET
00FB! 00A0                  restSocket      .EQU    0A0H    ; DISK SERVER REST OF CMD SOCKET
00FB! 0080                  brdSocket       .EQU    80H     ; DISK SERVER BROADCAST SOCKET
00FB! 0080                  srvSocket       .EQU    80H     ; SOCKET TO RECEIVE SERVER COMMANDS
00FB! 0212                  buffLen         .EQU    530.    ;length of send and recv buffers
00FB!
00FB!                                       .MACRO  flipRel
00FB!                                       .BYTE   <%1-omVectors> / 256
00FB!                                       .BYTE   <%1-omVectors> & 0FFH
00FB!                                       .ENDM
00FB!
00FB!                                       .MACRO  flip
00FB!                                       .BYTE   %1 / 256
00FB!                                       .BYTE   %1 & 0FFH
00FB!                                       .ENDM
00FB!
00FB!                                       .MACRO  ifUnder5
00FB!                                       CMPBIM  (BP+cmd),5
00FB!                                       JE      $10
00FB!                                       CMP     (BP+sendLen),5
00FB!                                       JB      %1
00FB!                       $10
00FB!                                       .ENDM
00FB!
00FB!                       ;*****************************************************************
00FB!                       ;
00FB!                       ; WORK_SPACE RECORD DESCRIPTION
00FB!                       ;
00FB!                       ;*****************************************************************
00FB!                                       .ASECT
0003!                                       .ORG    0
0000!                       sendBufAdd      .WORD                   ;address of users data
0002!                       recvBufAdd      .WORD                   ;address of user buffer
0004!                       sendSeg         .WORD                   ;seg reg to address the above
0006!                       recvSeg         .WORD
0008!
0008!                       sendLen         .WORD                   ;length of users data
000A!                       recvLen         .WORD                   ;length of users buffer
000C!
000C!                       cmd             .BYTE                   ;special handling for cmd 2
000D!                       server          .BYTE                   ;server network address
```

```
000E!                       waitTime        .BYTE
000F!                       retrys          .BYTE
0010!                       stopTime        .WORD                   ;used to calc abort time
0012!                       cmdByte         .BYTE                   ;used by retry/abort routines
0013!
0013!                       cmdAddr         .WORD                   ;xporter add of cmd vector
0015!                       rsltAddr        .WORD                   ;xporter add of result vector
0017!                       mask            .BYTE                   ;for setup routines
0018!
0018!                       xResult         .BYTE                   ;transporter status code
0019!                       xHost           .BYTE                   ;source host #
001A!                                       .WORD                   ;actual len of received data (msb first)
001C!
001C!                       xRetLen         .WORD                   ;length of data received from server
001E!                       xRetCode        .BYTE                   ;return code from server
001F!
001F!                       workEnd
001F! 001F                  workSize        .EQU    workEnd
001F!                                       .PSECT
00FB!
00FB!
00FB!                       ;*****************************************************************
00FB!                       ;
00FB!                       ;       OMNINET TRANSPORTER RESULT VECTORS, COMMAND VECTORS,
00FB!                       ;                       and BUFFERS
00FB!                       ;
00FB!                       ;*****************************************************************
00FB!                       omVectors
00FB!
00FB!                       ;*****************************************************************
00FB!                       ;
00FB!                       ; R E S U L T   V E C T O R S   A R E A
00FB!                       ;
00FB!                       ;*****************************************************************
00FB!
00FB!                       ;*****************************************************************
00FB!                       ;
00FB!                       ; RESULT VECTOR FOR RECEIVE MESSAGE FROM DISK SERVER -- COMMAND COMPLETE
00FB!                       ;
00FB!                       ;*****************************************************************
00FB! 00                    rResult         .BYTE   0               ;transporter status code
00FC! 00                    rResHost        .BYTE   0               ;source host #
00FD! 0000                                  .WORD   0               ;actual len of received data (msb first)
00FF!
00FF! 0000                  rHdrRetLen      .WORD   0               ;length of data received from server
0101! 00                    rHdrRetCode     .BYTE   0               ;return code from server
0102!
0102!                       ;*****************************************************************
0102!                       ;
0102!                       ; RESULT VECTOR FOR MESSAGE OF 1 TO 4 BYTES IN LENGTH
0102!                       ;
0102!                       ;*****************************************************************
0102! 00                    sResult         .BYTE   0               ;transporter status code
0103! 00 00 00                               .BLOCK  3               ;3 bytes filler
0106!
```

```
0106: 0000                  sHdrToLen       .WORD   0               ;# of bytes sent to server
0108:                       sHdrRetLen      flip    buffLen         ;# of bytes wanted from server
010A:
010A:                       ;*****************************************************************
010A:                       ;
010A:                       ; RESULT VECTOR FOR SERVER MESSAGE
010A:                       ;
010A:                       ;*****************************************************************
010A: 00                    srvResult       .BYTE   0               ;server xmit command result
010B:
010B:                       ;*****************************************************************
010B:                       ;
010B:                       ; RESULT VECTOR FOR SERVER RECEIVE MESSAGE
010B:                       ;
010B:                       ;*****************************************************************
010B: 00                    srvRResult      .BYTE   0
010C: 00                    srvRHost        .BYTE   0               ;host num of sender
010D: 0000                  srvRLen         .WORD   0               ;length of message
010F:
010F:                       ;*****************************************************************
010F:                       ;
010F:                       ; DISK SERVER SENDS BACK A MESSAGE SAYING GO
010F:                       ;
010F:                       ;*****************************************************************
010F: 00 00                 goData          .BLOCK  GO_LEN
0111:
0111:                       ;*****************************************************************
0111:                       ;
0111:                       ; RESULT VECTOR TO PROCEED MESSAGE WITH 5 BYTES OR MORE LENGTH
0111:                       ;
0111:                       ;*****************************************************************
0111: 00                    goResult        .BYTE   0               ;transporter status code
0112: 00                    goResHost       .BYTE   0               ;source host #
0113: 0000                                  .WORD   0               ;length of go msg
0115:
0115:                       ;*****************************************************************
0115:                       ;
0115:                       ; RESULT VECTOR FOR BYTES 5-LEN MESSAGE
0115:                       ;
0115:                       ;*****************************************************************
0115: 00                    restResult      .BYTE   0               ;return code
0116:
0116:                       ;*****************************************************************
0116:                       ;
0116:                       ; RESULT VECTOR FOR CANCEL SOCKET COMMAND
0116:                       ;
0116:                       ;*****************************************************************
0116: 00                    canResult       .BYTE   0
0117:
0117:                       ;*****************************************************************
0117:                       ;
0117:                       ; RESULT VECTOR FOR WHO AM I CMD
0117:                       ;
0117:                       ;*****************************************************************
0117: 00                    whoResult       .BYTE   0
```

```
0118!
0118!                       ;*****************************************************************
0118!                       ;
0118!                       ; RESULT VECTOR FOR ECHO CMD
0118!                       ;
0118!                       ;*****************************************************************
0118! 00                    echoResult      .BYTE   0
0119!
0119!                       ;*****************************************************************
0119!                       ;
0119!                       ; C O M M A N D   V E C T O R S   A R E A
0119!                       ;
0119!                       ;*****************************************************************
0119!
0119!                       ;*****************************************************************
0119!                       ;
0119!                       ; GO_CMD
0119!                       ;
0119!                       ; <<RECEIVE PROCEED MESSAGE FROM DISK SERVER -- TRANSPORTER COMMAND>>
0119!                       ;*****************************************************************
0119! F0                    goCmd           .BYTE   0F0H            ; TRANSPORTER SET UP RECEIVE CMD
011A! 00                                    .BYTE   0               ; MSB OF 3 BYTES OF
011B!                                       FLIPREL goResult        ; ADDR OF GO RESULT VECTOR
011D! B0                                    .BYTE   rcvSocket       ; SOCKET TO READ INTO
011E! 00                                    .BYTE   0               ; MSB OF 3 BYTES OF
011F!                                       FLIPREL goData          ; ADDRESS OF DATA
0121! 00 02                                 .BYTE   0,goLen         ; LENGTH OF DATA BUFFER
0123! 00                                    .BYTE   0               ; LENGTH OF HEADER
0124!
0124!                       ;*****************************************************************
0124!                       ;
0124!                       ;  CAN_RECV_CMD
0124!                       ;
0124!                       ;  <<CANCEL NORMAL RECEIVE SOCKET TRANSPORTER COMMAND>>
0124!                       ;
0124!                       ;*****************************************************************
0124! 10                    canRecvCmd      .BYTE   010H            ;TRANSPORTER CANCEL SOCKET CMD
0125! 00                                    .BYTE   0
0126!                                       FLIPREL canResult       ;ADDR OF RESULT VECTOR
0128! B0                                    .BYTE   rcvSocket       ;SOCKET TO CANCEL
0129!
0129!                       ;*****************************************************************
0129!                       ;
0129!                       ;  CAN_SRV_CMD
0129!                       ;
0129!                       ;  <<CANCEL CONST II RECEIVE SOCKET TRANSPORTER COMMAND>>
0129!                       ;
0129!                       ;*****************************************************************
0129! 10                    canSrvCmd       .BYTE   010H            ;TRANSPORTER CANCEL SOCKET CMD
012A! 00                                    .BYTE   0
012B!                                       FLIPREL canResult       ;ADDR OF RESULT VECTOR
012D! 80                                    .BYTE   srvSocket       ;SOCKET TO CANCEL
012E!
012E!                       ;*****************************************************************
012E!                       ;
```

```
012E:                           ; R_CMD
012E:                           ;
012E:                           ; <<SETUP RECEIVE - TRANSPORTER COMMAND>>
012E:                           ;*****************************************************************
012E: F0                        rCmd            .BYTE   0F0H            ; SETUP RECEIVE CMD
012F: 00                                        .BYTE   0               ; MSB OF 3 BYTES OF
0130:                                           FLIPREL rResult         ; ADDR OF RESULT VECTOR
0132: B0                                        .BYTE   rcvSocket       ; SOCKET TO READ INTO
0133: 00                                        .BYTE   0               ; MSB OF 3 BYTES OF
0134: ****                      rCmdData        .WORD   recvBuf-omvectors ;will be flipped
0136:                           rCmdLen         flip    buffLen         ; READ 512 BYTES
0138: 03                                        .BYTE   3               ; HEADER LENGTH IS 3
0139:
0139:                           ;*****************************************************************
0139:                           ;
0139:                           ; S_CMD
0139:                           ;
0139:                           ; <<TRANSMIT 1ST 4 BYTES TO DRIVE -- TRANSPORTER COMMAND>>
0139:                           ;*****************************************************************
0139: 40                        sCmd            .BYTE   40H             ; TRANSPORTER SEND MSG CMD
013A: 00                                        .BYTE   0               ; MSB OF 3 BYTES OF
013B:                                           FLIPREL sResult         ; ADDR OF RESULT VECTOR
013D: B0                                        .BYTE   sendSocket      ; SOCKET TO SEND TO
013E: 00                                        .BYTE   0               ; MSB OF 3 BYTES OF
013F: ****                      sCmdData        .WORD   sendBuf-omVectors ;ADDR OF DATA
0141: 00 04                                     .BYTE   0,4             ; DATA LENGTH IS 4
0143: 04                                        .BYTE   4               ; LENGTH OF CONTROL HEADER IS 4
0144: 00                                        .BYTE   0               ; NETWORK ADDRESS OF DISK SERVER
0145:
0145:                           ;*****************************************************************
0145:                           ;
0145:                           ; REST_CMD
0145:                           ;
0145:                           ; <<TRANSMIT BYTES 5-LEN TO DISK SERVER -- TRANSPORTER COMMAND>>
0145:                           ;*****************************************************************
0145: 40                        restCmd         .BYTE   40H             ; TRANSPORTER SEND MSG CMD
0146: 00                                        .BYTE   0               ; MSB OF 3 BYTES OF
0147:                                           FLIPREL restResult      ; ADDR OF RESULT VECTOR
0149: A0                                        .BYTE   restSocket      ; SOCKET TO TRANSMIT INTO
014A: 00                                        .BYTE   0               ; MSB OF 3 BYTES OF
014B: ****                      restData        .WORD   sendBuf-omVectors+4 ; ADDRESS OF DATA
014D: 0000                      restLen         .WORD   0               ; LENGTH OF DATA
014F: 00                                        .BYTE   0               ; NO HEADER
0150: 00                                        .BYTE   0               ; NETWORK ADDRESS OF DISK SERVER
0151:
0151:                           ;*****************************************************************
0151:                           ;
0151:                           ; SRV_CMD
0151:                           ;
0151:                           ; << transmits SERVER packet>>
0151:                           ;*****************************************************************
0151: 40                        srvCmd          .BYTE   40H             ; TRANSPORTER SEND MSG CMD
0152: 00                                        .BYTE   0
0153:                                           FLIPREL srvResult       ; ADD OF RESULT VECTOR
```

```
0155: 80                                .BYTE   srvSocket
0156: 00                                .BYTE   0
0157:
0157: ****              srvCmdData      .WORD   sendBuf-omvectors
0159: 0000              srvCmdLen       .WORD   0                       ; LENGTH OF MESSAGE
015B: 00                                .BYTE   0                       ; NO HEADER
015C: FF                                .BYTE   0FFH                    ; NETWORK ADD OF SERVER
015D:
015D:                   ;*****************************************************************
015D:                   ;
015D:                   ; SRV_R_CMD
015D:                   ;
015D:                   ; << sets up receive for  SERVER packet>>
015D:                   ;*****************************************************************
015D: F0                srvRCmd         .BYTE   0F0H                    ; set up receive
015E: 00                                .BYTE   0
015F:                                   FLIPREL srvRResult              ; add of result vector
0161: 80                                .BYTE   srvSocket
0162: 00                                .BYTE   0
0163: ****              srvRCmdData     .WORD   recvBuf-omVectors  ; add of data
0165:                   srvRCmdLen      flip    buffLen                 ; length of packet
0167: 00                                .BYTE   0                       ; length of header
0168:
0168:                   ;*****************************************************************
0168:                   ;
0168:                   ; WHO_CMD
0168:                   ;
0168:                   ;*****************************************************************
0168: 01                whoCmd          .BYTE   01                      ;who am i cmd
0169: 00                                .BYTE   0
016A:                                   FLIPREL whoResult
016C:
016C:
016C:                   ;*****************************************************************
016C:                   ;
016C:                   ; ECHO_CMD
016C:                   ;
016C:                   ;*****************************************************************
016C: 02                echoCmd         .BYTE   02                      ;echo cmd
016D: 00                                .BYTE   0
016E:                                   FLIPREL echoResult
0170: 00                echoHost        .BYTE   0
0171:
0171:                   ;*****************************************************************
0171:                   ;
0171:                   ; B_DATA    ( SENDS AN ILLEGAL COMMAND TO DRIVE)
0171:                   ;
0171:                   ;*****************************************************************
0171: 01 FE 01          bData           .BYTE   01,0FEH,01      ; PASSWORD TO DISK SERVER
0174: 00 01                             .BYTE   00,01           ; 1 BYTES TO DISKSERVER
0176: 00 00                             .BYTE   00,00           ; ZERO BYTES BACK
0178: FF                                .BYTE   0FFH            ; ILLEGAL COMMAND
0179:
0179:                   ;*****************************************************************
0179:                   ;
```

```
0179:                           ; B_CMD        TRANSMIT BROADCAST MESSAGE TO DISK SERVER
0179:                           ;
0179:                           ;*****************************************************************
0179: 40                        bCmd            .BYTE   40H             ; TRANSPORTER SEND MSG CMD
017A: 00                                        .BYTE   0
017B:                                           FLIPREL sResult         ; ADD OF RESULT VECTOR
017D: 80                                        .BYTE   srvSocket       ; SOCKET TO SEND TO
017E: 00                                        .BYTE   0
017F:                                           FLIPREL bData           ; ADD OF DATA TO SEND
0181: 00 08                                     .BYTE   0,08            ; LENGTH OF DATA
0183: 00                                        .BYTE   0               ; LENGTH OF HEADER
0184: FF                                        .BYTE   0FFH            ; broadcast this command
0185:
0185:                           ;*****************************************************************
0185:                           ;
0185:                           ; BR_CMD       RECEIVE BROADCAST MESSAGE FROM DISK SERVER
0185:                           ;
0185:                           ;*****************************************************************
0185: F0                        brCmd           .BYTE   0F0H            ; TRANSPORTER SET UP BLIND RECEIVE CMD
0186: 00                                        .BYTE   0
0187:                                           FLIPREL rResult         ; ADD OF RESULT VECTOR
0189: B0                                        .BYTE   rcvSocket       ; SOCKET TO READ INTO
018A: 00                                        .BYTE   0
018B: 0000                                      .WORD   0               ; ADDRESS OF DATA
018D: 0000                                      .WORD   0               ; LENGTH OF DATA
018F: 03                                        .BYTE   3               ; LENGTH OF HEADER
0190:
0190:                           omVectEnd
0190: 0095                      omCmdSize       .EQU    omVectEnd-omVectors
0190:
0190:                           ;*****************************************************************
0190:                           ;
0190:                           ; OMNINET SEND AND RECV BUFFERS (in transporter ram)
0190:                           ;
0190:                           ;*****************************************************************
0190:                           sendBuf                                 ;send buffer follows the last
0190:                           recvBuf                                 ; byte of vectors
0190:
0190:                           ;*****************************************************************
0190:                           ;
0190:                           ; END OF OMNINET DATA STRUCTURES
0190:                           ;
0190:                           ;*****************************************************************
```

```
0190!                   ;*****************************************************************
0190!                   ;
0190!                   ; CALL_TABLE -- call table for I/O services
0190!                   ;
0190!                   ;*****************************************************************
0190! ****              callTable       .WORD   romIo0
0192! ****                              .WORD   romIo1
0194! ****                              .WORD   romIo2
0196! ****                              .WORD   romIo3
0198! ****                              .WORD   romIo4
019A! ****                              .WORD   romIo1
019C! 0006              callEntries     .EQU    <$-callTable>/2
019C!
019C!                   ;*****************************************************************
019C!                   ;
019C!                   ; ROM_IO -- I/O services dispatcher
019C!                   ;
019C!                   ;*****************************************************************
019C!                   RomIo
019C! 82 FC 05                         CMP     AH,callEntries-1        ;valid request?
019F! 76**                            JNA     $10                     ; yes, continue
01A1! B0 FF                           MOV     AL,0FFH                 ;invalid request
01A3! CB                              RETL                            ;return
01A4!                   $10
01A4! 9C                              PUSHF                           ;save flags
01A5! FC                              CLD                             ;set direction forward
01A6!
01A6! 55                              PUSH    BP                      ;save all registers used
01A7! 06                              PUSH    ES                      ; except for AX AND CX
01A8! 1E                              PUSH    DS
01A9! 56                              PUSH    SI
01AA! 57                              PUSH    DI
01AB! 52                              PUSH    DX
01AC! 53                              PUSH    BX
01AD!
01AD! 83 EC 1F                        SUB     SP,workSize             ;allocate workspace on stack
01B0! 8B EC                           MOV     BP,SP                   ;workspace is addressable by
01B2!                                                                 ;(BP+displacement)
01B2! 8C 46 06                        MOV     (BP+recvSeg),ES         ;save all params in workspace
01B5! 8C 5E 04                        MOV     (BP+sendSeg),DS
01B8! 89 76 00                        MOV     (BP+sendbufAdd),SI
01BB! 89 7E 02                        MOV     (BP+recvBufAdd),DI
01BE! 89 4E 08                        MOV     (BP+sendLen),CX
01C1! 89 56 0A                        MOV     (BP+recvLen),DX
01C4! 88 46 0D                        MOV     (BP+server),AL
01C7! 88 5E 0E                        MOV     (BP+waitTime),BL
01CA! 88 7E 0F                        MOV     (BP+retrys),BH
01CD! 88 66 0C                        MOV     (BP+cmd),AH
01D0!
01D0! 32 C0                           XOR     AL,AL
01D2! 86 E0                           XCHG    AH,AL
01D4! D1 E0                           SHL     AX,1                    ;offset into call table
01D6! 8B F0                           MOV     SI,AX
```

```
01D8:
01D8: 2E FF 94 9001                     CALL    CS:callTable(SI)
01DD:
01DD: 83 C4 1F                          ADD     SP,workSize             ;deallocate workspace off
01E0:                                                                   ; of stack
01E0: 5B                                POP     BX                      ;restore all registers used
01E1: 5A                                POP     DX                      ; except AX and CX
01E2: 5F                                POP     DI
01E3: 5E                                POP     SI
01E4: 1F                                POP     DS
01E5: 07                                POP     ES
01E6: 5D                                POP     BP
01E7:
01E7: 9D                                POPF
01E8: CB                                RETL                            ;and return to caller
```

```
01E9:                   ;*************************************************************
01E9:                   ;OMINIT
01E9:                   ;       routine to initialize Omninet data structure
01E9:                   ;       at xporter buffer addr 0
01E9:                   ;
01E9:                   ;       flip the address bytes of all buffer address (assemblr restriction)
01E9:                   ;*************************************************************
01E9:                   omInit  ifFlatCable omInitEnd
01F1: 50                        PUSH    AX              ;save all registers user
01F2: 51                        PUSH    CX
01F3: 52                        PUSH    DX
01F4: 56                        PUSH    SI
01F5: 9C                        PUSHF
01F6:
01F6: 33 C0                     XOR     AX,AX           ;put at zero in xporter ram
01F8: E8 ****                   CALL    omSetCntr       ;set xporter ram counter
01FB: FC                        CLD
01FC: BE FB00                   LEA     SI,omVectors
01FF: B9 95 00                  MOV     CX,omCmdSize    ;copy vectors to xporter ram
0202: BA 4B 02                  MOV     DX,omData
0205:                   $5
0205: 2E AC                     LODSB   AL,CS:(SI)
0207: EE                        OUT     DX,AL
0208: E2FB                      LOOP    $5
020A:
020A: B8 39 00                  MOV     AX,rCmdData-omVectors   ;xchg the two bytes in the buffer
020D: E8 ****                   CALL    $10                     ; address fields of all cmd vectors
0210: B8 44 00                  MOV     AX,sCmdData-omVectors   ;the pascal assembler would not let the
0213: E8 ****                   CALL    $10                     ; fliprel macro work on forward refs
0216: B8 50 00                  MOV     AX,restData-omVectors
0219: E8 ****                   CALL    $10
021C: B8 5C 00                  MOV     AX,srvCmdData-omVectors
021F: E8 ****                   CALL    $10
0222: B8 68 00                  MOV     AX,srvRCmdData-omVectors
0225: E8 ****                   CALL    $10
0228:
0228: 9D                        POPF
0229: 5E                        POP     SI              ;restore all registers
022A: 5A                        POP     DX
022B: 59                        POP     CX
022C: 58                        POP     AX
022D: C3                        RET                     ;return
022E:
022E:                   $10
022E: 50                        PUSH    AX              ;xchg the two bytes in the xporter ram pointed
022F: E8 ****                   CALL    omSetCntr       ; to by AX
0232: EC                        IN      DX,AL
0233: 8A E8                     MOV     CH,AL
0235: EC                        IN      DX,AL
0236: 8A C8                     MOV     CL,AL
0238: 58                        POP     AX
0239: E8 ****                   CALL    omSetCntr
023C: 8A C1                     MOV     AL,CL
023E: EE                        OUT     DX,AL
```

```
023F: BA C5                             MOV     AL,CH
0241: EE                                OUT     DX,AL
0242:                           omInitEnd
0242: C3                                RET
0243:
0243:
0243:                           ;*****************************************************************
0243:                           ;
0243:                           ; ROM_IO_0 -- i/o service routine for interface identification
0243:                           ;
0243:                           ;*****************************************************************
0243:                           romIo0
0243:                                   ifFlatCable #5
024B: E8 ****                           CALL    setUpWho                ;get transporter network number
024E: B8 1C 00                          MOV     AX,whoResult-omVectors
0251: E8 ****                           CALL    omSetCntr
0254: BA 4B 02                          MOV     DX,omData
0257: EC                                IN      AL,DX
0258:
0258: 50                                PUSH    AX                      ;save result
0259: 50                                PUSH    AX
025A: B8 75 00                          MOV     AX,echoHost-omVectors   ;put our host number in echo packet
025D: E8 ****                           CALL    omSetCntr
0260: BA 4B 02                          MOV     DX,omData
0263: 58                                POP     AX                      ;get back our host number
0264: EE                                OUT     AL,DX                   ;put in xporter memory
0265:
0265: E8 ****                           CALL    setUpEcho               ;echo to our xporter number
0268: B8 1D 00                          MOV     AX,echoResult-omVectors ;get result
026B: E8 ****                           CALL    omSetCntr
026E: BA 4B 02                          MOV     DX,omData
0271: EC                                IN      AL,DX                   ;get it
0272: 3C C0                             CMP     AL,0C0H                 ;is somebody out there
0274: 58                                POP     AX
0275: 75**                              JNE     $10                     ;no
0277:                           $5
0277: B0 FF                             MOV     AL,0FFH                 ;mark as duplicate xporter
0279:                           $10
0279: 2E 8A 26 ****                     MOV     AH,CS:interface
027E: 86 E0                             XCHG    AH,AL
0280: C3                                RET
0281:
0281:                           ;*****************************************************************
0281:                           ;
0281:                           ; ROM_IO_1 -- interrupt service routine for general purpose
0281:                           ;             xmit/recv to drive
0281:                           ;
0281:                           ;*****************************************************************
0281:                           romIo1
0281:                                   ifOmniNet $10
0289: E9 ****                           JMP     flatIo1
028C:                           $10
028C: E8 ****                           CALL    omPutData               ;transfer user data to xporter buffer
028F:                                   ifUnder5  $50
029B:
```

```
029B: E8 ****                       CALL    setUpGo                 ;setup go receive for long cmd
029E: E8 ****                       CALL    setUpSend               ;send first 4 bytes of long cmd
02A1: 72**                          JC      romIo1err               ;cmd never made it to server
02A3:                       $23
02A3: B8 16 00                      MOV     AX,goResult-omVectors
02A6: E8 ****                       CALL    omSetCntr               ;point to go rslt vect on xporter
02A9:                       $24
02A9: BA 49 02                      MOV     DX,omHeader
02AC: EC                            IN      AL,DX                   ;get go_result transporter status
02AD: A8 80                         TEST    AL,80H
02AF: 74**                          JZ      $25                     ;we are posted with results
02B1: E8 ****                       CALL    chkTime
02B4: 73F3                          JNC     $24                     ;have not timed out yet
02B6: E9 ****                       JMP     romIo1err               ;abort call
02B9:                       $25
02B9: BA 48 02                      MOV     DX,omData               ;DX=r/w auto-increment port
02BC: EC                            IN      AL,DX
02BD: EC                            IN      AL,DX                   ;get go_res_host
02BE: 3A 46 0D                      CMP     AL,(BP+server)
02C1: 74**                          JE      $30                     ;jump if correct server answers
02C3: E8 ****                       CALL    setUpGo                 ;else try again
02C6: EBDB                          JMP     $23
02C8:                       $30
02C8: B8 14 00                      MOV     AX,goData-omVectors
02CB: E8 ****                       CALL    omSetCntr               ;GET 'G' OF GO
02CE: EC                            IN      AL,DX
02CF: A8 80                         TEST    AL,80H                  ;IS IT DISK SERVER RESTART
02D1: 75AE                          JNZ     romIo1                  ;YES RESTART
02D3:
02D3: E8 ****               $50     CALL    setUpReceive
02D6:                               ifUnder5  $60
02E2: E8 ****                       CALL    setUpRest
02E5: E9 ****                       JMP     $70
02E8:
02E8: E8 ****               $60     CALL    setUpSend
02EB: 72**                  $70     JC      romIo1err               ;xmit never made it to server
02ED: E8 ****                       CALL    omGetRslt       ;wait for completion & get result vector
02F0: 72**                          JC      romIo1err       ;abort call time out
02F2:
02F2: 8A 46 19                      MOV     AL,(BP+xHost)
02F5: 3A 46 0D                      CMP     AL,(BP+server)
02F8: 74**                          JE      $8
02FA: E8 ****                       CALL    setUpReceive
02FD: EBEC                          JMP     $70
02FF:
02FF: 8B 46 1C              $8      MOV     AX,(BP+xRetLen)         ;length of msg receved from drive
0302: 86 C4                         XCHG    AL,AH                   ;flip byte sex
0304: F6 C4 80                      TEST    AH,80H                  ;check for disk server restart
0307: 74**                          JZ      $9
0309: E9 75FF                       JMP     romIo1                  ;retry if yes
030C:
030C: E8 ****               $9      CALL    omGetData
030F:
030F: 8B 4E 1C                      MOV     CX,(BP+xRetLen)         ;save length for user in CX reg
0312: 86 E9                         XCHG    CH,CL
```

```
0314:
0314: 8A 46 1E                  MOV     AL,(BP+xRetCode)        ;return code in AL
0317: C3                        RET
0318:
0318:                   romIo1err
0318: E8 ****                   CALL    setUpCancel             ;cancel setup socket
031B:                   romIo2err
031B:                   romIo3err
031B:                   romIo4err
031B: B0 FF                     MOV     AL,0FFH                 ;abort return code
031D: C3                        RET
031E:
031E:                   ;*****************************************************************
031E:                   ;
031E:                   ; ROM_IO_2 -- interrupt service routine for general purpose
031E:                   ;               xmit data to server
031E:                   ;
031E:                   ;*****************************************************************
031E:                   romIo2
031E:                           ifFlatCable romIo2Err
0326: E8 ****                   CALL    omPutData               ;copy users data to xporter
0329: E8 ****                   CALL    setUpSrv                ;xmit cmd to server
032C: 73**                      JNC     $10                     ;all is ok
032E:
032E: B0 FF                     MOV     AL,0FFH                 ;set error return
0330: C3                        RET
0331:                   $10
0331: B0 00                     MOV     AL,0                    ;set normal return
0333: C3                        RET
0334:
0334:                   ;*****************************************************************
0334:                   ;
0334:                   ; ROM_IO_4 -- interrupt service routine for finding a disk server
0334:                   ;               by broadcasting a cmd
0334:                   ;
0334:                   ;*****************************************************************
0334:                   romIo4
0334:                           ifFlatCable romIo4Err
033C: C6 46 0D FF               MOVBIM  (BP+server),brdCast     ;broadcast the message
0340: E8 ****                   CALL    setUpBr                 ;set up broadcast receive
0343:                   $05
0343: E8 ****                   CALL    setUpB                  ;xmit the broadcast message
0346: E8 ****                   CALL    omGetRslt               ;wait for result
0349: 73**                      JNC     $10                     ;didnt time out
034B:
034B: FE 4E 0F                  DECMB   (BP+retrys)             ;try again?
034E: 75F3                      JNZ     $05                     ; yes
0350: E8 ****                   CALL    setUpCancel             ;cancel socket
0353: B0 FF                     MOV     AL,0FFH                 ;error, not found
0355: C3                        RET
0356:                   $10
0356: 32 C0                     XOR     AL,AL                   ;good result
0358: 8A 66 19                  MOV     AH,(BP+xHost)           ;address of disk server
035B: C3                        RET
```

```
035C:
035C:                   ;*********************************************************************
035C:                   ;
035C:                   ; ROM_IO_3 -- interrupt service routine for general purpose
035C:                   ;               xmit/recv data to server
035C:                   ;
035C:                   ;*********************************************************************
035C:                   romIo3
035C:                           ifFlatCable romIc3err
0364: E8 ****                   CALL    omPutData               ;copy users data to xporter
0367: E8 ****                   CALL    setUpRSrv               ;set up socket
036A: E8 ****                   CALL    setUpSrv                ;xmit to server
036D: 72**                      JC      $20                     ;error, abort
036F:                   $05
036F: B8 10 00                  MOV     AX,srvRResult-omvectors ;result vector add in xporter
0372: E8 ****                   CALL    omSetCntr
0375:                   $10
0375: BA 49 02                  MOV     DX,omHeader
0378: EC                        IN      AL,DX
0379:
0379: A9 80 00                  TEST    AX,80H                  ;is anything in yet?
037C: 74**                      JZ      $20                     ;yes
037E:
037E: E8 ****                   CALL    chkTime                 ;have we timed out
0381: 72**                      JC      romIo3exit              ; yes
0383: 73F0                      JNC     $10                     ;wait some more
0385:                   $20
0385: 80 7E 0D FF               CMPBIM  (BP+server),brdCast     ;was this a broadcast message
0389: 74**                      JE      romIo3end               ;yes, result from any server is valid
038B:
038B: B8 11 00                  MOV     AX,srvRHost-omvectors   ;check who message is from
038E: E8 ****                   CALL    omSetCntr
0391: BA 4B 02                  MOV     DX,omData
0394: EC                        IN      AL,DX
0395: 3A 46 0D                  CMP     AL,(BP+server)          ;was it from correct server?
0398: 74**                      JE      romIo3end               ;yes
039A:
039A: E8 ****                   CALL    setUpRSrv               ;resetup socket
039D: EBD0                      JMP     $05                     ;and wait again
039F:                   romIo3end
039F: E8 ****                   CALL    omGetData               ;copy data to user area
03A2:
03A2: B8 12 00                  MOV     AX,srvRLen-omvectors    ;get len of message
03A5: E8 ****                   CALL    omSetCntr
03A8: BA 4B 02                  MOV     DX,omData               ;in CX
03AB: EC                        IN      AL,DX
03AC: 8A E8                     MOV     CH,AL
03AE: EC                        IN      AL,DX
03AF: 8A C8                     MOV     CL,AL
03B1:
03B1: B0 00                     MOV     AL,0                    ;set normal return
03B3: C3                        RET
03B4:                   romIo3exit
03B4: E8 ****                   CALL    setUpScan               ;cancel socket
03B7: B0 FF                     MOV     AL,0FFH                 ;set error return
```

```
03B9! C3                            RET
03BA!
03BA!                       ;*****************************************************************************
03BA!                       ;
03BA!                       ; FLAT_IO_1 -- xmit/recv to drive via flat cable interface
03BA!                       ;
03BA!                       ;*****************************************************************************
03BA!                       flatIo1
03BA! E8 ****                       CALL    getStopTime                 ;this is for time out
03BD!                       $00
03BD! BA EF 02                      MOV     DX,flatStatus
03C0! FA                            CLI                                 ;disAllow interrupts
03C1! EC                            IN      AL,DX                       ;getStatus byte
03C2! A8 01                         TEST    AL,flatReady                ;is drive ready
03C4! 74**                          JZ      $02                         ;yes continue
03C6! FB                            STI                                 ;allow interrupts
03C7! E8 ****                       CALL    chkTime                     ;have we timed out
03CA! 73F1                          JNC     $00                         ;no wait again
03CC!
03CC! B0 FF                         MOV     AL,0FFH                     ;set abort error--time out
03CE! C3                            RET                                 ;and return
03CF!                       $02
03CF! 8B 76 00                      MOV     SI,(BP+sendBufAdd)          ;address of users buffer
03D2! 8E 5E 04                      MOV     DS,(BP+sendSeg)             ;segment of   "    "
03D5! 8B 4E 08                      MOV     CX,(BP+sendLen)             ;number of bytes to send
03D8! E8 ****                       CALL    $10                         ;send the bytes
03DB!
03DB! 80 7E 0C 05                   CMPBIM  (BP+cmd),5                  ;is this a write command
03DF! 75**                          JNE     flatTurn                    ;no
03E1!
03E1! 8B 76 02                      MOV     SI,(BP+recvBufAdd)          ;data portion of write cmd
03E4! 8E 5E 06                      MOV     DS,(BP+recvSeg)
03E7! 8B 4E 0A                      MOV     CX,(BP+recvLen)
03EA! E8 ****                       CALL    $05
03ED! E9 ****                       JMP     flatTurn
03F0!
03F0!                       ;
03F0!                       ; send the command to the drive
03F0!                       ;
03F0!                       $05
03F0! EC                            IN      AL,DX                       ;status and sending byte
03F1! A8 01                         TEST    AL,flatReady                ;so mux won't give up if we are
03F3! 75FB                          JNZ     $05                         ;interrupted
03F5!                       $10
03F5! AC                            LODSB                               ;AL=data byte
03F6! 4A                            DEC     DX                          ;DX=data port #
03F7! EE                            OUT     DX,AL                       ;sent to drive
03F8! FB                            STI                                 ;enable interrupts
03F9! 42                            INC     DX                          ;DX=status port #
03FA! E2F4                          LOOP    $05                         ;loop til byte count=0
03FC! C3                            RET
03FD!                       ;
03FD!                       ; wait for interface to turn around drive --> host (avoid glitches)
03FD!                       ;
03FD!
```

```
03FD:                           flatTurn
03FD: E8 ****                           CALL    getStopTime
0400: B9 0F 00                          MOV     CX,15.
0403:                           $10
0403: 49                                DEC     CX
0404: 75FD                              JNZ     $10
0406:                           $15
0406: E8 ****                           CALL    chkTime                 ;have we timed out yet
0409: 73**                              JNC     $17                     ;no
040B:
040B: B0 FF                             MOV     AL,0FFH                 ;indicate timeout
040D: C3                                RET
040E:                           $17
040E: BA EF 02                          MOV     DX,flatStatus
0411: EC                                IN      AL,DX                   ;get status
0412: A8 02                             TEST    AL,flatBusDir
0414: 75F0                              JNZ     $15                     ;loop until 'drive to host'
0416: A8 01                             TEST    AL,flatReady
0418: 75EC                              JNZ     $15                     ;check ready
041A: B9 0F 00                          MOV     CX,15.                  ;delay to avoid glitches
041D:                           $20
041D: 49                                DEC     CX
041E: 75FD                              JNZ     $20
0420:
0420: EC                                IN      AL,DX                   ;get status
0421: A8 02                             TEST    AL,flatBusDir           ;is it drive to host
0423: 75E1                              JNZ     $15                     ;no wait
0425: A8 01                             TEST    AL,flatReady            ;is it ready
0427: 75DD                              JNZ     $15                     ;no
0429:                           ;
0429:                           ; load info about where user wants data
0429:                           ;
0429:
0429: 8B 7E 02                          MOV     DI,(BP+recvBufAdd)
042C: 8E 46 06                          MOV     ES,(BP+recvSeg)
042F: 8B 4E 0A                          MOV     CX,(BP+recvLen)
0432: 33 DB                             XOR     BX,BX                   ;count of bytes received
0434:
0434: 80 7E 0C 05                       CMPBIM  (BP+cmd),5              ;is this a write command
0438: 75**                              JNE     $30                     ;no
043A: 33 C9                             XOR     CX,CX                   ;no bytes back on write
043C:
043C:                           ;
043C:                           ; command has been sent to the drive, now receive from drive
043C:                           ;
043C:                           $30
043C: EC                                IN      AL,DX
043D: A8 01                             TEST    AL,flatReady
043F: 75FB                              JNZ     $30                     ;check ready
0441: EC                                IN      AL,DX
0442: A8 02                             TEST    AL,flatBusDir           ;check direction
0444: 75**                              JNZ     $50                     ;may be done if bus turned around
0446: 4A                                DEC     DX                      ;DX=data port #
0447: EC                                IN      AL,DX
0448: 42                                INC     DX                      ;DX=status port #
```

```
0449: 0B DB                         OR      BX,BX                       ;is this 1st byte?
044B: 75**                          JNZ     $32                         ;no
044D: 88 46 1E                      MOV     (BP+xRetCode),AL            ;save return code
0450: 74**                          JZ      $35                         ;always
0452:                       $32
0452: E3**                          JCXZ    $35                         ;user doesnt want anything more back
0454: AA                            STOSB                               ;save byte
0455: 49                            DEC     CX                          ;users byte count
0456:                       $35
0456: 43                            INC     BX                          ;byte count
0457: EBE3                          JMP     $30                         ;loop til done
0459:                       ;
0459:                       ;check for transfer completion
0459:                       ;
0459: EC                    $50     IN      AL,DX
045A: A8 01                         TEST    AL,flatReady                ;check for drive ready
045C: 75DE                          JNZ     $30
045E: A8 02                         TEST    AL,flatBusDir               ;check direction
0460: 74DA                          JZ      $30                         ;continue if still 'drive to host'
0462:
0462:                       ;
0462:                       ;all done put byte count in CX return code put in AL
0462:                       ;
0462: 8B CB                         MOV     CX,BX
0464: 8A 46 1E                      MOV     AL,(BP+xRetCode)
0467: C3                            RET
0468:
0468:                       ;*****************************************************************
0468:                       ;OMPUTDATA
0468:                       ;       routine to transfer data from host mem to xporter buffer
0468:                       ;*****************************************************************
0468:                       omPutData
0468: 8B 4E 08                      MOV     CX,(BP+sendLen)             ;bytes to send to drive
046B: B8 95 00                      MOV     AX,sendBuf-omVectors
046E: E8 ****                       CALL    omSetCntr
0471:
0471: 8B 76 00                      MOV     SI,(BP+sendBufA)            ;address of user data
0474: 8E 5E 04                      MOV     DS,(BP+sendSeg)
0477:
0477: BA 4B 02                      MOV     DX,omData                   ;read/write xporter port
047A:                       $20
047A: AC                            LODSB
047B: EE                            OUT     AL,DX
047C: E2FC                          LOOP    $20
047E:
047E: 80 7E 0C 05                   CMPBIM  (BP+cmd),5                  ;is this write cmd
0482: 75**                          JNE     $40
0484:
0484: 8B 76 02                      MOV     SI,(BP+recvBufA)            ;address of data for write
0487: 8E 5E 06                      MOV     DS,(BP+recvSeg)
048A: 8B 4E 0A                      MOV     CX,(BP+recvLen)             ;length of data portion of write
048D:                       $30
048D: AC                            LODSB
048E: EE                            OUT     AL,DX
048F: E2FC                          LOOP    $30
```

```
0491:                           $40
0491: B8 0B 00                          MOV     AX,sHdrToLen-omVectors  ;fix up length in cmd vectors
0494: 8B 4E 08                          MOV     CX,(BP+sendLen)
0497: 80 7E 0C 05                       CMPBIM  (BP+cmd),5              ;is this write cmd
049B: 75**                              JNE     $45                     ;no
049D: 03 4E 0A                          ADD     CX,(BP+recvLen)         ;add in data length
04A0:                           $45
04A0: E8 ****                           CALL    $50
04A3:
04A3: B8 5E 00                          MOV     AX,srvCmdLen-omVectors
04A6: E8 ****                           CALL    $50
04A9:
04A9: B8 52 00                          MOV     AX,restLen-omVectors
04AC: 83 E9 04                          SUB     CX,4
04AF:                           $50
04AF: E8 ****                           CALL    omSetCntr               ;set xporter address
04B2: 8A C5                             MOV     AL,CH                   ;most sig byte
04B4: EE                                OUT     AL,DX
04B5: 8A C1                             MOV     AL,CL
04B7: EE                                OUT     AL,DX
04B8: C3                                RET
04B9:
04B9:
04B9:                           ;*************************************************************************
04B9:                           ;OMGETDATA
04B9:                           ;       routine to move data from transporter buffer to host mem
04B9:                           ;*************************************************************************
04B9:                           omGetData
04B9: 80 7E 0C 05                       CMPBIM  (BP+cmd),5              ;is this write command
04BD: 74**                              JE      $10                     ;yes dont send any bytes back
04BF:
04BF: B8 95 00                          MOV     AX,recvBuf-omVectors    ;addr of transporter buffer
04C2: E8 ****                           CALL    omSetCntr
04C5: BA 4B 02                          MOV     DX,omData               ;DX=data port #
04C8:
04C8: 8B 7E 02                          MOV     DI,(BP+recvBufA)        ;address of user buffer
04CB: 8E 46 06                          MOV     ES,(BP+recvSeg)
04CE:
04CE: 8B 4E 0A                          MOV     CX,(BP+recvLen)
04D1: 0B C9                             OR      CX,CX                   ;does user want anything?
04D3: 74**                              JZ      $10                     ; no, exit
04D5:
04D5: EC                        $5      IN      AL,DX
04D6: AA                                STOSB
04D7: E2FC                              LOOP    $5
04D9: C3                        $10     RET
04DA:
04DA:
04DA:                           ;*************************************************************************
04DA:                           ;OMSETCNTR
04DA:                           ;       routine to set the buffer pointer counter on transporter board
04DA:                           ;
04DA:                           ;input
04DA:                           ;       AX = new counter value
```

```
04DA:                   ;*************************************************************************
04DA:
04DA:                   omSetCntr
04DA: 52                        PUSH    DX
04DB:
04DB: BA 4A 02                  MOV     DX,omLoCntr             ;DX=port # to set low byte of cntr
04DE: EE                        OUT     DX,AL
04DF:
04DF: BA 48 02                  MOV     DX,omHiCntr
04E2: 8A C4                     MOV     AL,AH
04E4: EE                        OUT     DX,AL                   ;set hi byte
04E5:
04E5: 5A                        POP     DX
04E6: C3                        RET
04E7:
04E7:                   ;*************************************************************************
04E7:                   ;
04E7:                   ; FIX_SERVER_ADD -- puts address of server cmd vector
04E7:                   ;
04E7:                   ;*************************************************************************
04E7:                   fixServerAdd
04E7: 8B 46 13                  MOV     AX,(BP+cmdAddr) ;xporter address of cmd vector
04EA: 05 0B 00                  ADD     AX,11.          ;offset for server address
04ED: EB EAFF                   CALL    omSetCntr
04F0: BA 48 02                  MOV     DX,omData
04F3: 8A 46 0D                  MOV     AL,(BP+server)
04F6: EE                        OUT     AL,DX           ;put disk server add in cmd
04F7: C3                        RET
04F8:
04F8:                   ;*************************************************************************
04F8:                   ;OMSETUP
04F8:                   ;       routine to setup cmd vectors in transporter buffer
04F8:                   ;
04F8:                   ; input
04F8:                   ;       cmd vector strobe addr is stored in CMD_ADDR
04F8:                   ;*************************************************************************
04F8:                   omSetUp
04F8: 8A 46 0F                  MOV     AL,(BP+retrys)
04FB: 32 E4                     XOR     AH,AH
04FD: 50                        PUSH    AX              ;save retrys on stack
04FE:                   omSetUp1
04FE: BA 4B 02                  MOV     DX,omData
0501: 8B 46 13                  MOV     AX,(BP+cmdAddr) ;get strobe addr
0504: 50                        PUSH    AX              ;save it
0505: E8 D2FF                   CALL    omSetCntr       ;address of transporter cmd
0508: EC                        IN      AL,DX           ;get transporter cmd byte
0509: 88 46 12                  MOV     (BP+cmdByte),AL ;save it in variable
050C: 58                        POP     AX              ;restore cmd vector address
050D:
050D: 40                        INC     AX
050E: 40                        INC     AX              ;AX=pointer to result vector addr
050F: E8 C8FF                   CALL    omSetCntr       ;set xporter counter
0512:
0512: EC                        IN      AL,DX
0513: 8A E0                     MOV     AH,AL           ;hi byte of result vect addr
```

```
0515: EC                              IN      AL,DX
0516: 89 46 15                        MOV     (BP+rsltAddr),AX ;save it
0519: E8 BEFF                        CALL    omSetCntr        ;set xporter counter to reslt vect
051C: B0 FF                           MOV     AL,0FFH          ;initialize return code
051E:
051E: BA 48 02                        MOV     DX,omData        ;data output port #
0521: EE                              OUT     DX,AL
0522:
0522: E8 ****                         CALL    omniStrobe       ;send command vector to transporter
0525:
0525: 8B 46 15                        MOV     AX,(BP+rsltAddr) ;addr of result vect
0528: E8 AFFF                         CALL    omSetCntr        ;set xporter counter
052B: BA 49 02                        MOV     DX,omHeader      ;data input without auto-inc
052E:
052E: EC                      $0      IN      AL,DX
052F: 3C FF                           CMP     AL,0FFH          ;inspect transpoter status return code
0531: 74FB                            JE      $0               ;loop waiting if not changed
0533:
0533: EC                              IN      AL,DX            ;ported memory problems -- must do 2nd read
0534: 84 46 17                        TEST    AL,(BP+mask)     ;is correct bit cleared
0537: 74**                            JZ      omSetUpOk        ;everything OK, exit
0539:
0539: 80 7E 12 40                     CMPBIM  (BP+cmdByte),xmitCmd
053D: 75**                            JNE     $10              ;error exit
053F: 80 7E 0F FF                     CMPBIM  (BP+retrys),0FFH ;is this continious retrys?
0543: 74B9                            JE      omSetUp1         ; yes
0545: FE 4E 0F                        DECMB   (BP+retrys)      ; any more retrys
0548: 75B4                            JNZ     omSetUp1         ; yes
054A:                         $10
054A: F9                              STC                      ;set error flag
054B: 72**                            JC      omSetUpExit      ; forced branch
054D:                         omSetUpOk
054D: 80 7E 12 40                     CMPBIM  (BP+cmdByte),xmitCmd
0551: 75**                            JNE     $10
0553: 80 7E 0D FF                     CMPBIM  (BP+server),brdcast
0557: 75**                            JNE     $10              ;retry count for broadcast is successfull tries
0559: 80 7E 0C 02                     CMPBIM  (BP+cmd),2       ;is this a transmit only command
055D: 75**                            JNE     $10
055F: FE 4E 0F                        DECMB   (BP+retrys)      ;are there any more retrys left?
0562: 759A                            JNZ     omSetUp1         ;yes do it again
0564:                         $10
0564: F8                              CLC                      ;carry is error flag
0565:                         omSetUpExit
0565: 58                              POP     AX
0566: 88 46 0F                        MOV     (BP+retrys),AL   ;restore retry count
0569: C3                              RET
056A:
056A:                         ;*****************************************************************
056A:                         ;
056A:                         ;OMNI STROBE -- strobe transported with add of cmd vector
056A:                         ;
056A:                         ;*****************************************************************
056A:                         omniStrobe
056A: 32 E4                           XOR     AH,AH
056C: E8 ****                         CALL    $10                ;first msb is 0
```

```
056F: BA 66 14                  MOV     AH,(BP+cmdAddr+1) ;next msb
0572: E8 ****                   CALL    $10
0575: 8A 66 13                  MOV     AH,(BP+cmdAddr)   ;fall thru
0578: E8 ****                   CALL    $10
057B:
057B: E9 ****                   JMP     getStopTime     ;calc time out time
057E:
057E:                   $10
057E: BA 48 02                  MOV     DX,omStatus     ;transporter status port #
0581: EC                $20     IN      AL,DX
0582: A8 80                     TEST    AL,80H          ;is transporter ready
0584: 74FB                      JZ      $20             ;loop til transporter ready
0586:
0586: BA 49 02                  MOV     DX,omniPort     ;strobe port #
0589: 8A C4                     MOV     AL,AH           ;get output byte to AL
058B: EE                        OUT     DX,AL           ;send the command byte
058C: C3                        RET
058D:
058D:                   ;*****************************************************************
058D:                   ;
058D:                   ;GET_STOP_TIME -- calculates time out time from user params
058D:                   ;
058D:                   ;*****************************************************************
058D:                   getStopTime
058D: 32 E4                     XOR     AH,AH           ;read time of day clock
058F: CD 1A                     INT     1AH             ;low count is in DX
0591: 8A 46 0E                  MOV     AL,(BP+waitTime)
0594: 32 E4                     XOR     AH,AH
0596: B1 04                     MOV     CL,4            ;multiply by 16
0598: D3 E0                     SHL     AX,CL
059A:
059A: 03 D0                     ADD     DX,AX
059C: 89 56 10                  MOV     (BP+stopTime),DX  ;time to give up at
059F: C3                        RET
05A0:                   ;*****************************************************************
05A0:                   ;
05A0:                   ; SET_UP_RECEIVE
05A0:                   ;
05A0:                   ;*****************************************************************
05A0:
05A0:                   setUpReceive
05A0: E8 ****                   CALL    setUpCancel
05A3: C7 46 13 33 00            MOV     (BP+cmdAddr),rCmd-omVectors
05A8: C6 46 17 01               MOVBIM  (BP+mask),01H   ;wait for lsb to clear
05AC: E9 49FF                   JMP     omSetup
05AF:
05AF:                   ;*****************************************************************
05AF:                   ;
05AF:                   ; SET_UP_SEND
05AF:                   ;
05AF:                   ;*****************************************************************
05AF:                   setUpSend
05AF: C7 46 13 3E 00            MOV     (BP+cmdAddr),sCmd-omVectors
05B4: C6 46 17 80               MOVBIM  (BP+mask),80H   ;wait for msb to clear
```

```
05B8: E8 2CFF                       CALL    fixServerAdd    ;put server add in cmd vector
05BB: E9 3AFF                       JMP     omSetup
05BE:
05BE:                       ;*****************************************************************
05BE:                       ;
05BE:                       ; SET_UP_GO
05BE:                       ;
05BE:                       ;*****************************************************************
05BE:                       setUpGo
05BE: E8 ****                       CALL    setUpCancel
05C1: C7 46 13 1E 00                MOV     (BP+cmdAddr),goCmd-omVectors
05C6: C6 46 17 01                   MOVBIM  (BP+mask),01H   ;wait for lsb to clear
05CA: E9 28FF                       JMP     omSetup
05CD:
05CD:                       ;*****************************************************************
05CD:                       ;
05CD:                       ; SET_UP_REST
05CD:                       ;
05CD:                       ;*****************************************************************
05CD:                       setupRest
05CD: C7 46 13 4A 00                MOV     (BP+cmdAddr),restCmd-omVectors
05D2: C6 46 17 80                   MOVBIM  (BP+mask),80H   ;wait for msb to clear
05D6: E8 0EFF                       CALL    fixServerAdd    ;put server add in cmd vector
05D9: E9 1CFF                       JMP     omSetUp
05DC:
05DC:                       ;*****************************************************************
05DC:                       ;
05DC:                       ; SET_UP_SRV
05DC:                       ;
05DC:                       ;*****************************************************************
05DC:                       setUpSrv
05DC: C7 46 13 56 00                MOV     (BP+cmdAddr),srvCmd-omVectors
05E1: C6 46 17 80                   MOVBIM  (BP+mask),80H   ;wait for msb to clear
05E5: E8 FFFE                       CALL    fixServerAdd    ;put server add in cmd vector
05E8: E9 0DFF                       JMP     omSetUp
05EB:
05EB:                       ;*****************************************************************
05EB:                       ;
05EB:                       ; SET_UP_R_SRV
05EB:                       ;
05EB:                       ;*****************************************************************
05EB:                       setUpRSrv
05EB: E8 ****                       CALL    setUpSCancel
05EE: C7 46 13 62 00                MOV     (BP+cmdAddr),srvRCmd-omVectors
05F3: C6 46 17 01                   MOVBIM  (BP+mask),01H               ;wait for lsb to clear
05F7: E9 FEFE                       JMP     omSetUp
05FA:
05FA:                       ;*****************************************************************
05FA:                       ;
05FA:                       ; SET_UP_CAN
05FA:                       ;
05FA:                       ;*****************************************************************
```

```
05FA:                           setUpCan
05FA: C7 46 13 29 00                    MOV     (BP+cmdAddr),canRecvCmd-omVectors
05FF: C6 46 17 80                       MOVBIM  (BP+mask),80H                   ;wait for msb to clear
0603: E9 F2FE                           JMP     omSetUp
0606:
0606:                           ;*******************************************************************
0606:                           ;
0606:                           ; SET_UP_S_CAN
0606:                           ;
0606:                           ;*******************************************************************
0606:                           setUpSCan
0606: C7 46 13 2E 00                    MOV     (BP+cmdAddr),canSrvCmd-omVectors
060B: C6 46 17 80                       MOVBIM  (BP+mask),80H                   ;wait for msb to clear
060F: E9 E6FE                           JMP     omSetUp
0612:
0612:                           ;*******************************************************************
0612:                           ;
0612:                           ; SET_UP_WHO
0612:                           ;
0612:                           ;*******************************************************************
0612:                           setUpWho
0612: C7 46 13 6D 00                    MOV     (BP+cmdAddr),whoCmd-omVectors
0617: C6 46 17 80                       MOVBIM  (BP+mask),80H
061B: E9 DAFE                           JMP     omSetUp
061E:
061E:                           ;*******************************************************************
061E:                           ;
061E:                           ; SET_UP_ECHO
061E:                           ;
061E:                           ;*******************************************************************
061E:                           setUpEcho
061E: C7 46 13 71 00                    MOV     (BP+cmdAddr),echoCmd-omVectors
0623: C6 46 17 01                       MOVBIM  (BP+mask),01H                   ;wait for lsb to clear
0627: E9 CEFE                           JMP     omSetUp
062A:
062A:                           ;*******************************************************************
062A:                           ;
062A:                           ; SET_UP_B
062A:                           ;
062A:                           ;*******************************************************************
062A:                           setUpB
062A: C7 46 13 7E 00                    MOV     (BP+cmdAddr),bCmd-omVectors
062F: C6 46 17 80                       MOVBIM  (BP+mask),80H
0633: E9 C2FE                           JMP     omSetUp
0636:
0636:                           ;*******************************************************************
0636:                           ;
0636:                           ; SET_UP_BR
0636:                           ;
0636:                           ;*******************************************************************
0636:                           setUpBr
0636: E8 C1FF                           CALL    setUpCancel
```

```
0639: C7 46 13 8A 00            MOV     (BP+cmdAddr),brCmd-omVectors
063E: C6 46 17 01               MOVBIM  (BP+mask),01H               ;wait for lsb to clear
0642: E9 B3FE                   JMP     omSetUp
0645:
0645:                   ;*******************************************************
0645:                   ;
0645:                   ; OM_GET_RESULT -- waits for disk server to finish, copys 1st
0645:                   ;                  seven bytes to ram
0645:                   ;
0645:                   ;*******************************************************
0645:                   omGetRslt
0645: B8 00 00                  MOV     AX,rResult-omVectors
0648: E8 8FFE                   CALL    omSetCntr
064B:
064B: BA 49 02          $10     MOV     DX,omHeader
064E: EC                        IN      AL,DX               ;check for operation complete
064F: A8 80                     TEST    AL,80H
0651: 74**                      JZ      $15
0653: E8 ****                   CALL    chkTime             ;have we timed out yet?
0656: 73F3                      JNC     $10                 ; no
0658: C3                        RET                         ;return with error flag
0659:                   $15
0659: 33 F6                     XOR     SI,SI
065B:
065B: B9 07 00                  MOV     CX,7
065E: BA 48 02                  MOV     DX,omData
0661: EC                $20     IN      AL,DX
0662: 89 42 18                  MOV     (SI)(BP+xResult),AX
0665: 46                        INC     SI
0666: E2F9                      LOOP    $20
0668: F8                        CLC                         ;return with no errors
0669: C3                        RET
066A:                   ;*******************************************************
066A:                   ;
066A:                   ;CHK_TIME -- checks the timer to see if it is time to abort
066A:                   ;
066A:                   ;*******************************************************
066A:                   chkTime
066A: 80 7E 0E 00               CMPBIM  (BP+waitTime),0         ;0 = do not time out
066E: 74**                      JE      chkTimeOk               ;dont test if 0
0670:
0670: B4 00                     MOV     AH,0                    ;time of day call
0672: CD 1A                     INT     1AH
0674: 0A C0                     OR      AL,AL                   ;has clock wrapped around to zero
0676: 74**                      JZ      $10                     ;no
0678:                           ;
0678:                           ; clock has passed 24 hours -- to make life simple stop time is
0678:                           ;       re-calculated
0678:                           ;
0678: 8A 46 0E                  MOV     AL,(BP+waitTime)
067B: 32 E4                     XOR     AH,AH
067D: B1 04                     MOV     CL,4                ;multiply by 16
067F: D3 E0                     SHL     AX,CL
0681:
0681: 03 D0                     ADD     DX,AX
```

```
0683: 89 56 10                  MOV     (BP+stopTime),DX  ;time to give up at
0686: E9 ****                   JMP     chkTimeOk
0689:                   $10
0689: 3B 56 10                  CMP     DX,(BP+stopTime)        ;is it time to abort
068C: 72**                      JB      chkTimeOk               ; no
068E:
068E: F9                        STC                             ;set abort flag
068F: C3                        RET
0690:                   chkTimeOk
0690: F8                        CLC                             ;not timed out flag
0691: C3                        RET
0692:
0692:                   ;***************************************************
0692:                   ;
0692:                   ; HELLO
0692:                   ;
0692:                   ;***************************************************
0692:                   hello
0692: 32 E4                     XOR     AH,AH                   ;video I/O call
0694: B0 02                     MOV     AL,2                    ;80x25 black and white mode
0696: CD 10                     INT     10H
0698: BE 0C00                   LEA     SI,helloMsg
069B: E9 ****                   JMP     printStr                ;print hello message and return
069E:
069E:                   ;***************************************************
069E:                   ;
069E:                   ; LOAD_BOOT -- loads the IBM constellation II boot program into
069E:                   ;              ram and transfers control to it
069E:                   ;
069E:                   ;***************************************************
069E:                   loadBoot
069E: CD 12                     INT     12H                     ;memory size check
06A0: 3D 40 00                  CMP     AX,40H                  ;is there at least 64k
06A3: 73**                      JNB     $05
06A5: E9 ****                   JMP     need64err               ;no
06A8:                   $05
06A8: BB 00 00                  MOV     BX,0
06AB: 8E DB                     MOV     DS,BX
06AD: 8E C3                     MOV     ES,BX
06AF:
06AF: FA                        CLI                             ;disable while fixing stack
06B0: 8E D3                     MOV     SS,BX                   ;make sure stack doesnt clobber us
06B2: BC FE FF                  MOV     SP,stackAdd
06B5: FB                        STI                             ;enable interrupts
06B6:
06B6: E8 D9FF                   CALL    hello                   ;hello msg
06B9:
06B9:                           ifFlatCable loadBtstart         ;go directly to disk
06C1:
06C1: B4 00                     MOV     AH,0                    ;identify entry point
06C3: 9A 0600 00DF              CALLL   ioEntry,romSeg          ;ROM I/O entry point
06C8:                                                           ;AH=workstation number
06C8: 80 FC FF                  CMP     AH,0FFH                 ;is there a duplicate xporter out there
06CB: 75**                      JNE     $10
06CD: E9 ****                   JMP     xportErr
```

```
06D0:                           $10
06D0: B4 04                             MOV     AH,4                    ;find disk server request
06D2: B3 08                             MOV     BL,8                    ;wait 6.88 seconds
06D4: B7 02                             MOV     BH,2                    ;send message twice
06D6:
06D6: 9A 0600 00DF                      CALLL   ioEntry,romSeg          ;find any disk server
06DB: 3C 00                             CMP     AL,0                    ;did we find a disk server
06DD: 74**                              JE      loadBtstart             ;yes
06DF: E9 ****                           JMP     noSrvErr                ;no time to give up
06E2:                           ;
06E2:                           ; we have found a disk server -- now we must read in the boot program
06E2:                           ;
06E2:                           loadBtstart
06E2: 33 DB                             XOR     BX,BX
06E4: 88 A7 0070                        MOV     (BX+bootServer),AH      ;save server
06E8: C6 87 0170 44                     MOVBIM  (BX+bootSource+bootCmd),044H    ;const II read boot block
06ED: C6 87 0270 09                     MOVBIM  (BX+bootSource+bootType),ibmType    ;computer type
06F2: C6 87 0370 00                     MOVBIM  (BX+bootSource+boctBlock),0     ;start with block zero
06F7:
06F7: C7 87 0470 00 7C                  MOV     (BX+loadAddReg),loadAdd         ;start loading here
06FD:                           loadBt01
06FD: BB 00 00                          MOV     BX,0
0700: BE 01 70                          MOV     SI,bootSource
0703: B9 03 00                          MOV     CX,3                    ;send 3 bytes
0706:
0706: 8B BF 0470                        MOV     DI,(BX+loadAddReg)
070A: BA 00 02                          MOV     DX,512.                 ;read one block
070D:
070D: 8A 87 0070                        MOV     AL,(BX+bootServer)      ;disk server address
0711:
0711: B3 08                             MOV     BL,8                    ;wait for disk server 6.88 seconds
0713: B7 02                             MOV     BH,2                    ;disk server is there (retrys)
0715:
0715: B4 01                             MOV     AH,1                    ;send drive cmd
0717: 9A 0600 00DF                      CALLL   ioEntry,romSeg          ;read one block
071C: 3C 00                             CMP     AL,0                    ;was read ok?
071E: 75**                              JNE     loadBtErr               ;no, we have a problem
0720:
0720: BB 00 00                          MOV     BX,0
0723: 81 BF 0470 00 7C                  CMP     (BX+loadAddReg),loadAdd ;is this the first read?
0729: 75**                              JNE     $10                     ;no
072B:
072B: 8B 87 007C                        MOV     AX,(BX+loadAdd)         ;get 1st word of 1st block
072F: 8B C8                             MOV     CX,AX                   ;save it
0731: D0 ED                             SHR     CH,1                    ;CH = number of blocks
0733: 25 FF 01                          AND     AX,1FFH                 ;is length multiple of 512
0736: 74**                              JZ      $05                     ; yes
0738: FE C5                             INC     CH                      ;add 1 to block count
073A:                           $05
073A: 88 AF 007C                        MOV     (BX+loadAdd),CH         ;store number of blocks in first byte
073E:                           $10
073E: FE 8F 007C                        DECMB   (BX+loadAdd)            ;decrement block count
0742: 74**                              JZ      loadBtEnd               ;all done
0744: FE 87 0370                        INCMB   (BX+bootSource+bootBlock)
0748: 81 87 0470 00 02                  ADD     (BX+loadAddReg),200H    ;add to load address
```

```
074E:
074E: EBAD                          JMP     loadBtOi                ;read in next block
0750:                       loadBtend
0750: 8A 87 0070                    MOV     AL,(BX+bootServer)      ;jump into code with server add
0754:                                                                       ;in AL
0754: EA 027C 0000                  JMPL    loadAdd+2,0             ;jmp into code
0759:                       xportErr
0759: BE 8C00                       LEA     SI,dupXporterMsg
075C: E9 ****                       JMP     printMsg
075F:                       loadBtErr
075F: BE 7000                       LEA     SI,bootMsg              ;error message
0762: E9 ****                       JMP     printMsg
0765:                       noSrvErr
0765: BE BD00                       LEA     SI,noSrvMsg             ;no server on network
0768: E9 ****                       JMP     printMsg
076B:                       need64err
076B: BE E700                       LEA     SI,need64msg            ;need 64k message
076E: E9 ****                       JMP     printMsg
0771:                       printMsg
0771: E8 ****                       CALL    printStr
0774:                       $20
0774: EBFE                          JMP     $20                     ;loop forever
0776:                       printStr
0776: 2E AC                         LODSB   AL,CS:(SI)
0778: 0A C0                         OR      AL,AL                   ;is it the end
077A: 74**                          JE      $10                     ;yes
077C: B4 0E                         MOV     AH,14.                  ;TTY write command
077E: B7 00                         MOV     BH,0                    ;write page zero
0780: CD 10                         INT     10H                     ;call ibm rom i/o routines
0782: EBF2                          JMP     printStr
0784:                       $10
0784: C3                            RET
0785:
0785: 00 00 00 00 00 00 00          .ORG    7FEH                    ;end of Rom
07FE: 01                    interFace       .BYTE   cardType        ;identifys card type
07FF: 15                    versionMark     .BYTE   version
0800:                               .END
```

AB - Absolute LB - Label UD - Undefined MC - Macro
RF - Ref DF - Def PR - Proc FC - Func
PB - Public PV - Private CS - Consts

```
BCMD     LB 0179  BDATA    LB 0171  BOOTBLOC AB 0002  BOOTCMD  AB 0000  BOOTEND  AB 0003  BOOTLEN  AB 0003  BOOTMSG  LB 0070
BOOTSERV AB 7000  BOOTSOUR AB 7001  BOOTTYPE AB 0001  BRCMD    LB 0185  BRDCAST  AB 00FF  BRDSOCKE AB 0080  BUFFLEN  AB 0212
CALLENTR AB 0006  CALLTABL LB 0190  CANRECVC LB 0124  CANRESUL LB 0116  CANSRVCM LB 0129  CARDTYPE AB 0001  CHKTIME  LB 066A
CHKTIMED LB 0690  CMD      AB 000C  CMDADDR  AB 0013  CMDBYTE  AB 0012  COLDCONT LB 0069  COLDSTAR LB 0000  DUMMYINT LB 0009
DUMMYIRE LB 006F  DUPXPORT LB 008C  ECHOCMD  LB 016C  ECHOHOST LB 0170  ECHORESU LB 0118  FIXSERVE LB 04E7  FLATBUSD AB 0002
FLATCBL  AB 0001  FLATDATA AB 02EE  FLATIO1  LB 03BA  FLATREAD AB 0001  FLATSTAT AB 02EF  FLATTURN LB 03FD  FLIP     MC ----
FLIPREL  MC ----  GETSTOPT LB 058D  GOCMD    LB 0119  GODATA   LB 010F  GOLEN    AB 0002  GORESHOS LB 0112  GORESULT LB 0111
HELLO    LB 0692  HELLOMSG LB 000C  IBMROM   PR ----  IBMTYPE  AB 0009  IFFLATCA MC ----  IFOMNINE MC ----  IFUNDER5 MC ----
INTERFAC LB 07FE  IOENTRY  AB 0006  IOSTART  LB 0006  LOADADD  AB 7C00  LOADADDR AB 7004  LOADBOOT LB 069E  LOADBTO1 LB 06FD
LOADBTEN LB 0750  LOADBTER LB 075F  LOADBTST LB 06E2  MASK     AB 0017  NEED64ER LB 076B  NEED64MS LB 00E7  NOSRVERR LB 0765
NOSRVMSG LB 00BD  OMCMDSIZ AB 0095  OMDATA   AB 0248  OMGETDAT LB 0489  OMGETRSL LB 0645  OMHEADER AB 0249  OMHICNTR AB 0248
OMINIT   LB 01E9  OMINITEN LB 0242  OMLOCNTR AB 024A  OMNINET  AB 0000  OMNIPORT AB 0249  OMNISTRO LB 056A  OMPUTDAT LB 0468
OMRDYBIT AB 0080  OMSETCNT LB 04DA  OMSETUP  LB 04F8  OMSETUP1 LB 04FE  OMSETUPE LB 0565  OMSETUPO LB 054D  OMSTATUS AB 0248
OMVECTEN LB 0190  OMVECTOR LB 00FB  PRINTMSG LB 0771  PRINTSTR LB 0776  RCMD     LB 012E  RCMDDATA LB 0134  RCMDLEN  LB 0136
RCVSOCKE AB 00B0  RECVBUF  LB 0190  RECVBUFA AB 0002  RECVLEN  AB 000A  RECVSEG  AB 0006  RESTCMD  LB 0145  RESTDATA LB 014B
RESTLEN  LB 014D  RESTRESU LB 0115  RESTSOCK AB 0040  RETRYS   AB 000F  RHDRRETC LB 0101  RHDRRETL LB 00FF  ROMIO    LB 019C
ROMIO0   LB 0243  ROMIO1   LB 0281  ROMIO1ER LB 0318  ROMIO2   LB 031E  ROMIO2ER LB 031B  ROMIO3   LB 035C  ROMIO3EN LB 039F
ROMIO3ER LB 031B  ROMIO3EX LB 03B4  ROMIO4   LB 0334  ROMIO4ER LB 031B  ROMSEG   AB DF00  RRESHOST LB 00FC  RRESULT  LB 00FB
RSLTADDR AB 0015  SCMD     LB 0139  SCMDDATA LB 013F  SENDBUF  LB 0190  SENDBUFA AB 0000  SENDLEN  AB 0008  SENDSEG  AB 0004
SENDSOCK AB 00B0  SERVER   AB 000D  SETUPB   LB 062A  SETUPBR  LB 0636  SETUPCAN LB 05FA  SETUPECH LB 061E  SETUPGO  LB 05BE
SETUPREC LB 05A0  SETUPRES LB 05CD  SETUPRSR LB 05E8  SETUPSCA LB 0606  SETUPSEN LB 05AF  SETUPSRV LB 05DC  SETUPWHO LB 0612
SHDRRETL LB 0108  SHDRTOLE LB 0106  SRESULT  LB 0102  SRVCMD   LB 0151  SRVCMDDA LB 0157  SRVCMDLE LB 0159  SRVRCMD  LB 015D
SRVRCMDD LB 0163  SRVRCMDL LB 0165  SRVRESUL LB 010A  SRVRHOST LB 010C  SRVRLEN  LB 010D  SRVRRESU LB 010B  SRVSOCKE AB 0080
STACKADD AB FFFE  STOPTIME AB 0010  VERSION  AB 0015  VERSIONM LB 07FF  WAITTIME AB 000E  WARMCONT LB 0065  WARMSTAR LB 0003
WHOCMD   LB 0168  WHORESUL LB 0117  WORKEND  AB 001F  WORKSIZE AB 001F  XHOST    AB 0019  XMITCMD  AB 0040  XPORTERR LB 0759
XRESULT  AB 0018  XRETCODE AB 001E  XRETLEN  AB 001C
```

```
Assembly complete:      1754 lines
     0   errors flagged on this assembly
```