```
;
PAGE 60,132
;
;*****************************************************************************
;                       Filename is PCSINIT.ASM                                 |
;*****************************************************************************
; This file contains code and data for Emulator primary initialization        |
; described in Sections 9.2 and 9.3 of the Program Logic Description          |
; for the OMNINET PCShare Disk Server Emulator, Version 2.0, dated 7          |
; November 1983.  The code in this file is executed as an independent         |
; program, not linked to the Emulator.                                        |
;                                                                             |
; Primary initialization obtains the location of the Corvus utilities         |
; jump table by a device driver input call to the device PCSMHOOK (as         |
; do all the Corvus utilities).  The offset of the ECT is in the first        |
; word following the jump table.  To assure that EI-INIT is matched           |
; with the correct version of the Emulator, the EC.IDEN field of a            |
; dummy ECT allocated within EI-INIT is compared with the same field          |
; in the driver.  Initialization is abandoned if the two fields are           |
; not the same.                                                               |
;                                                                             |
; At completion of its other processing EI-INIT calls the warm start          |
; entry of the jump table to initialize the transporter and start the         |
; operation of the Emulator.                                                  |
;*****************************************************************************
;                                                                             |
; Version:  2.0                                                               |
;                                                                             |
; Last Update:  28 December 1983                                              |
;                                                                             |
; Written by:  R.B. Talmadge, Computer Technology Ltd                         |
;                                                                             |
; Updates:  None                                                              |
;                                                                             |
;*****************************************************************************
;
        ENDIF
;
;*****************************************************************************
;       Special Macros for this initialization program
;*****************************************************************************
;
;-----------------------------------------------------------------------------
;       Macro Name:  SHOW String,Rply,crlf                                    |
;                                                                             |
;       Function: Displays the character string located at the offset         |
;                 string in the data segment.  Requests reply wait for        |
;                 Rply characters if Rply is present.  Requests no            |
;                 CRLF sequence at end if crlf is present                     |
;-----------------------------------------------------------------------------
;
SHOW    MACRO   String,Rply,crlf
        MOV     DX,OFFSET String        ;;string location
        IFB     <Rply>
        MOV     CX,0            ;;Set reply count
```

```
                ELSE
                MOV     CX,Rply
                ENDIF
                IFNB    <crlf>
                MOV     CH,1            ;;set crlf suppression
                ENDIF
                CALL    ISMsg           ;;Do output etc
                ENDM
        ;
        ;--------------------------------------------------------------------------
        ;       Macro Name:  LEAVE                                                 |
        ;                                                                          |
        ;       Function: Ends program and returns control to DOS                  |
        ;--------------------------------------------------------------------------
        ;
        LEAVE   MACRO
                MOV     AH,4CH          ;;Thats the code
                INT     21H
                ENDM
        ;
        ;****************************************************************************
        ;
                START   <EI>,<TRACE>    ;Initialization structures
        ;
        ;****************************************************************************
        ;                       Segment Usage Definition
        ;****************************************************************************
        ;
0000    PCS     SEGMENT  PARA
                ASSUME  CS:PCS,DS:PCS,ES:PCS
        ;
        ;****************************************************************************
        ;                           Allocated data                                 |
        ;****************************************************************************
        ;
= 0000  Dta_Bgn EQU     THIS BYTE       ;Start of Data area
        ;--------------------------------------------------------------------------
        ;       Constants
        ;
= 0020  $VerRec EQU     32              ;User file record with date
= 0044  $FrmDte EQU     44H             ;Offset in record to date
= 0012  $FrmSrv EQU     12H             ;Offset in record to server name
= 0007  $Beep   EQU     7               ;Character to beep console
= 07BE  $1982   EQU     1982            ;Value date should have
        ;
        ;--------------------------------------------------------------------------
        ;       Dummy Emulator Communication Table.  Emulator key offsets
        ;       table location will be stored in fields DDRH,DDRHS and the
        ;       ECT offset within the Emulator is saved in field TckO
        ;
0000  50 43 53 48 41 52 DmECT   ECT     ()              ;Allocated to compare fields
      45 32
0008  43
0009  03
000A  01
```

```
000B  01
000C  01
000D  01
000E  0064
0010  0000
0012  50 43 53 48 41 52
      45 20
001A  50 43 53 48 41 52
      45 20
0022     0A [
                20
                    ]
002C  00
002D  00
002E  00
002F  C0
0030  20
0031  20
0032  00
0033  01
0034  0000
0036  0000
0038  0000
003A  0000
003C  0000
003E  0000
```

```
                        ;
                        ;--------------------------------------------------------------------------
                        ;       Message Strings
                        ;
0040  0A0D              CRLF2$  DW      0A0DH           ;Two carriage return line feed combo
0042  0A0D              CRLF$   DW      0A0DH           ;Carriage return line feed combo
0044  24                Blank$  DB      '$'             ;  used for messages and record end
0045  0D                CR$     DB      0DH             ;Carriage return
                        ;
0046  1B                HiThr   DB      1BH
0047  5B 32 4A                 DB      '[2J'           ;Clear screen
004A  1B                        DB      1BH
004B  5B 35 3B 32 34 48         DB      '[5;24H'        ;Cursor to line 5 column 24
0051  1B                        DB      1BH
0052  5B 37 6D                 DB      '[7m'           ;Reverse video
0055  50 43 53 68 61 72        DB      'PCShare Emulator Initialization'
      65 20 45 6D 75 6C
      61 74 6F 72 20 49
      6E 69 74 69 61 6C
      69 7A 61 74 69 6F
      6E
0074  1B                        DB      1BH
0075  5B 30 6D                 DB      '[0m'           ;Cancel reverse video
0078  1B                        DB      1BH
0079  5B 39 3B 31 48 24        DB      '[9;1H$'        ;Cursor to line 9 column 4
                        ;
007F  45 6D 75 6C 61 74 NoPCS   DB      'Emulator not installed$'
      6F 72 20 6E 6F 74
```

```
                20 69 6E 73 74 61
                6C 6C 65 64 24
0096  45 6D 75 6C 61 74        NoVer   DB      'Emulator version does not match initialization code$'
      6F 72 20 76 65 72
      73 69 6F 6E 20 64
      6F 65 73 20 6E 6F
      74 20 6D 61 74 63
      68 20 69 6E 69 74
      69 61 6C 69 7A 61
      74 69 6F 6E 20 63
      6F 64 65 24
00CA  43 61 6E 6E 6F 74        NoDir   DB      'Cannot find Emulator directory$'
      20 66 69 6E 64 20
      45 6D 75 6C 61 74
      6F 72 20 64 69 72
      65 63 74 6F 72 79
      24
00E9  41 6E 20 65 73 73        NoFle   DB      'An essential Emulator file is missing$'
      65 6E 74 69 61 6C
      20 45 6D 75 6C 61
      74 6F 72 20 66 69
      6C 65 20 69 73 20
      6D 69 73 73 69 6E
      67 24
010F  43 6F 75 6C 64 20        NoLogf  DB      'Could not create a log file$'
      6E 6F 74 20 63 72
      65 61 74 65 20 61
      20 6C 6F 67 20 66
      69 6C 65 24
012B  4C 6F 67 20 66 69        IncLog  DB      'Log file not as large as requested$'
      6C 65 20 6E 6F 74
      20 61 73 20 6C 61
      72 67 65 20 61 73
      20 72 65 71 75 65
      73 74 65 64 24
014E  43 6F 75 6C 64 20        NoSrd   DB      'Could not read system files$'
      6E 6F 74 20 72 65
      61 64 20 73 79 73
      74 65 6D 20 66 69
      6C 65 73 24
016A  55 6E 65 78 70 65        NoDte   DB      'Unexpected firmware version date, check installation$'
      63 74 65 64 20 66
      69 72 6D 77 61 72
      65 20 76 65 72 73
      69 6F 6E 20 64 61
      74 65 2C 20 63 68
      65 63 6B 20 69 6E
      73 74 61 6C 6C 61
      74 69 6F 6E 24
019F  41 72 65 20 69 6E        TBintI  DB      'Are interrupts installed on transporter board? [Y/N]: [Y]'
      74 65 72 72 75 70
      74 73 20 69 6E 73
      74 61 6C 6C 65 64
      20 6F 6E 20 74 72
      61 6E 73 70 6F 72
```

```
                74 65 72 20 62 6F
                61 72 64 3F 20 5B
                59 2F 4E 5D 3A 20
                5B 59 5D
01D8  1B                                DB      1BH
01D9  5B 32 44 24                       DB      '[2D$'          ;Cursor back two
01DD  49 73 20 45 6D 75         TBrenb  DB      'Is Emulator to run with interrupts enabled? [Y/N]: [Y]'
      6C 61 74 6F 72 20
      74 6F 20 72 75 6E
      20 77 69 74 68 20
      69 6E 74 65 72 72
      75 70 74 73 20 65
      6E 61 62 6C 65 64
      3F 20 5B 59 2F 4E
      5D 3A 20 5B 59 5D
0213  1B                                DB      1BH
0214  5B 32 44 24                       DB      '[2D$'          ;Cursor back two
0218  49 73 20 45 6D 75         TBrtck  DB      'Is Emulator to be run by timer tick interrupt? [Y/N]: [N]'
      6C 61 74 6F 72 20
      74 6F 20 62 65 20
      72 75 6E 20 62 79
      20 74 69 6D 65 72
      20 74 69 63 6B 20
      69 6E 74 65 72 72
      75 70 74 3F 20 5B
      59 2F 4E 5D 3A 20
      5B 4E 5D
0251  1B                                DB      1BH
0252  5B 32 44 24                       DB      '[2D$'          ;Cursor back two
0256  41 72 65 20 65 72         WhERR1  DB      'Are error messages to be displayed at console? [Y/N]: [N]'
      72 6F 72 20 6D 65
      73 73 61 67 65 73
      20 74 6F 20 62 65
      20 64 69 73 70 6C
      61 79 65 64 20 61
      74 20 63 6F 6E 73
      6F 6C 65 3F 20 5B
      59 2F 4E 5D 3A 20
      5B 4E 5D
028F  1B                                DB      1BH
0290  5B 32 44 24                       DB      '[2D$'          ;Cursor back two
0294  50 43 53 68 61 72         InDone  DB      'PCShare Emulator initialization completed$'
      65 20 45 6D 75 6C
      61 74 6F 72 20 69
      6E 69 74 69 61 6C
      69 7A 61 74 69 6F
      6E 20 63 6F 6D 70
      6C 65 74 65 64 24
02BE  54 79 70 65 20 71         GoDrv   DB      'Type q to quit, anything else to call Emulator:$'
      20 74 6F 20 71 75
      69 74 2C 20 61 6E
      79 74 68 69 6E 67
      20 65 6C 73 65 20
      74 6F 20 63 61 6C
      6C 20 45 6D 75 6C
```

```
      61 74 6F 72 3A 24
02EE  50 72 65 73 73 20       NxDrv   DB      'Press any key to stop Emulator calls$'
      61 6E 79 20 6B 65
      79 20 74 6F 20 73
      74 6F 70 20 45 6D
      75 6C 61 74 6F 72
      20 63 61 6C 6C 73
      24
                              ;
0313  07                      Beeper  DB      $Beep           ;Beeps console and moves
0314  1B                              DB      1BH             ;  cursor back one
0315  5B 31 44 24                     DB      '[1D$'
                              ;
                              IF TRACE$
0319  4F 70 65 72 61 74       WhTrce  DB      'Operate with trace on? [Y/N]: [N]'
      65 20 77 69 74 68
      20 74 72 61 63 65
      20 6F 6E 3F 20 5B
      59 2F 4E 5D 3A 20
      5B 4E 5D
033A  1B                              DB      1BH
033B  5B 32 44 24                     DB      '[2D$'          ;Cursor back two
033F  4F 75 74 70 75 74       WhTdsk  DB      'Output trace records to disk? [Y/N]: [Y]'
      20 74 72 61 63 65
      20 72 65 63 6F 72
      64 73 20 74 6F 20
      64 69 73 6B 3F 20
      5B 59 2F 4E 5D 3A
      20 5B 59 5D
0367  1B                              DB      1BH
0368  5B 32 44 24                     DB      '[2D$'          ;Cursor back two
036C  4F 75 74 70 75 74       WhTcon  DB      'Output trace records to console? [Y/N]: [N]'
      20 74 72 61 63 65
      20 72 65 63 6F 72
      64 73 20 74 6F 20
      63 6F 6E 73 6F 6C
      65 3F 20 5B 59 2F
      4E 5D 3A 20 5B 4E
      5D
0397  1B                              DB      1BH
0398  5B 32 44 24                     DB      '[2D$'          ;Cursor back two
                              ENDIF
                              ;
                              ;--------------------------------------------------------------------------
                              ;       File Control Data
                              ;
039C  FF                      InFCB   EFCB    ()              ;Standard extended file control block
039D     05 [
               00
                  ]
03A2  00
03A3  00
03A4     08 [
               20
                  ]
```

```
03AC     03 [
              20
                  ]
03AF  0000
03B1  0000
03B3  0000
03B5  0000
03B7  0000
03B9     0A [
              00
                  ]
03C3  00
03C4  0000
03C6  0000

03C8  43                        InDSA   DB      'C'             ;Changed to logged disk letter
03C9  3A 5C                             DB      ':\'            ;For return to previous value
03CB     40 [                   InDSA1  DB      64 DUP(0)       ;64 bytes in which to save directory
              00
                  ]

040B  43                        InDIR   DB      'C'             ;Changed to Emulator disk letter
040C  3A 5C                             DB      ':\'            ;For setting directory path
040E     08 [                   InDIR1  DB      $$DIR DUP(' ')  ;Holds Emulator directory name
              20
                  ]

0416  00                                DB      0               ;End if all eight bytes used
0417     80 [                   InDTA   DB      128 DUP(0)      ;DTA for receive
              00
                  ]

0497  0001                      InLogm  DW      1               ;DTA and first record if we write log
0499  43 6F 72 76 75 73                 DB      'Corvus PCShare Disk Server Emulator, Version 2.0'
      20 50 43 53 68 61
      72 65 20 44 69 73
      6B 20 53 65 72 76
      65 72 20 45 6D 75
      6C 61 74 6F 72 2C
      20 56 65 72 73 69
      6F 6E 20 32 2E 30
04C9     4C [                           DB      76 DUP(' ')
              20
                  ]

0515  0D0A                              DW      0D0AH           ;  end of DTA will have CRLF
                                ;
0517  50 43 53 4D 48 4F         EmNme   DB      'PCSMHOOK'      ;Driver name of Emulator
      4F 4B
051F  07BE                      EmVers  DW      $1982           ;Version date expected (reverse bytes)
                                ;
0521  55 53 52                  InEXT   DB      'USR'           ;Holds extension for file names
0524  53 59 31                          DB      'SY1'
0527  53 59 32                          DB      'SY2'
052A  4C 4F 47                          DB      'LOG'
```

```
                        ;
                        ;*************************************************************************
                        ;                       Internal Subroutines                             |
                        ;*************************************************************************
                        ;
                        ;-------------------------------------------------------------------------
                        ;       Requests open of current file in FCB InFCB.  Input is            |
                        ;               (AL) = attribute for file                                |
                        ;            DS:(SI) = offset of extension                               |
                        ;       Condition code set on exit to zero (equal) if open fails         |
                        ;       and (SI) = (SI)+ 3                                               |
                        ;-------------------------------------------------------------------------
                        ;
052D                    ISOpn   PROC    NEAR
052D  A2 03A2 R                 MOV     InFCB.ATR,AL    ;Set desired attribute
0530  AC                        LODSB                           ;Insert three bytes
0531  A2 03AC R                 MOV     BYTE PTR InFCB.EXT,AL   ;  from string into
0534  AC                        LODSB                           ;  into the extension
0535  A2 03AD R                 MOV     BYTE PTR InFCB.EXT+1,AL ;  and increment the
0538  AC                        LODSB                           ;  string position
0539  A2 03AE R                 MOV     BYTE PTR InFCB.EXT+2,AL
                        ;
053C  BA 039C R                 MOV     DX,OFFSET InFCB
053F  B4 0F                     MOV     AH,0FH          ;Try to open file
0541  CD 21                    INT     21H
0543  3C FF                    CMP     AL,0FFH         ;Set condition code
0545  C3                       RET                     ;Exit
0546                    ISOpn   ENDP
                        ;
                        ;-------------------------------------------------------------------------
                        ;       Requests close of current file in FCB InFCB                      |
                        ;-------------------------------------------------------------------------
                        ;
0546                    ISCls   PROC    NEAR
0546  BA 039C R                 MOV     DX,OFFSET InFCB
0549  B4 10                     MOV     AH,10H          ;Set close code
054B  CD 21                    INT     21H             ;  and call DOS
054D  C3                       RET
054E                    ISCls   ENDP
                        ;
                        ;-------------------------------------------------------------------------
                        ;       Requests output of a message with optional reply.  Input is      |
                        ;               (CL) = 0 then no reply; not 0, reply                     |
                        ;               (CH) = 0 end with CRLF; not 0, no CRLF                   |
                        ;            DS:(DX) = offset of message string                          |
                        ;       (CL) contains (one character) reply if requested.                |
                        ;-------------------------------------------------------------------------
                        ;
054E                    ISMsg   PROC    NEAR
054E  B4 09                     MOV     AH,9            ;Call Dos to
0550  CD 21                    INT     21H             ;  output message
0552  80 F9 00                 CMP     CL,0            ;If no reply
0555  74 08                    JE      ISMsg1          ;  output CRLF
                        ;
0557  B0 01                     MOV     AL,1            ;Set one character reply
```

```
0559  B4 0C                         MOV     AH,0CH          ;Go clear buffer
055B  CD 21                         INT     21H             ;  and wait for character
055D  8A C8                         MOV     CL,AL           ;Save character
                                    ;
055F  80 FD 00                ISMsg1: CMP     CH,0            ;Exit if
0562  75 07                         JNE     ISMsg2          ;  CRLF supressed
0564  BA 0042 R                     MOV     DX,OFFSET CRLF$ ;  Else put out
0567  B4 09                         MOV     AH,9            ;  the CRLF
0569  CD 21                         INT     21H
056B  C3                      ISMsg2: RET                     ;And exit
056C                          ISMsg   ENDP
                              ;
                              ;-----------------------------------------------------------------------------
                              ;       Tests a reply for a given default.  Input is                        |
                              ;               (CL) = Reply character                                      |
                              ;               (CH) = default character                                    |
                              ;       Returns zero if reply is carriage return or equal to default        |
                              ;       character in either upper or lower case; non-zero otherwise         |
                              ;-----------------------------------------------------------------------------
                              ;
056C                          CHKrpy  PROC    NEAR
                              ;
056C  3A 0E 0045 R                    CMP     CL,CR$          ;Exit if reply
0570  74 06                           JE      CHKxit          ;  is carriage return
                              ;
0572  81 C9 2020                      OR      CX,2020H        ;Assure both lower case
0576  3A E9                           CMP     CH,CL
                              ;
0578  C3                      CHKxit: RET                     ;Return to caller
                              ;
0579                          CHKrpy  ENDP
                              ;
                              ;-----------------------------------------------------------------------------
                              ;       Forces a reply of 'y', 'n', or carriage return.  Input is           |
                              ;               (CL) = Reply character                                      |
                              ;               (CH) = 'y' or 'n', whichever is default                     |
                              ;       Returns zero if reply is same as CH character; non-zero if not.     |
                              ;       CR same as that given for character in CH; no distinction made      |
                              ;       betweeen upper and lower case                                       |
                              ;-----------------------------------------------------------------------------
                              ;
0579                          CHKyon  PROC    NEAR
                              ;
0579  51                              PUSH    CX              ;Save input
057A  E8 056C R                       CALL    CHKrpy          ;Check for default
057D  74 22                           JZ      CHKYnd          ;Exit if so
                              ;
057F  80 FD 6E                        CMP     CH,'n'          ;Set opposite
0582  B5 79                           MOV     CH,'y'          ;  condition from
0584  74 02                           JE      CHKg1           ;  default entry
0586  B5 6E                           MOV     CH,'n'
0588  E8 056C R               CHKg1:  CALL    CHKrpy          ;Check for it
058B  74 12                           JZ      CHKYpn          ;Got to set end if so
                              ;
                                      SHOW    Beeper,1,no     ;Beep and get another response
```

```
058D  BA 0313 R           +          MOV     DX,OFFSET Beeper
0590  B9 0001             +          MOV     CX,1
0593  B5 01               +          MOV     CH,1
0595  E8 054E R           +          CALL    ISMsg
0598  8B D1                         MOV     DX,CX           ;Save response
059A  59                           POP     CX              ;Set up as original
059B  8A CA                        MOV     CL,DL           ;  into register
059D  EB DA                        JMP     CHKyon          ;  and go back to process
                             ;
059F  0A ED                  CHKYpn: OR      CH,CH           ;Set non-zero indication
05A1  9C                     CHKynd: PUSHF                   ;Save condition code
                                     SHOW    Blank$          ;Go to next line
05A2  BA 0044 R           +          MOV     DX,OFFSET Blank$
05A5  B9 0000             +          MOV     CX,0
05A8  E8 054E R           +          CALL    ISMsg
                             ;
05AB  9D                           POPF                    ;Restore condition code
05AC  59                           POP     CX              ;   and register
05AD  C3                           RET                     ;Return to caller
                             ;
05AE                         CHKyon  ENDP
                             ;
                             ;*****************************************************************************
                             ;                          Initialization procedure                          |
                             ;*****************************************************************************
                             ;
                             ;-----------------------------------------------------------------------------
                             ;       Set segment registers and find the Emulator                          |
                             ;-----------------------------------------------------------------------------
                             ;
05AE                         EI_INIT PROC NEAR
                             ;
05AE                         Begin   LABEL NEAR
                             ;
05AE  8C C8                        MOV     AX,CS           ;All segment registers
05B0  8E D8                        MOV     DS,AX           ;  are to have the same value
05B2  8E C0                        MOV     ES,AX
                                   SHOW    HiThr           ;Print start message
05B4  BA 0046 R           +        MOV     DX,OFFSET HiThr
05B7  B9 0000             +        MOV     CX,0
05BA  E8 054E R           +        CALL    ISMsg
                             ;
                                   MOVEB   InFCB.NAME,EmNme,,8     ;Emulator Name to FCB
05BD  BE 0517 R           +         MOV   SI,OFFSET EmNme   ;OFFSET OF "FROM" LOCATION
05C0  BF 03A4 R           +         MOV   DI,OFFSET InFCB.NAME      ;OFFSET OF "TO" LOCATION
05C3  B9 0008             +         MOV   CX,8        ;LENGTH VALUE SPECIFIED
05C6  F3/ A4              +         REP MOVSB                 ;MOVE DATA
05C8  B0 00                        MOV     AL,0                    ;Set zero attributes
05CA  BE 03AC R                    MOV     SI,OFFSET InFCB.EXT     ;Extension to itself
05CD  E8 052D R                    CALL    ISOpn                   ;Try to open
05D0  75 0D                        JNE     Cont            ;Continue if ok
                             ;
                             ;       If Emulator not there, output 'not installed' message and exit
                             ;
                                   SHOW    NoPCS           ;Output and
```

```
05D2  BA 007F R           +          MOV     DX,OFFSET NoPCS
05D5  B9 0000             +          MOV     CX,0
05D8  E8 054E R           +          CALL    ISMsg
                                     LEAVE                   ;  go back to DOS
05DB  B4 4C               +          MOV     AH,4CH
05DD  CD 21               +          INT     21H
                        ;
                        ;--------------------------------------------------------------------------
                        ;       Call Emulator to obtain jump table location                       |
                        ;--------------------------------------------------------------------------
                        ;
05DF  BA 0417 R           Cont:      MOV     DX,OFFSET InDTA         ;Set DTA to the
05E2  B4 1A                          MOV     AH,1AH                  ;  location of the
05E4  CD 21                          INT     21H                     ;  internal log record
                        ;
05E6  BA 039C R                      MOV     DX,OFFSET InFCB         ;Go fetch the Device
05E9  B4 14                          MOV     AH,14H                  ;  Driver jump table
05EB  CD 21                          INT     21H                     ;  location
                        ;
05ED  8B 3E 0417 R                   MOV     DI,WORD PTR InDTA       ;Fetch offset, save for
05F1  89 3E 003E R                   MOV     DmECT.TckS,DI           ;  TLD call, position
05F5  83 C7 14                       ADD     DI,$PosKey              ;  to Key offset table
05F8  89 3E 0038 R                   MOV     DmECT.DDRH,DI           ;  and save for future
05FC  8B 36 0419 R                   MOV     SI,WORD PTR InDTA+2     ;Fetch and save segment
0600  89 36 003A R                   MOV     DmECT.DDRHS,SI
                        ;
                        ;--------------------------------------------------------------------------
                        ;       Check to make sure we have the right location and version         |
                        ;--------------------------------------------------------------------------
                        ;
0604  8E C6                          MOV     ES,SI                   ;Fetch segment for ECT
0606  26: 8B 3D                      MOV     DI,ES:[DI].KY_EC        ;Fetch ECT offset
0609  89 3E 003C R                   MOV     DmECT.TckO,DI           ;  and save it
060D  BE 0000 R                      MOV     SI,OFFSET DmECT.Iden
0610  B9 0008                       MOV     CX,8                ;Compare 8 characters
0613  FC                             CLD                         ;  in forward direction
0614  F3/ A6                         REPZ CMPSB                  ;Comparison is match
0616  74 0D                         JZ      Cont1               ;Continue if ok
                        ;
                        ;       If Emulator versions do not match (or read has failed)
                        ;       output 'incorrect version' message and exit
                        ;
                                     SHOW    NoVer               ;Output and
0618  BA 0096 R           +          MOV     DX,OFFSET NoVer
061B  B9 0000             +          MOV     CX,0
061E  E8 054E R           +          CALL    ISMsg
                                     LEAVE                   ;  go back to DOS
0621  B4 4C               +          MOV     AH,4CH
0623  CD 21               +          INT     21H
                        ;
                        ;--------------------------------------------------------------------------
                        ;       Fetch ECT data and store it into save areas and FCB               |
                        ;--------------------------------------------------------------------------
                        ;
0625  8B 36 003C R        Cont1:     MOV     SI,DmECT.TckO           ;Recover ECT offset
```

```
0629  26: 8A 44 08                    MOV     AL,ES:[SI].EC_DAT       ;Get drive letter
062D  A2 03C8 R                       MOV     InDSA,AL                ;Store letter in save area
0630  A2 040B R                       MOV     InDIR,AL                ;  and new directory area
0633  26: 8A 44 09                    MOV     AL,ES:[SI].EC_Drv       ;Move drive number
0637  A2 03A3 R                       MOV     InFCB.DRV,AL            ;  into file control block
                                ;
063A  1E                              PUSH    DS                      ;Restore ES to
063B  07                              POP     ES                      ;  this segment
063C  8E 1E 003A R                    MOV     DS,DmECT.DDRHS          ;Set DS to ECT segment
0640  8D 74 12                        LEA     SI,[SI].Dir             ;Position to and fetch
                                      MOVEB   InDIR1,,,$$DIR          ;  directory string
0643  BF 040E R                 +      MOV   DI,OFFSET InDIR1   ;OFFSET OF "TO" LOCATION
0646  B9 0008                   +      MOV   CX,$$DIR           ;LENGTH VALUE SPECIFIED
0649  F3/ A4                    +      REP MOVSB                ;MOVE DATA
                                      SCANB   InDIR1,,$$DIR,' '       ;Find first blank
064B  B0 20                     +      MOV   AL,' '     ;PUT THE BYTE TO SEARCH FOR IN AL
064D  BF 040E R                 +      MOV   DI,OFFSET InDIR1  ;OFFSET OF "WHERE" LOCATION
0650  B9 0008                   +      MOV   CX,$$DIR           ;LENGTH TO SEARCH
0653  F2/ AE                    +      REPNZ SCASB              ;SCAN FOR SPECIFIED BYTE
0655  75 01                     +      JNZ   ??0000             ;IF NOT FOUND
0657  4F                        +      DEC   DI                 ;BACK UP ONE
0658  90                        + ??0000: NOP
0659  26: C6 05 00                    MOV     BYTE PTR ES:[DI],0      ;Put a zero in its place
                                ;
065D  26: 8B 36 003C R                MOV     SI,ES:DmECT.TckO        ;Recover ECT offset and
0662  8D 74 1A                        LEA     SI,[SI].Nam             ;  position to file name
                                      MOVEB   InFCB.NAME,,,$$NAM      ;Move name to FCB
0665  BF 03A4 R                 +      MOV   DI,OFFSET InFCB.NAME      ;OFFSET OF "TO" LOCATION
0668  B9 0008                   +      MOV   CX,$$NAM           ;LENGTH VALUE SPECIFIED
066B  F3/ A4                    +      REP MOVSB                ;MOVE DATA
                                ;
                                ;-----------------------------------------------------------------------
                                ;       Save current directory and change to that of Emulator          |
                                ;-----------------------------------------------------------------------
                                ;
066D  06                              PUSH    ES                      ;Restore DS to
066E  1F                              POP     DS                      ;  this segment
066F  BE 03CB R                       MOV     SI,OFFSET InDSA1        ;Offset to Save area
0672  B4 47                           MOV     AH,47H                  ;Request directory path
0674  8A 16 03A3 R                    MOV     DL,InFCB.DRV            ;  of Emulator drive
0678  CD 21                           INT     21H
                                ;
067A  BA 040B R                       MOV     DX,OFFSET InDIR         ;Change to
067D  B4 3B                           MOV     AH,3BH                  ;  Emulator directory
067F  CD 21                           INT     21H
0681  73 0D                           JNC     Cont2           ;Continue if change ok
                                ;
                                      SHOW    NoDIR           ;Send message
0683  BA 00CA R                 +      MOV     DX,OFFSET NoDIR
0686  B9 0000                   +      MOV     CX,0
0689  E8 054E R                 +      CALL    ISMsg
                                      LEAVE                   ;  and exit
068C  B4 4C                     +      MOV     AH,4CH
068E  CD 21                     +      INT     21H
                                ;
```

```
                        ;--------------------------------------------------------------------------
                        ;       Try to open user area file and read the version record           |
                        ;--------------------------------------------------------------------------
                        ;
0690  8E 06 003A R      Cont2:  MOV     ES,DmECT.DDRHS          ;Set ES to Emulator segment
0694  8B 3E 003C R              MOV     DI,DmECT.TckO           ;Recover ECT offset
0698  26: 8A 45 2C              MOV     AL,ES:[DI].EC_AtrU      ;Fetch user attributes
069C  BE 0521 R                 MOV     SI,OFFSET InEXT         ;Extension will say USR
069F  E8 052D R                 CALL    ISOpn                   ;Try to open file
06A2  75 17                     JNE     Cont3                   ;Continue if open is ok
                        ;
                        ;       Exit if the user or any of the system files cannot be opened
                        ;
06A4                    Nofile: SHOW    NoFle                   ;Print message
06A4  BA 00E9 R       +         MOV     DX,OFFSET NoFle
06A7  B9 0000         +         MOV     CX,0
06AA  E8 054E R       +         CALL    ISMsg
06AD  E8 0546 R         CmXit:  CALL    ISCls                   ;Close last file
06B0  BA 03C8 R                 MOV     DX,OFFSET InDSA         ;Change back to
06B3  B4 3B                     MOV     AH,3BH                  ;  previous directory
06B5  CD 21                     INT     21H
                                LEAVE                           ;  and exit
06B7  B4 4C           +         MOV     AH,4CH
06B9  CD 21           +         INT     21H
                        ;
06BB  BA 039C R         Cont3:  MOV     DX,OFFSET InFCB         ;Fetch FCB offset
06BE  C7 06 03C4 R 0020         MOV     InFCB.LRN,$VerRec       ;Set record number
06C4  B4 21                     MOV     AH,21H                  ;Try to read in
06C6  CD 21                     INT     21H                     ;  the record
06C8  3C 00                     CMP     AL,0                    ;Treat as uninitialized
06CA  75 22                     JNE     Cont4                   ;  if read fails
                        ;
06CC  A1 045B R                 MOV     AX,WORD PTR InDTA+$FrmDte       ;Fetch version date
06CF  86 C4                     XCHG    AL,AH                           ;Adjust for order
06D1  3B 06 051F R              CMP     AX,EmVers               ;Treat as uninitialized
06D5  75 17                     JNE     Cont4                   ;  if dates do not compare
                        ;
                        ;--------------------------------------------------------------------------
                        ;       Move Server name from user record to the ECT                     |
                        ;--------------------------------------------------------------------------
                        ;
06D7  8B 3E 003C R              MOV     DI,DmECT.TckO           ;Fetch ECT location
06DB  83 C7 22                  ADD     DI,OFFSET Serv          ;Position to server
06DE  BE 0012                   MOV     SI,$FrmSrv              ;Set source as
06E1  81 C6 0417 R              ADD     SI,OFFSET InDTA         ;  user record location
06E5  1E                        PUSH    DS
06E6  06                        PUSH    ES                      ;Set source segment
06E7  1F                        POP     DS                      ;  to be Emulator
                                MOVEB   ,,,10                   ;Fetch the name
06E8  B9 000A         +          MOV    CX,10       ;LENGTH VALUE SPECIFIED
06EB  F3/ A4          +          REP MOVSB                      ;MOVE DATA
06ED  1F                        POP     DS                      ;Restore segment register
                        ;
                        ;--------------------------------------------------------------------------
                        ;       Now try to open the other Emulator files                        |
```

← Push SI

X

← SI

```
                                ;--------------------------------------------------------------------------
                                ;
06EE  E8 0546 R                 Cont4:  CALL    ISCls                   ;Close user file
06F1  8E 06 003A R                      MOV     ES,DmECT.DDRHS          ;Set ES to Emulator segment
06F5  8B 3E 003C R                      MOV     DI,DmECT.TckO           ;Recover offset of ECT
06F9  26: 8A 45 2D                      MOV     AL,ES:[DI].EC_AtrS      ;Fetch system attributes
06FD  E8 052D R                         Call    ISOpn                   ;Try to open
0700  74 A2                             JE      Nofile                  ;Exit if failure
0702  E8 0546 R                         CALL    ISCls                   ;Close the file
0705  26: 8A 45 2D                      MOV     AL,ES:[DI].EC_AtrS      ;Do same for
0709  E8 052D R                         CALL    ISOpn                   ;  second system file
070C  74 96                             JE      Nofile
070E  E8 0546 R                         CALL    ISCls
                                ;
0711  26: 8A 45 2E                      MOV     AL,ES:[DI].EC_AtrL      ;Now try for
0715  E8 052D R                         CALL    ISOpn                   ;  log file
0718  75 60                             JNE     Cont5                   ;Continue if present
                                ;
                                ;--------------------------------------------------------------------------
                                ;       If log file not present create one and write its records          |
                                ;--------------------------------------------------------------------------
                                ;
071A  BA 039C R                         MOV     DX,OFFSET InFCB         ;Point to FCB
071D  B4 16                             MOV     AH,16H                  ;Try to create
071F  CD 21                             INT     21H                     ;  the log file
0721  3C FF                             CMP     AL,0FFH                 ;If the create fails,
0723  75 0C                             JNE     Colog                   ;  send message, then continue
                                        SHOW    NoLogf
0725  BA 010F R                 +       MOV     DX,OFFSET NoLogf
0728  B9 0000                   +       MOV     CX,0
072B  E8 054E R                 +       CALL    ISMsg
072E  EB 4A 90                          JMP     Cont5
                                ;
0731  BA 0497 R                 Colog:  MOV     DX,OFFSET InLogm        ;Set DTA to location
0734  B4 1A                             MOV     AH,1AH                  ;  of log record area
0736  CD 21                             INT     21H
                                ;
0738  C6 06 03C3 R 00                   MOV     InFCB.CUR,0             ;Set record positions
073D  C7 06 03C4 R 0000                 MOV     InFCB.LRN,0             ; to zero
0743  BA 039C R                         MOV     DX,OFFSET InFCB         ;Set FCB location
0746  B4 15                             MOV     AH,15H                  ;Go write
0748  CD 21                             INT     21H                     ;  first record
                                ;
074A  1E                                PUSH    DS                      ;Restore ES to
074B  07                                POP     ES                      ;  current segment
                                        FILLIT  InLogm,,126,' '         ;Now fill with blanks
074C  BF 0497 R                 +        MOV   DI,OFFSET InLogm  ;OFFSET OF "WHERE" LOCATION
074F  B9 007E                   +        MOV   CX,126     ;LENGTH TO REPLICATE
0752  B0 20                     +        MOV   AL,' '     ;BYTE TO BE REPLICATED
0754  F3/ AA                    +        REP STOSB                ;REPLICATE
                                ;
0756  8E 06 003A R                      MOV     ES,DmECT.DDRHS          ;Set ES back to Emulator
075A  8B 3E 003C R                      MOV     DI,DmECT.TckO           ;  and DI pointing to ECT
075E  26: 8B 4D 0E                      MOV     CX,ES:[DI].Max          ;Fetch number of records
                                ;
```

```
0762  B4 15                    Colg1:  MOV     AH,15H                  ;Go write
0764  CD 21                            INT     21H                     ;  a record
0766  3C 00                            CMP     AL,0                    ;Quit if disk
0768  75 02                            JNE     Colg2                   ;  is full or all
076A  E2 F6                            LOOP    Colg1                   ;  records written
                               ;
076C  83 F9 00                 Colg2:  CMP     CX,0                    ;Write message
076F  74 09                            JE      Cont5                   ;  if disk full
                                       SHOW    IncLog
0771  BA 012B R                +       MOV     DX,OFFSET IncLog
0774  B9 0000                  +       MOV     CX,0
0777  E8 054E R                +       CALL    ISMsg
                               ;
                               ;---------------------------------------------------------------------------
                               ;       Check for automatic response file and use it if present             |
                               ;---------------------------------------------------------------------------
                               ;
077A  90                       Cont5:  NOP                             ;To be coded later
                               ;
                               ;---------------------------------------------------------------------------
                               ;       Set conditions of transporter board usage                           |
                               ;---------------------------------------------------------------------------
                               ;
077B  8B 3E 003C R                     MOV     DI,DmECT.TckO           ;Fetch ECT location
077F  26: C6 45 2F 00                  MOV     ES:[DI].EC_TBI,0        ; and turn off all usage flags
0784  C6 06 002F R 00                  MOV     DmECT.EC_TBI,0          ; in Emulator and here
                               ;
                                       SHOW    TBintI,1,no             ;Prompt for interrupt installed
0789  BA 019F R                +       MOV     DX,OFFSET TBintI
078C  B9 0001                  +       MOV     CX,1
078F  B5 01                    +       MOV     CH,1
0791  E8 054E R                +       CALL    ISMsg
0794  B5 79                            MOV     CH,'y'                  ;Check for default
0796  E8 0579 R                        CALL    CHKyon                  ;  yes reply
0799  75 1F                            JNZ     Cont6                   ;Test for timer opn if not
079B  26: 80 4D 2F 80                  OR      ES:[DI].EC_TBI,$TBintI  ;Turn on flag if so
                               ;
                                       SHOW    TBrenb,1,no             ;Prompt for interrupt usage
07A0  BA 01DD R                +       MOV     DX,OFFSET TBrenb
07A3  B9 0001                  +       MOV     CX,1
07A6  B5 01                    +       MOV     CH,1
07A8  E8 054E R                +       CALL    ISMsg
07AB  B5 79                            MOV     CH,'y'                  ;Check for default
07AD  E8 0579 R                        CALL    CHKyon                  ;  yes reply
07B0  75 08                            JNZ     Cont6                   ;Test for timer opn if not
07B2  26: 80 4D 2F 40                  OR      ES:[DI].EC_TBI,$TBrenb  ;If so, turn on flag
07B7  EB 18 90                         JMP     Cont7                   ;  and go to display question
                               ;
07BA                           Cont6:  SHOW    TBrtck,1,no             ;Prompt for timer tick usage
07BA  BA 0218 R                +       MOV     DX,OFFSET TBrtck
07BD  B9 0001                  +       MOV     CX,1
07C0  B5 01                    +       MOV     CH,1
07C2  E8 054E R                +       CALL    ISMsg
07C5  B5 6E                            MOV     CH,'n'                  ;Check for default
07C7  E8 0579 R                        CALL    CHKyon                  ;  no reply
```

```
07CA  74 05                             JZ      Cont7                   ;Display question if so
07CC  80 0E 002F R 20                   OR      DmECT.EC_TBI,$TBrtck    ;Set to turn on flag if not
                                ;
                                ;-----------------------------------------------------------------------
                                ;       Prompt for error messages displayed at console                  |
                                ;-----------------------------------------------------------------------
                                ;
07D1  26: C6 45 0A 01           Cont7:  MOV     ES:[DI].EC_Flg,$NOCONS  ;Assume no console output
                                ;
                                        SHOW    WhERR1,1,no     ;Prompt for error message at console
07D6  BA 0256 R                 +       MOV     DX,OFFSET WhERR1
07D9  B9 0001                   +       MOV     CX,1
07DC  B5 01                     +       MOV     CH,1
07DE  E8 054E R                 +       CALL    ISMsg
07E1  B5 6E                             MOV     CH,'n'          ;Check for default
07E3  E8 0579 R                         CALL    CHKyon          ;  no reply
07E6  74 05                             JZ      Cont8
07E8  26: C6 45 0A 00                   MOV     ES:[DI].EC_Flg,$Cons    ;Reset if not
                                ;
07ED  E8 0546 R                 Cont8:  CALL    ISCls                   ;Close any open file
07F0  BA 03C8 R                         MOV     DX,OFFSET InDSA         ;Change back to
07F3  B4 3B                             MOV     AH,3BH                  ;   previous directory
07F5  CD 21                             INT     21H
                                ;
                                IF TRACE$
                                ;-----------------------------------------------------------------------
                                ;       Check for trace request and set flag accordingly                |
                                ;-----------------------------------------------------------------------
                                ;
07F7  26: C6 45 0B 01                   MOV     ES:[DI].Trce,$False     ;Assume trace off
                                        SHOW    WhTrce,1,no     ;Display message and get reply
07FC  BA 0319 R                 +       MOV     DX,OFFSET WhTrce
07FF  B9 0001                   +       MOV     CX,1
0802  B5 01                     +       MOV     CH,1
0804  E8 054E R                 +       CALL    ISMsg
0807  B5 6E                             MOV     CH,'n'          ;Check for a
0809  E8 0579 R                         CALL    CHKyon          ;  default no
080C  74 33                             JZ      Cont9
080E  26: C6 45 0B 00                   MOV     ES:[DI].Trce,$True      ;Turn trace on
                                ;
                                        SHOW    WhTdsk,1,no     ;Display disk message and get reply
0813  BA 033F R                 +       MOV     DX,OFFSET WhTdsk
0816  B9 0001                   +       MOV     CX,1
0819  B5 01                     +       MOV     CH,1
081B  E8 054E R                 +       CALL    ISMsg
081E  B5 79                             MOV     CH,'y'          ;Check for a
0820  E8 0579 R                         CALL    CHKyon          ;  default yes
0823  75 05                             JNZ     Cont8a          ;Turn on disk trace flag if so
0825  80 0E 002F R 01                   OR      DmECT.EC_TBI,$TRCdsk
                                ;
082A                            Cont8a: SHOW    WhTcon,1,no     ;Display console message and get reply
082A  BA 036C R                 +       MOV     DX,OFFSET WhTcon
082D  B9 0001                   +       MOV     CX,1
0830  B5 01                     +       MOV     CH,1
0832  E8 054E R                 +       CALL    ISMsg
```

```
0835  B5 6E                          MOV     CH,'n'          ;Check for a
0837  E8 0579 R                      CALL    CHKyon          ;  default no
083A  74 05                          JZ      Cont9           ;Turn on console trace flag if not
083C  80 0E 002F R 02                OR      DmECT.EC_TBI,$TRCcon
                                ENDIF
                                ;
                                ;-----------------------------------------------------------------------------
                                ;     Save interrupt vectors and set intercepts for Emulator, timer, DOS    !
                                ;-----------------------------------------------------------------------------
                                ;
0841                            Cont9:  FIV     21H                     ;Fetch DOS intercept vector
0841  06                      +         PUSH    ES
0842  B4 35                    +         MOV     AH,35H
0844  B0 21                    +         MOV     AL,21H
0846  CD 21                    +         INT     21H
0848  8C C0                    +         MOV     BX,ES
084A  07                      +         POP     ES
084B  26: 89 5D 34                      MOV     ES:[DI].DOSdkO,BX       ;  and save in Emulator ECT
084F  26: 89 55 36                      MOV     ES:[DI].DOSdkS,DX
0853  87 D3                             XCHG    DX,BX                   ;Now set up the
                                        SIV     ,$CallDS                ;  software equivalent
0855  1E                      +         PUSH    DS
0856  8E DB                  +         MOV     DS,BX
0858  B4 25                  +         MOV     AH,25H
085A  B0 66                  +         MOV     AL,$CallDS
085C  CD 21                  +         INT     21H
085E  1F                      +         POP     DS
                                ;
                                        FIV     $CallTK                 ;Fetch timer intercept vector
085F  06                      +         PUSH    ES
0860  B4 35                   +         MOV     AH,35H
0862  B0 1C                   +         MOV     AL,$CallTK
0864  CD 21                   +         INT     21H
0866  8C C0                   +         MOV     BX,ES
0868  07                      +         POP     ES
0869  26: 89 5D 3C                      MOV     ES:[DI].TckO,BX         ;  and save in Emulator ECT
086D  26: 89 55 3E                      MOV     ES:[DI].TckS,DX
0871  87 D3                             XCHG    DX,BX                   ;Now set up the
                                        SIV     ,$CallTM                ;  software call
0873  1E                      +         PUSH    DS
0874  8E DB                  +         MOV     DS,BX
0876  B4 25                  +         MOV     AH,25H
0878  B0 67                  +         MOV     AL,$CallTM
087A  CD 21                  +         INT     21H
087C  1F                      +         POP     DS
                                ;
087D  8B 1E 0038 R                      MOV     BX,DmECT.DDRH           ;Offset to keys table
0881  26: 8B 57 04                      MOV     DX,ES:[BX].KY_INT       ;Fetch offset to int driver
                                        SIV     DmECT.DDRHS,$CallEM     ;Set Emulator interrupt vector
0885  1E                      +         PUSH    DS
0886  8B 1E 003A R            +         MOV     BX,DmECT.DDRHS
088A  8E DB                  +         MOV     DS,BX
088C  B4 25                  +         MOV     AH,25H
088E  B0 0A                  +         MOV     AL,$CallEM
0890  CD 21                  +         INT     21H
```

```
0892  1F                        +          POP     DS
                                ;
0893  8B 1E 0038 R                        MOV     BX,DmECT.DDRH           ;Offset to Keys table
0897  26: 8B 57 06                       MOV     DX,ES:[BX].KY_DTM       ;Offset to timer intercept
                                         SIV     DmECT.DDRHS,$CallTK     ;Set timer intercept vector
089B  1E                        +          PUSH    DS
089C  8B 1E 003A R              +          MOV     BX,DmECT.DDRHS
08A0  8E DB                     +          MOV     DS,BX
08A2  B4 25                     +          MOV     AH,25H
08A4  B0 1C                     +          MOV     AL,$CallTK
08A6  CD 21                     +          INT     21H
08A8  1F                        +          POP     DS
                                ;
08A9  A0 002F R                          MOV     AL,DmECT.EC_TBI         ;Fetch run flags
08AC  8B 1E 0038 R                       MOV     BX,DmECT.DDRH           ;Refresh keys offset
08B0  A8 C0                              TEST    AL,$TBint1+$TBrenb
08B2  74 12                              JZ      Cont10                  ;If transporter is
08B4  26: 8B 57 08                       MOV     DX,ES:[BX].KY_DOS       ;  to run enabled,
                                         SIV     DmECT.DDRHS,21H         ;  set DOS intercept vector
08B8  1E                        +          PUSH    DS
08B9  8B 1E 003A R              +          MOV     BX,DmECT.DDRHS
08BD  8E DB                     +          MOV     DS,BX
08BF  B4 25                     +          MOV     AH,25H
08C1  B0 21                     +          MOV     AL,21H
08C3  CD 21                     +          INT     21H
08C5  1F                        +          POP     DS
                                ;
                                ;---------------------------------------------------------------
                                ;       Request Transporter Logical Device initialization       !
                                ;---------------------------------------------------------------
                                ;
08C6  26: 08 45 2F              Cont10:  OR      ES:[DI].EC_TBI,AL       ;Flags to Emulator
08CA  53                                 PUSH    BX                      ;Save offset of keys table
08CB  A1 003E R                          MOV     AX,DmECT.TckS           ;Fetch offset to
08CE  A3 0038 R                          MOV     DmECT.DDRH,AX           ;  jump table
                                ;
08D1  FF 1E 0038 R                       CALL DWORD PTR DmECT.DDRH       ;Call TLD start (cold = warm)
08D5  5E                                 POP     SI                      ;Retrieve keys offset
                                         SHOW    InDone                  ;Print end message
08D6  BA 0294 R                 +        MOV     DX,OFFSET InDone
08D9  B9 0000                   +        MOV     CX,0
08DC  E8 054E R                 +        CALL    ISMsg
                                ;
                                ;---------------------------------------------------------------
                                ;       Set transporter in motion by itself or through external driver  !
                                ;---------------------------------------------------------------
                                ;
08DF  F6 06 002F R C0                    TEST    DmECT.EC_TBI,$TBint1+$TBrenb
08E4  74 05                              JZ      Cont11                  ;Go drive if no interrupts
08E6  FB                                 STI                             ;Assure interrupts on
                                         LEAVE                           ;  and exit
08E7  B4 4C                     +        MOV     AH,4CH
08E9  CD 21                     +        INT     21H
                                ;
08EB  26: 8B 5C 02              Cont11:  MOV     BX,ES:[SI].KY_EXT       ;Set up call
```

```
08EF  89 1E 0038 R                      MOV     DmECT.DDRH,BX           ;  to Driver
                                        ;
08F3                                    ShowL:  SHOW    GoDrv,1         ;Request start or stop
08F3  BA 02BE R                 +               MOV     DX,OFFSET GoDrv
08F6  B9 0001                   +               MOV     CX,1
08F9  E8 054E R                 +               CALL    ISMsg
08FC  80 C9 20                                  OR      CL,20H          ;Stop is either a capital
08FF  80 F9 71                                  CMP     CL,'q'          ;  or lower case 'q'
0902  75 04                                     JNE     Cont12
                                                LEAVE                   ;Exit on stop
0904  B4 4C                     +               MOV     AH,4CH
0906  CD 21                     +               INT     21H
                                        ;
0908                                    Cont12: SHOW    NxDRV           ;Issue stop message
0908  BA 02EE R                 +               MOV     DX,OFFSET NxDRV
090B  B9 0000                   +               MOV     CX,0
090E  E8 054E R                 +               CALL    ISMsg
0911  FF 1E 0038 R                      DCalEM: CALL DWORD PTR DmECT.DDRH       ;Call the Emulator
                                        ;
0915  B0 FF                                     MOV     AL,0FFH         ;Test for character
0917  B4 06                                     MOV     AH,6            ;  input at keyboard
0919  CD 21                                     INT     21H
091B  75 F4                                     JNZ     DCalEM          ;Call Emulator if not
091D  EB D4                                     JMP     ShowL           ;If so, go ask again
                                        ;
091F                                    EI_INIT ENDP
                                        ;
091F                                    PCS     ENDS
                                        ;
0000                                    STACK   SEGMENT PARA STACK 'STACK'
0000  0200 [                                     DB      512 DUP(0FFH)
              FF
                   ]

0200                                    STACK   ENDS
                                        ;
                                        END Begin
```

Macros:

Name	Length
COMPARE.	000F
ENTER.	000D
EXIT	0005
EYE.	0001
FILLIT	0011
FIV.	0002
GEN_START.	0001
LEAVE.	0001
LOG.	0005
MOVEB.	000F
MOVEONE.	000E
RESTORE.	0001
SCANB.	0012
SELEXIT.	0002
SELREST.	000D
SELSAVE.	0018
SHOW	0004
SIV.	0003
START.	0019
STATUS	0001
TIME	0001
TRACE.	0009

Structures and records:

Name	Width / Shift	# fields / Width	Mask	Initial
DACB	02EB	000F		
DA_EI.	0000			
DA_EC.	0001			
DA_FLG	0002			
OPN.	0003			
DA_CDSK.	0004			
DSA.	0007			
DTA.	0047			
VSA.	004B			
VDT.	004F			
DDA.	006B			
ECT.	0040	001A		
IDEN	0000			
EC_DAT	0008			
EC_DRV	0009			
EC_FLG	000A			
TRCE	000B			
COLD	000C			
EC_TRM	000D			
MAX.	000E			
NEXT	0010			
DIR.	0012			
NAM.	001A			
SERV	0022			

```
  EC_ATRU. . . . . .  002C
  EC_ATRS. . . . . .  002D
  EC_ATRL. . . . . .  002E
  EC_TBI . . . . . .  002F
  TLTMCK . . . . . .  0030
  TLTMRS . . . . . .  0031
  KEEPOUT. . . . . .  0032
  INTDFR . . . . . .  0033
  DOSDKO . . . . . .  0034
  DOSDKS . . . . . .  0036
  DDRH . . . . . .  0038
  DDRHS. . . . . .  003A
  TCKO . . . . . .  003C
  TCKS . . . . . .  003E
EEFCB. . . . . .  00BC  001C
  EFCB1. . . . . .  0000
  SY1. . . . . .  0010
  EFCB2. . . . . .  002C
  SY2. . . . . .  003C
  EFCB3. . . . . .  0058
  USR. . . . . .  0068
  EFCB4. . . . . .  0084
  EE_LOG . . . . . .  0094
  ERR_TAB. . . . . .  00B0
EFCB . . . . . .  002C  000F
  FF . . . . . .  0000
  RSV. . . . . .  0001
  ATR. . . . . .  0006
  DRV. . . . . .  0007
  NAME . . . . . .  0008
  EXT. . . . . .  0010
  CB . . . . . .  0013
  RS . . . . . .  0015
  LFS. . . . . .  0017
  HFS. . . . . .  0019
  EF_DAT . . . . . .  001B
  RSV1 . . . . . .  001D
  CUR. . . . . .  0027
  LRN. . . . . .  0028
  HRN. . . . . .  002A
IORB . . . . . .  0012  000A
  IO_FLG . . . . . .  0000
  IO_CLS . . . . . .  0001
  NUM. . . . . .  0004
  DSPL . . . . . .  0006
  DSPM . . . . . .  0008
  ADR. . . . . .  000A
  FCB. . . . . .  000E
KEYS . . . . . .  000A  0005
  KY_EC. . . . . .  0000
  KY_EXT . . . . . .  0002
  KY_INT . . . . . .  0004
  KY_DTM . . . . . .  0006
  KY_DOS . . . . . .  0008
LRB. . . . . .  0002  0002
```

```
  CDE. . . . . .          0000
  LR_LEN . . . . . .      0001

Segments and groups:

                N a m e         Size    align   combine class

PCS. . . . . .          091F    PARA    NONE
STACK. . . . . .        0200    PARA    STACK   'STACK'

Symbols:

                N a m e         Type    Value   Attr

BEEPER . . . . . .      L BYTE  0313    PCS
BEGIN. . . . . .        L NEAR  05AE    PCS
BLANK$ . . . . . .      L BYTE  0044    PCS
CHKG1. . . . . .        L NEAR  0588    PCS
CHKRPY . . . . . .      N PROC  056C    PCS     Length =000D
CHKXIT . . . . . .      L NEAR  0578    PCS
CHKYND . . . . . .      L NEAR  05A1    PCS
CHKYON . . . . . .      N PROC  0579    PCS     Length =0035
CHKYPN . . . . . .      L NEAR  059F    PCS
CI$. . . . . .          Number  0000
CMXIT. . . . . .        L NEAR  06AD    PCS
COLG1. . . . . .        L NEAR  0762    PCS
COLG2. . . . . .        L NEAR  076C    PCS
COLOG. . . . . .        L NEAR  0731    PCS
CONT . . . . . .        L NEAR  05DF    PCS
CONT1. . . . . .        L NEAR  0625    PCS
CONT10 . . . . . .      L NEAR  08C6    PCS
CONT11 . . . . . .      L NEAR  08EB    PCS
CONT12 . . . . . .      L NEAR  0908    PCS
CONT2. . . . . .        L NEAR  0690    PCS
CONT3. . . . . .        L NEAR  06BB    PCS
CONT4. . . . . .        L NEAR  06EE    PCS
CONT5. . . . . .        L NEAR  077A    PCS
CONT6. . . . . .        L NEAR  07BA    PCS
CONT7. . . . . .        L NEAR  07D1    PCS
CONT8. . . . . .        L NEAR  07ED    PCS
CONT8A . . . . . .      L NEAR  082A    PCS
CONT9. . . . . .        L NEAR  0841    PCS
CR$. . . . . .          L BYTE  0045    PCS
CRLF$. . . . . .        L WORD  0042    PCS
CRLF2$ . . . . . .      L WORD  0040    PCS
DCALEM . . . . . .      L NEAR  0911    PCS
DEBUG$ . . . . . .      Number  0000
DM$. . . . . .          Number  0000
DMECT. . . . . .        L 0040  0000    PCS
DR$. . . . . .          Number  0000
DTA_BGN. . . . . .      E BYTE  0000    PCS
EI$. . . . . .          Number  0001
EI_INIT. . . . . .      N PROC  05AE    PCS     Length =0371
EMNME. . . . . .        L BYTE  0517    PCS
EMVERS . . . . . .      L WORD  051F    PCS
```

EYE$	Number	0000		
GODRV.	L BYTE	02BE	PCS	
HITHR.	L BYTE	0046	PCS	
INCLOG	L BYTE	012B	PCS	
INDIR.	L BYTE	040B	PCS	
INDIR1	L BYTE	040E	PCS	Length =0008
INDONE	L BYTE	0294	PCS	
INDSA.	L BYTE	03C8	PCS	
INDSA1	L BYTE	03CB	PCS	Length =0040
INDTA.	L BYTE	0417	PCS	Length =0080
INEXT.	L BYTE	0521	PCS	
INFCB.	L 002C	039C	PCS	
INLOGM	L WORD	0497	PCS	
IS$.	Number	0000		
ISCLS.	N PROC	0546	PCS	Length =0008
ISMSG.	N PROC	054E	PCS	Length =001E
ISMSG1	L NEAR	055F	PCS	
ISMSG2	L NEAR	056B	PCS	
ISOPN.	N PROC	052D	PCS	Length =0019
LC$.	Number	0000		
NA$.	Number	0000		
NOCOM$	Number	0000		
NODIR.	L BYTE	00CA	PCS	
NODTE.	L BYTE	016A	PCS	
NOFILE	L NEAR	06A4	PCS	
NOFLE.	L BYTE	00E9	PCS	
NOLOGF	L BYTE	010F	PCS	
NOPCS.	L BYTE	007F	PCS	
NOSRD.	L BYTE	014E	PCS	
NOVER.	L BYTE	0096	PCS	
NXDRV.	L BYTE	02EE	PCS	
PI$.	Number	0000		
SHOWL.	L NEAR	08F3	PCS	
SM$.	Number	0000		
TBINTA$.	Number	0001		
TBINTI	L BYTE	019F	PCS	
TBRENB	L BYTE	01DD	PCS	
TBRTCK	L BYTE	0218	PCS	
TL$.	Number	0000		
TRACE$	Number	0001		
WHERR1	L BYTE	0256	PCS	
WHTCON	L BYTE	036C	PCS	
WHTDSK	L BYTE	033F	PCS	
WHTRCE	L BYTE	0319	PCS	
$$DIR.	Number	0008		
$$EXT.	Number	0003		
$$LOG.	Number	0003		
$$NAM.	Number	0008		
$$NAME	Number	0008		
$$NEXT	Number	0004		
$$SERV	Number	000A		
$$SY1.	Number	0003		
$$SY2.	Number	0003		
$$USR.	Number	0003		
$1982.	Number	07BE		

$ARCHIVE	Number	0020
$ATRNRM.	Number	0000
$BEEP.	Number	0007
$CALLDS.	Number	0066
$CALLEM.	Number	000A
$CALLTK.	Number	001C
$CALLTM.	Number	0067
$CONS.	Alias	$TRUE
$DISKERR	Number	0001
$ERRSY1.	Number	0001
$ERRSY2.	Number	0002
$FAILURE	Number	00FF
$FALSE	Number	0001
$FRMDTE.	Number	0044
$FRMSRV.	Number	0012
$HIDDEN.	Number	0002
$LOGOPEN	Number	0008
$MAXMSG.	Number	0066
$MND00	Number	0001
$MND01	Number	0009
$MND02	Number	000A
$MND03	Number	000B
$MND04	Number	000C
$MND05	Number	0011
$MNT00	Number	0002
$MNT01	Number	0003
$MNT010.	Number	0010
$MNT02	Number	0004
$MNT03	Number	0005
$MNT04	Number	0006
$MNT05	Number	0007
$MNT06	Number	0008
$MNT07	Number	000D
$MNT08	Number	000E
$MNT09	Number	000F
$NOCONS.	Alias	$FALSE
$NODERR.	Number	0000
$ONE	Number	0001
$P82EML.	Number	0004
$P82ENB.	Number	00FA
$P82EOI.	Number	0020
$P82EPT.	Number	0005
$P82OP0.	Number	0020
$P82OP1.	Number	0021
$P82RIS.	Number	000B
$POSKEY.	Number	0014
$RDONLY.	Number	0001
$RDWRITE	Number	00FE
$SUBDIR.	Number	0010
$SUCCESS	Number	0000
$SY1OPEN	Number	0001
$SY2OPEN	Number	0002
$SYSTEM.	Number	0004
$TBINTI.	Number	0080
$TBRENB.	Number	0040

```
$TBRTCK. . . . . .          Number  0020
$TICKMX. . . . . .          Number  0020
$TRCCON. . . . . .          Number  0002
$TRCDSK. . . . . .          Number  0001
$TRUE. . . . . .            Number  0000
$USROPEN . . . . . .        Number  0004
$VERREC. . . . . .          Number  0020
$VOLLAB. . . . . .          Number  0008
$ZERO. . . . . .            Number  0000
??0000 . . . . . .          L NEAR  0658    PCS

Warning Severe
Errors  Errors
0       0
```

```
                        ;
                        PAGE 60,132
                        ;
                        ;*******************************************************************************
                        ;                       Filename is PCSCH1.ASM                                 |
                        ;*******************************************************************************
                        ; This file contains the code for Emulator initialization described in         |
                        ; Section 2.8 of the Program Logic Description for the OMNINET PCShare         |
                        ; Disk Server Emulator, Version 2.0, dated 7 November 1983.  The file          |
                        ; also contains the code to interface to PC DOS as an installed device        |
                        ; driver, and code to provide entry points to the Corvus utilities.           |
                        ;                                                                             |
                        ; The requirements for device driver code and data are described in           |
                        ; Chapter 14 of the IBM PC DOS 2.0 manual.  The Emulator as a device          |
                        ; driver is named PCSMHOOK and the procedure is DR-DEV-DRVR.  EI-INIT          |
                        ; (Chapter 9) obtains the location of the Corvus utilities jump table        |
                        ; by a device driver input call (as do the Corvus utilities), and the        |
                        ; offset of the ECT from the first word following the jump table.  At        |
                        ; the completion of its other processing EI-INIT calls the warm start        |
                        ; entry of the jump table to initialize the transporter and start the        |
                        ; operation of the Emulator.                                                 |
                        ;                                                                             |
                        ; Allocated data in the file includes the Device Driver header and the        |
                        ; Emulator Communication Table, so the derived object file must be the       |
                        ; first one seen by the linker.                                              |
                        ;*******************************************************************************
                        ;
                        ;*******************************************************************************
                        ;
                                ENDIF
                        ;
                        ;*******************************************************************************
                        ;
                        ;*******************************************************************************
                        ;
                                GEN_START   (DR)            ;Driver structures
                         +      .LALL
                         +      ENDIF
                         +      ENDIF
                         +      ENDIF
                         +      IFIDN    (TRACE),(TRACE)
= 0001                   + TRACE$  =      1
                         +      ENDIF
                         +      ENDIF
                         +      ENDIF
                         +      ENDIF
                         +      ENDIF
                         +      ENDIF
                         +      ENDIF
                         +      ENDIF
                         +      IFIDN    (DEBUG),(DEBUG)
= 0001                   + DEBUG$  =      1
                         +      ENDIF
                         +      ENDIF
                        C+      INCLUDE PCSTRUC.CRV
```

```
C+ ;*****************************************************************************
C+ ;                        Filename is PCSTRUC.CRV                               |
C+ ;*****************************************************************************
C+ ; This file contains all structure definitions for the implementation         |
C+ ; of the Corvus OMNINET PCShare Disk Server Emulator.  It is included         |
C+ ; in each of the assembly files when the START macro is invoked.  The         |
C+ ; selectors specified with that macro invocation cause the associated         |
C+ ; structure definitions to be included in the assembly.  Any set of           |
C+ ; definitions can be included with an appropriate set of selectors.           |
C+ ;*****************************************************************************
C+ ;                                                                             |
C+ ; Version:  2.0                                                               |
C+ ;                                                                             |
C+ ; Last Update:  14 January 1984                                               |
C+ ;                                                                             |
C+ ;*****************************************************************************
C+ IF DR$
C+ ;-----------------------------------------------------------------------------
C+ ;                     Operating System Structures                             |
C+ ;-----------------------------------------------------------------------------
C+ ;
C+ ;-----------------------------------------------------------------------------
C+ ;       Structure Name:  Device Driver Header                                 |
C+ ;       Reference: IBM PC DOS 2.0 Manual, page 14-5                           |
C+ ;-----------------------------------------------------------------------------
C+ ;
C+ DDH     STRUC
0000  FF FF FF FF          C+ Ndev    DD      -1              ;No next device
0004  8000                 C+ Attr    DW      8000H           ;Character device not supporting IOCTL
0006  0000                 C+ Strtn   DW      0               ;Offset to strategy routine
0008  0000                 C+ Dvrtn   DW      0               ;Offset to device exec routine
000A  50 43 53 4D 48 4F    C+ Dvnme   DB      'PCSMHOOK'      ;Corvus utilities call by this name
      4F 4B                C+
0012                       C+ DDH     ENDS
                           C+ ;
                           C+ ;-----------------------------------------------------------------------------
                           C+ ;       Structure Name:  Request Header                                       |
                           C+ ;       Reference: IBM PC DOS 2.0 Manual, page 14-11                          |
                           C+ ;-----------------------------------------------------------------------------
                           C+ ;
                           C+ ;       Associated return status flag values
                           C+ ;
= 0080                     C+ $RQerr  EQU     80H             ;Indicates an error
= 0001                     C+ $RQdun  EQU     1               ;Indicates completion of request
                           C+ ;
                           C+ ;       The structure itself
                           C+ ;
                           C+ RQH     STRUC
0000  14                   C+ RQlen   DB      20              ;Length of the RQ header plus data
0001  00                   C+ RQucde  DB      0               ;Unit code (N/A for char device)
0002  00                   C+ RQopn   DB      0               ;The command to be executed
0003  0000                 C+ RQstat  DW      0               ;Return status placed here
0005     08 [              C+         DB      8 DUP(0)        ;DOS uses this
               00          C+
                    ]      C+
```

```
                        C+
000D  00                C+ RQdta   DB      0               ;Data always starts with byte
000E  00 00 00 00       C+ RQrtn   DD      0               ;  followed by buffer address
0012  0000              C+ RQct    DW      0               ;  followed by data count
0014                    C+ RQH     ENDS
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ IF (DR$ OR EI$)
                        C+ ;--------------------------------------------------------------------------
                        C+ ;       Structure Name:  Emulator Access Table                          |
                        C+ ;       Reference: Program Logic description, Sections 9.2 and 9.3      |
                        C+ ;--------------------------------------------------------------------------
                        C+ ;
                        C+ ;       Associated constant values
                        C+ ;
= 0014                  C+ $PosKey EQU     20              ;Offset to locate the keys
                        C+ ;
                        C+ KEYS    STRUC
0000  0000              C+ KY_EC   DW      0               ;Place for offset to ECT
0002  0000              C+ KY_EXT  DW      0               ;Place for offset to external driver
0004  0000              C+ KY_INT  DW      0               ;Place for offset to interrupt driver
0006  0000              C+ KY_DTM  DW      0               ;Place for Offset to timer intercept
0008                    C+ KEYS    ENDS
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ ;--------------------------------------------------------------------------
                        C+ ;                   Emulator Resident Data Structures                   |
                        C+ ;--------------------------------------------------------------------------
                        C+ ;
                        C+ ;       Universal value definitions
                        C+ ;
= 0000                  C+ $True   EQU     0               ;Usually indicates OK or normal
= 0001                  C+ $False  EQU     1               ;Usually indicates NG or unusual
= 0000                  C+ $Zero   EQU     0               ;What is there to say
= 0001                  C+ $One    EQU     1               ;  about these?
= 0060                  C+ $SKnum  EQU     96              ;Number of internal stack entries
= 0001                  C+ $Vir_Drv1 EQU   1               ;Value for Virtual Drive 1
                        C+ ;
                        C+ ;--------------------------------------------------------------------------
                        C+ ;       8259A PIC usage values                                          |
                        C+ ;       Reference: Intel 8086 User's Manual                             |
                        C+ ;--------------------------------------------------------------------------
                        C+ ;
= 0021                  C+ $P82op1 EQU     21H             ;Port for OCW1 commands
= 0020                  C+ $P82op0 EQU     20H             ;Port for OCW2, OCW3 commands
                        C+ ;
= 000B                  C+ $P82Ris EQU     0BH             ;Read Interrupt Service register
= 0020                  C+ $P82Eoi EQU     20H             ;Simple end of interrupt
= 0001                  C+ $P82TML EQU     1               ;Timer interrupt level bit
= 0005                  C+ $P82EML EQU     4+$P82TML       ;Transporter and timer levels
= 00FA                  C+ $P82enb EQU     0FFH-$P82EML    ;Enable xporter and timer
                        C+ ;
                        C+ ;--------------------------------------------------------------------------
```

```
                        C+ ;       Structure Name:  Emulator Communication Table                |
                        C+ ;       Reference: Program Logic description, Section 1.4            |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ ;       Associated values and flags
                        C+ ;
= 0080                  C+ $Cons    EQU     80H             ;Write messages to console
= 0040                  C+ $ConsPT  EQU     40H             ;Prompt for console at initialization
= 0000                  C+ $AtrNrm  EQU     0               ;Standard attributes of Emulator files
= 000A                  C+ $TickMX  EQU     10              ;Default timer tick refresh count
                        C+ ;
= 0080                  C+ $TBintI  EQU     80H             ;Transporter interrupts installed
= 0040                  C+ $TBrenb  EQU     40H             ;Transporter to run enabled
= 0020                  C+ $TBrtck  EQU     20H             ;Transporter to run off timer tick
= 00DF                  C+ $TBnotk  EQU     0FFH-$TBrtck    ;Mask to clear timer tick on
= 0010                  C+ $TBinit  EQU     10H             ;Prompt for interrupts at init
= 0004                  C+ $TRChrd  EQU     4               ;Trace output tp hard disk
= 0002                  C+ $TRCcon  EQU     2               ;Trace output to console
= 0001                  C+ $TRCdsk  EQU     1               ;Trace output to disk
                        C+ ;
                        C+ ;       Interrupt vector numbers
                        C+ ;
= 000A                  C+ $CallEM  EQU     0AH             ;Emulator interrupt
= 001B                  C+ $CallCB  EQU     1BH             ;Control_break has occurred
= 001C                  C+ $CallTK  EQU     1CH             ;Timer interrupt appendage
= 0021                  C+ $CallDS  EQU     21H             ;DOS function call
= 0023                  C+ $CallCA  EQU     23H             ;Control_break appendage
= 0034                  C+ $CallKR  EQU     34H             ;Gets location In-DOS flag
= 0066                  C+ $CallDI  EQU     66H             ;DOS intercept
= 0067                  C+ $CallTM  EQU     67H             ;Timer appendage recall
= 0001                  C+ $CallTP  EQU     1               ;Trap interrupt
                        C+ ;
= 00CD                  C+ $INT$    EQU     0CDH            ;Interrupt instruction code
= 0001                  C+ $TrapFL  EQU     1               ;Position of trap flag in AH
                        C+ ;
                        C+ ;       Field lengths
                        C+ ;
= 0008                  C+ $$Dir    EQU     8
= 0008                  C+ $$Nam    EQU     8
= 0004                  C+ $$Next   EQU     4
= 000A                  C+ $$Serv   EQU     10
                        C+ ;
                        C+ ;
                        C+ ECT      STRUC
0000  50 43 53 48 41 52 C+ Iden     DB      'PCSHARE2'      ;Identifier of the version
      45 32             C+
0008  43                C+ EC_Dat   DB      'C'             ;Drive for Emulator files (character)
0009  03                C+ EC_Drv   DB      3               ;Same drive numeric     |NEW FIELD|
000A  40                C+ EC_Flg   DB      $ConsPT         ;Flag for log output to console
000B  01                C+ Trce     DB      $False          ;Trace active switch    |NEW FIELD|
000C  01                C+ Cold     DB      $False          ;Cold start completed   |NEW FIELD|
000D  01                C+ EC_Trm   DB      $False          ;Emulator terminated    |NEW FIELD|
000E  0064              C+ Max      DW      100             ;Maximum number records in log file
0010  0000              C+ Next     DW      0               ;RRN for next log record
0012  50 43 53 48 41 52 C+ Dir      DB      'PCSHARE '      ;Name of directory for Emulator files
```

```
      45 20                     C+
001A  50 43 53 48 41 52         C+ Nam      DB      'PCSHARE '       ;Filename for Emulator files
      45 20                     C+
0022     0A [                   C+ Serv     DB      $$Serv DUP(' ') ;Emulator name as a disk server
                20               C+
                    ]            C+
                                 C+
002C  00                         C+ EC_AtrU  DB      $AtrNrm          ;Attributes of User area file      |N F|
002D  00                         C+ EC_AtrS  DB      $AtrNrm          ;Attributes of System area files   |E I|
002E  00                         C+ EC_AtrL  DB      $AtrNrm          ;Attributes of Log file            |W E|
002F  C0                         C+ EC_TBI   DB      $TBintI+$TBrenb ;Transporter int flags             |  L|
0030  0A                         C+ TLtmck   DB      $TickMX          ;Current timer tick count          |  D|
0031  0A                         C+ TLtmrs   DB      $TickMX          ;Value to reset timer tick count   |  S|
0032  01                         C+ KEEPout  DB      $False           ;DOS has been entered flag         |   |
0033  01                         C+ INTdfr   DB      $False           ;Deferred interrupt flag           |   |
0034  0000                       C+ DOSdkO   DW      0                ;Holds location of DOS             |   |
0036  0000                       C+ DOSdkS   DW      0                ;  Critical section (In-DOS) Flag  |   |
0038  0000                       C+ DDRH     DW      0                ;Holds location of request header when
003A  0000                       C+ DDRHS    DW      0                ;   Device Driver called    |NEW FIELD|
003C  0000                       C+ TckO     DW      0                ;Holds location of timer tick int when
003E  0000                       C+ TckS     DW      0                ;   no transporter ints     |NEW FIELD|
0040                             C+ ECT      ENDS
                                 C+ ;
                                 C+ ;-------------------------------------------------------------------------
                                 C+ ;       Structure Name:  Log Request Block                              |
                                 C+ ;       Reference: Program Logic Description, Section 4.27              |
                                 C+ ;-------------------------------------------------------------------------
                                 C+ ;
                                 C+ ;       Associated values
                                 C+ ;
= 0066                           C+ $MAXMSG EQU     102              ;Maximum message length
                                 C+ ;
                                 C+ ;       Log message numbers
                                 C+ ;
= 0001                           C+ $MND00  EQU     1                ;Initialization message
= 0002                           C+ $MNT00  EQU     2
= 0003                           C+ $MNT01  EQU     3
= 0004                           C+ $MNT02  EQU     4
= 0005                           C+ $MNT03  EQU     5
= 0006                           C+ $MNT04  EQU     6
= 0007                           C+ $MNT05  EQU     7
= 0008                           C+ $MNT06  EQU     8
= 0009                           C+ $MND01  EQU     9
= 000A                           C+ $MND02  EQU     10
= 000B                           C+ $MND03  EQU     11
= 000C                           C+ $MND04  EQU     12
= 000D                           C+ $MNT07  EQU     13
= 000E                           C+ $MNT08  EQU     14
= 000F                           C+ $MNT09  EQU     15
= 0010                           C+ $MNT010 EQU     16
= 0011                           C+ $MND05  EQU     17
= 0012                           C+ $MNT011 EQU     18
= 0013                           C+ $MND06  EQU     19               ;ORB missing
= 0014                           C+ $MNT012 EQU     20               ;Bad pipes tables
                                 C+ ;
```

```
                        C+ LRB      STRUC
0000  00                C+ Cde      DB      0             ;Module ID code
0001  00                C+ LR_Len   DB      0             ;Length of the message string
0002                    C+ LRB      ENDS
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ IF (DR$ OR TL$ OR LC$)
                        C+ ;-------------------------------------------------------------------------
                        C+ ;                    Transporter Usage Parameters                      |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
= 0100                  C+ $InitTA EQU      100H          ;Initial transporter CCB address
= 7FFF                  C+ $MaxDly EQU      7FFFH         ;Max loop count for operation complete
= 0010                  C+ $MaxRty EQU      10H           ;Maximum number operation retries
= 000A                  C+ $MaxWte EQU      10            ;Maximum wait time in seconds
= 01FE                  C+ $ProID  EQU      01FEH         ;protocol identifier
= 00CF                  C+ $SndNAD EQU      0CFH          ;Invalid command for NAD reply
= 0211                  C+ $MaxTBD EQU      529           ;Longest data transfer (w/o disk stat)
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;                 Transporter Socket Usage Conventions                 |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
= 00B0                  C+ $SokTD  EQU      0B0H          ;To transmit data
= 0080                  C+ $SokTM  EQU      80H           ;To transmit an NLM
= 00B0                  C+ $SokRM  EQU      80H           ;To receive an NLM
= 00A0                  C+ $SokRC  EQU      0B0H          ;To receive initial command sequence
                        C+ $SokRL  EQU      0A0H          ;To receive remainder of long command
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Name Lookup Message                              |
                        C+ ;       Reference: Program Logic Description, Section 2.2.3               |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ ;       Associated message type codes (most significant byte)
                        C+ ;
= 0000                  C+ $NLHi   EQU      0             ;Hello
= 0001                  C+ $NLAny  EQU      1             ;Any disk server (old protocol)
= 0002                  C+ $NLWho  EQU      2             ;Who are you?
= 0003                  C+ $NLWhr  EQU      3             ;Where are you?
= 0010                  C+ $NLIdn  EQU      10H           ;Identification
= 00FF                  C+ $NLBye  EQU      0FFH          ;Goodbye
                        C+ ;
                        C+ ;       Associated device types of interest to Emulator
                        C+ ;
= 0001                  C+ $NLDsk  EQU      1             ;Disk Server
= 00FF                  C+ $NLDny  EQU      0FFH          ;All Devices
                        C+ ;
                        C+ ;       Associated Name Length
                        C+ ;
= 000A                  C+ $NameLN EQU      10            ;Needed because STRUC is weak
                        C+ ;
                        C+ ;       The structure itself (arithmetic reverses byte order
                        C+ ;       for transporter memory storage)
```

```
                        C+ ;
                        C+ NLM      STRUC
0000  FE01               C+ Pid      DW      $ProID/256 + 256*($ProID AND 0FFH)
0002  0000               C+ Mgt      DW      $NLhi             ;Message Type
0004  0000               C+ Src      DW      0                 ;Station Number of sender
0006  0100               C+ Dev      DW      256*$NLDsk        ;Device Type
0008  50 43 53 48 41 52  C+ NL_Name  DB      'PCSHARE   '      ;Device or user name
      45 20 20 20         C+
0012                    C+ NLM      ENDS
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Receive ICS control area                       |
                        C+ ;       Reference: Corvus OMNINET Programmer' Guide                     |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ RRRCV    STRUC
0000  0000               C+ Lofcm    DW      0                 ;length of the command being sent
0002  0000               C+ ELofRp   DW      0                 ;Expected reply length (w/o status)
0004                    C+ RRRCV    ENDS
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Send Reply control area                        |
                        C+ ;       Reference: Corvus OMNINET Programmer' Guide                     |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ RRSND    STRUC
0000  0000               C+ ALofRp   DW      0                 ;Actual length of reply (incl status)
0002  00                 C+ Dstat    DB      0                 ;Disk status
0003                    C+ RRSND    ENDS
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Transporter Status Table Header                |
                        C+ ;       Reference: Program Logic description, Section 2.3               |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ TSTH     STRUC
0000  00                 C+ Int      DB      0                 ;Interrupt depth indicator
0001  01                 C+ TT_CI    DB      $False            ;Command Interpreter invocation flag
0002  00                 C+ NE       DB      0                 ;Number of entries in TST
0003  00                 C+ TT_NAD   DB      0                 ;Network address of Emulator
0004                    C+ TSTH     ENDS
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Transporter Status Table Entry                 |
                        C+ ;       Reference: Program Logic Description, Section 2.3               |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ TSTE     STRUC
0000  01                 C+ PN       DB      $False            ;ORB pending flag
0001  00                 C+ TT_ST    DB      0                 ;Status code from result record
0002  00                 C+ SK       DB      0                 ;Socket to be used
0003  00                 C+ TT_OP    DB      0                 ;Transporter commmand  !NEW USAGE!
0004  0000               C+ CCB      DW      0                 ;Transporter address of the CCB
0006  0000               C+ RR       DW      0                 ;Transporter address of result record
0008  0000               C+ Dtl      DW      0                 ;Transporter address of data area
```

```
000A  0000                    C+ Dsz      DW      0               ;Length of data area in bytes
000C  0000                    C+ Isr      DW      0               ;Offset address of int service routine
000E  0000                    C+ Isrs     DW      0               ;  Reserved for segment address
0010  0000                    C+ TT_Orb   DW      0               ;Offset address of pending ORB
0012  0000                    C+ TT_Orbs  DW      0               ;  Reserved for segment address
0014                          C+ TSTE     ENDS
                              C+ ;
                              C+ ENDIF
                              C+ ;
                              C+ ENDIF
                              C+ ;
                              C+ IF (DR$ OR TL$)
                              C+ ;-------------------------------------------------------------------------
                              C+ ;       Structure Name:  Command Request Queue Header                  |
                              C+ ;       Reference: Program Logic Description, Section 2.5              |
                              C+ ;-------------------------------------------------------------------------
                              C+ ;
                              C+ ;       Associated queue status indicator values
                              C+ ;
= 0000                        C+ $RQmt    EQU     0               ;Queue is empty
= 0001                        C+ $RQent   EQU     1               ;Queue has entries but is not full
= 00FF                        C+ $RQful   EQU     0FFH            ;Queue is full
                              C+ ;
= 0080                        C+ $RQdse   EQU     80H             ;Disk status error flag bit
                              C+ ;
                              C+ CRQH     STRUC
0000  00                      C+ Qst      DB      $RQmt           ;Queue status indicator
0001  00                      C+ Dsc      DB      0               ;Current disk status code
0002  0000                    C+ End      DW      0               ;Offset address following CRQ
0004  0000                    C+ Top      DW      0               ;Offset address of head entry
0006  0000                    C+ Bot      DW      0               ;Offset address of tail entry
0008                          C+ CRQH     ENDS
                              C+ ;
                              C+ ENDIF
                              C+ ;
                              C+ ENDIF
                              C+ ;
                              C+ IF (DR$ OR TL$ OR LC$)
                              C+ ;-------------------------------------------------------------------------
                              C+ ;           Transporter structures and associated values               |
                              C+ ;           Reference: Corvus OMNINET Programmer's Guide               |
                              C+ ;-------------------------------------------------------------------------
                              C+ ;
                              C+ ;       Transporter Command Control Block
                              C+ ;
                              C+ CCBLK    STRUC
0000  00                      C+ CC_Cmnd  DB      0               ;Command
0001  00                      C+          DB      0               ;fill for high address byte
0002  0000                    C+ RRA      DW      0               ;address Result Record
0004  00                      C+ Sock     DB      0               ;socket to use
0005  00                      C+          DB      0               ;fill for high address byte
0006  0000                    C+ CC_Data  DW      0               ;Data address
0008  0000                    C+ Datl     DW      0               ;Data length
000A  00                      C+ Conln    DB      0               ;Control length
000B  00                      C+ DHost    DB      0               ;Destination host (if write)
```

```
000C                            C+ CCBLK    ENDS
                                C+ ;
                                C+ ;        Transporter Result Record structure
                                C+ ;
                                C+ RRKD     STRUC
0000  FF                        C+ RCode    DB      0FFH            ;Return Code
0001  00                        C+ SHost    DB      0               ;Source Host (if read, 0 write)
0002  0000                      C+ Rcvlen   DW      0               ;Length data received (if read)
0004                            C+ RRKD     ENDS
                                C+ ;
                                C+ ;        Transporter port definitions
                                C+ ;
= 0249                          C+ $RdRAM   EQU     249H            ;To read without incrementing counter
= 024B                          C+ $RdRAMI  EQU     24BH            ;To read and increment counter
= 0248                          C+ $RdStat  EQU     248H            ;To read status
                                C+ ;
= 0248                          C+ $WrCtrH  EQU     248H            ;To set high byte of counter
= 024A                          C+ $WrCtrL  EQU     24AH            ;To set low byte of counter
= 024B                          C+ $WrRAM   EQU     24BH            ;To write data and increment counter
                                C+ ;
= 0249                          C+ $WrStbe  EQU     249H            ;To strobe command address
                                C+ ;
= 024C                          C+ $DiINT   EQU     24CH            ;To disable (disarm) interrupts
= 024D                          C+ $ClINT   EQU     24DH            ;To clear interrupt latch
= 024E                          C+ $EnINT   EQU     24EH            ;To enable (arm) interrupts
= 024F                          C+ $StINT   EQU     24FH            ;To read interrupt status
                                C+ ;
                                C+ ;        Transporter Status Flags
                                C+ ;
= 0080                          C+ $TRedy   EQU     80H             ;Transporter ready
= 0020                          C+ $TRenb   EQU     20H             ;Transporter interrupts enabled
= 0010                          C+ $TRpnd   EQU     10H             ;Transporter interrupt pending
= 0008                          C+ $TRlat   EQU     8               ;Interrupt latch
                                C+ ;
                                C+ ;        Transporter command codes
                                C+ ;
= 0040                          C+ $Send    EQU     40H             ;Send message
= 00F0                          C+ $Setup   EQU     0F0H            ;Setup to receive message
= 0010                          C+ $EndRcv  EQU     10H             ;End receive if no message
= 0020                          C+ $InitT   EQU     20H             ;Initialize transporter
= 0001                          C+ $WhoAmI  EQU     01H             ;Finds transporters NAD
= 0002                          C+ $Echo    EQU     02H             ;Test presence of specified transporter
                                C+ ;$PP     EQU     08H             ;Peek/Poke not used
                                C+ ;
                                C+ ;        Transporter result codes
                                C+ ;
= 0000                          C+ $TrOK    EQU     00H             ;Command completed successfully
= 007F                          C+ $TrST    EQU     7FH             ;Successful transmit retry maximum
= 0080                          C+ $TrNG    EQU     80H             ;Transmit failure, retry count exceeded
= 0081                          C+ $TrDL    EQU     81H             ;Data too long for receive buffer
= 0082                          C+ $TrNS    EQU     82H             ;Receiver's socket not set up
= 0083                          C+ $TrCL    EQU     83H             ;Control length mismatch
= 0084                          C+ $TrIS    EQU     84H             ;Invalid socket
= 0085                          C+ $TrNR    EQU     85H             ;receive socket in use
= 0086                          C+ $TrIT    EQU     86H             ;invalid transporter address
```

```
= 00C0                  C+ $TrEA   EQU    0C0H            ;Echo command acknowledged
= 00FE                  C+ $TrSS   EQU    0FEH            ;Receive socket setup successful
                        C+ ;
                        C+ ;       Transporter usage values
                        C+ ;
= 0040                  C+ $NADok  EQU    64              ;First invalid NAD value
= 00FF                  C+ $AllNAD EQU    0FFH            ;NAD for broadcast
= 00FF                  C+ $TrSU   EQU    0FFH            ;Initialization value for result code
                        C+ ;
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ IF (DM$ or DR$ or EI$)
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Input/Output Request Block                    |
                        C+ ;       Reference: Program Logic Description, section 4.6              |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ ;       Associated completion codes
                        C+ ;
                        C+ ;
= 0000                  C+ $SUCCESS EQU   0
= 00FF                  C+ $FAILURE EQU   0FFH
                        C+ ;
                        C+ ;       The structure itself
                        C+ ;
                        C+ IORB     STRUC
0000  00                C+ IO_FLG   DB    0               ;FLAGS:
                        C+ ;                               00-OPERATION SUCCESSFUL
                        C+ ;                               FF-OPERATION NOT SUCCESSFUL
0001  01                C+ IO_Cls   DB    $False          ;Close file indicator
0002     02 [           C+          DB    2  DUP(0)       ;RESERVED
               00       C+
                   ]    C+
                        C+
0004  0000              C+ NUM      DW    0               ;NUMBER OF 128 BYTE RECORDS
0006  0000              C+ DSPL     DW    0               ;RRN OF FIRST 128 BYTE RECORD LSB
0008  0000              C+ DSPM     DW    0               ;RRN OF FIRST 128 BYTE RECORD MSB
000A  0000              C+ ADR      DW    0               ;ADDRESS OF BUFFER (OFFSET)
000C  0000              C+          DW    0               ;ADDRESS OF BUFFER (BASE)
000E  0000              C+ FCB      DW    0               ;ADDRESS OF FCB (OFFSET)
0010  0000              C+          DW    0               ;ADDRESS OF FCB (BASE)
0012                    C+ IORB     ENDS
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ ENDIF
                        C+ ENDIF
                        C+ ;
```

```
                        C+ ENDIF
                        C+
                            ;
                            ;*****************************************************************************
                            ;                       Segment Usage Definition
                            ;*****************************************************************************
                            ;
                            PCS  GROUP  PC_CODE,PC_DATA,PC_INIT
                                    ASSUME  CS:PCS,DS:PC_DATA,ES:PC_DATA
                            ;
                            ;*****************************************************************************
                            ;                       External Symbol Definitions                          |
                            ;*****************************************************************************
                            ;
                                    EXTRN   TT:BYTE                 ;TST header
                                    EXTRN   TTbgn:BYTE              ;Initial non-local TST entry
                                    EXTRN   TTend:BYTE              ;End of TST entries
                                    EXTRN   TrmNAD:BYTE             ;TST entry to send NAD
                                    EXTRN   RcvICS:BYTE             ;TST entry for ICS receive
                                    EXTRN   RcvNLM:BYTE             ;TST entry for NLM receive
                                    EXTRN   VarOP:BYTE              ;TST entry for variable ops
                                    EXTRN   IDmsg:BYTE              ;NL message image
                                    EXTRN   MND001:WORD             ;Initialization message insert
                                    EXTRN   IO:BYTE                 ;Internal IORB
                                    EXTRN   TL_BYT_OUT:NEAR         ;Send byte to transporter
                                    EXTRN   TL_WRD_OUT:NEAR         ;Send word to transporter
                                    EXTRN   TL_DTA_OUT:NEAR         ;Send data to transporter
                                    EXTRN   TL_DO_CMND:NEAR         ;Execute transporter command
                                    EXTRN   TL_SET_RECV:NEAR        ;Set up to receive message
                                    EXTRN   TL_SEND_NLM:NEAR        ;Send an NL message
                                    EXTRN   DM_ERR_LOG:NEAR         ;Log routine
                                    EXTRN   TL_INT_DRVR:NEAR        ;Main interrupt driver
                                    EXTRN   TL_INT_TEST:NEAR        ;Entry to hardware int driver
                                    EXTRN   TL_INT_DTME:NEAR        ;Timer tick interrupt driver
                            ;
                            ;*****************************************************************************
                            ;
0000                        PC_CODE  SEGMENT  PUBLIC 'CODE'
                            ;
0000                        PC_START LABEL NEAR             ;For starting address
                            ;
                            ;*****************************************************************************
                            ;       Allocated Data:  Note that for this file only there is               |
                            ;                        data allocated in the code segment                  |
                            ;*****************************************************************************
                            ;
                            ;-----------------------------------------------------------------------------
                            ;       Device Driver Header
                            ;
0000  FF FF FF FF           PCSH    DDH     (,,DR_STR,DR_INT)     ;Offsets are to  device strategy
0004  8000
0006  0043 R
0008  0050 R
000A  50 43 53 4D 48 4F
      4F 4B
```

```
                                ;                                       and interrupt routines
                                ;
                                ;-----------------------------------------------------------------------
                                ;       Corvus Utilities Transfer Table
                                ;
= 0012                          UTIL_HOOK EQU THIS BYTE
0012  E9 00E9 R                         JMP NEAR PTR UTIL_INIT          ;Cold Start Entry
0015  0000                      UTIL_CS DW      0                       ;  place for segment
0017  E9 00E9 R                         JMP NEAR PTR UTIL_INIT          ;Warm Start entry
001A  0000                              DW      0                       ;  place for segment
001C  E9 00E6 R                         JMP NEAR PTR UTIL_IO            ;Entry to call I/O Services
001F  0000                              DW      0                       ;  place for segment
0021  E9 00E8 R                         JMP NEAR PTR UTIL_DMMY          ;Entry to dummy return
0024  0000                              DW      0                       ;  place for segment
= 0026                          UTIL_HEND EQU THIS BYTE
                                ;
                                ;-----------------------------------------------------------------------
                                ;       Structure with offsets to Key Emulator locations
                                ;       NOTE:  This must immediately follow the UTIL_HOOK table
                                ;
0026  0000 R                    DDkeys  KEYS    (OFFSET PCS:EC) ;Filled in from table values or
0028  0000
002A  0000
002C  0000
                                ;                                       by driver initialization
                                ;
                                ;-----------------------------------------------------------------------
                                ;       Device driver function dispatching table
                                ;
= 002E                          DDtrt   EQU     THIS WORD
002E  0094 R                            DW      DDinit          ;Device driver initialization
0030  0086 R                            DW      DDxit           ;Media Check is null
0032  0086 R                            DW      DDxit           ;Build BPB is null
0034  0086 R                            DW      DDxit           ;IOCTL not supported
0036  00A7 R                            DW      DDUtil          ;Input gets utility pointer
0038  00BA R                            DW      DDUtib          ;ND Input gets one byte of ptr
003A  0086 R                            DW      DDxit           ;Input status is not busy
003C  00DE R                            DW      DDUfx           ;Any output resets the
003E  00DE R                            DW      DDUfx           ;  ND Input byte pointer
= 0040                          DDtrte  EQU     THIS WORD       ;All others are null, so we
                                ;                                cut off the table here
                                ;
                                ;-----------------------------------------------------------------------
                                ;       Data required for device driver functions
                                ;
0040  0000                      DDUctr  DW      0               ;Byte position next returned
0042  01                        DDIdun  DB      $False          ;Driver initialization flag
                                ;
0043                            PC_CODE  ENDS
                                ;
                                ;*****************************************************************************
                                ;                       Allocated data (Data Segment)                       |
                                ;*****************************************************************************
```

```
                                ;
0000                            PC_DATA  SEGMENT  PUBLIC 'DATA'
                                ;
= 0000                          Dta_Bgn EQU     THIS BYTE       ;Start of Data segment area
                                ;-----------------------------------------------------------------------
                                ;       Emulator Communication Table
                                ;
0000  50 43 53 48 41 52         EC      ECT     ()              ;STRUC values may be modified
      45 32
0008  43
0009  03
000A  40
000B  01
000C  01
000D  01
000E  0064
0010  0000
0012  50 43 53 48 41 52
      45 20
001A  50 43 53 48 41 52
      45 20
0022     0A [
                20
                    ]
002C  00
002D  00
002E  00
002F  C0
0030  0A
0031  0A
0032  01
0033  01
0034  0000
0036  0000
0038  0000
003A  0000
003C  0000
003E  0000

                                ;
                                ;-----------------------------------------------------------------------
                                ;       Prototype CCB for transporter initialization
                                ;
0040  00                        SetupT  CCBLK   ()              ;Filled in by TL-INIT-TLD
0041  00
0042  0000
0044  00
0045  00
0046  0000
0048  0000
004A  00
004B  00

                                ;
004C                            PC_DATA  ENDS
```

```
                        ;
                        ;********************************************************************************
                        ;                   Device Driver and Corvus Utility Routines                   |
                        ;********************************************************************************
                        ;
0043                    PC_CODE  SEGMENT  PUBLIC 'CODE'
                        ;
                        ;********************************************************************************
                        ;Module name:  Device Driver Routines                                            |
                        ;                                                                               |
                        ;Version:  2.0                                                                  |
                        ;                                                                               |
                        ;Last Update: 23 December 1983                                                  |
                        ;                                                                               |
                        ;Function:  The Device Driver routines implement the installed device           |
                        ;           driver functions required by PC DOS 2.0.                            |
                        ;                                                                               |
                        ;           DR-STR is the entry point to the device strategy routine            |
                        ;              whiose only function is to save the location of the              |
                        ;              request header.                                                  |
                        ;                                                                               |
                        ;           DR-INT is the entry point to the device interrupt routine.          |
                        ;              The INIT function carries out the Emulator cold start            |
                        ;              procedure.  The two INPUT device functions return a              |
                        ;              long pointer to the Corvus utility jump table (which is          |
                        ;              the UTILHOOK function).  All other device functions are          |
                        ;              null operations.                                                 |
                        ;                                                                               |
                        ;Procedure:  See individual routines                                            |
                        ;                                                                               |
                        ;Called by:  DOS using device driver protocol                                   |
                        ;                                                                               |
                        ;Routines called:  EI-INIT                                                      |
                        ;                                                                               |
                        ;Input:  ES:(BX) = Location request header (PC-DEV-STR only)                    |
                        ;                                                                               |
                        ;Output:  Modified request header (PC-DEV-INT only)                             |
                        ;                                                                               |
                        ;Error procedures:  None                                                        |
                        ;                                                                               |
                        ;Written by: R.B. Talmadge, Computer Technology Ltd                             |
                        ;                                                                               |
                        ;Updates:  None                                                                 |
                        ;                                                                               |
                        ;********************************************************************************
                        ;
0043                    DR_DEV_DRVR  PROC  FAR
                        ;
0043                    DR_STR  LABEL  NEAR
                        ;
0043  56                        PUSH    SI                  ;Get offset for
0044  BE 0038 R                MOV     SI,OFFSET PCS:EC.DDRH  ;address storage
0047  2E: 89 1C                MOV     CS:[SI],BX          ;Save full 8088 address
004A  2E: 8C 44 02              MOV     CS:[SI+2],ES        ;  of request header
004E  5E                        POP     SI                  ;  and return to caller
```

```
004F  CB                                RET
                                ;
0050                            DR_INT  LABEL  NEAR
                                ;
                                        ENTER   Dta_Bgn          ;Save registers and set environment
0050  50                      +         PUSH    AX
0051  53                      +         PUSH    BX
0052  51                      +         PUSH    CX
0053  52                      +         PUSH    DX
0054  56                      +         PUSH    SI
0055  57                      +         PUSH    DI
0056  55                      +         PUSH    BP
0057  1E                      +         PUSH    DS
0058  06                      +         PUSH    ES
0059  8C C8                   +         MOV     AX,CS
005B  BB 0000 R                +         MOV     BX,OFFSET PCS:Dta_Bgn
005E  D1 EB                    +         SHR     BX,1
0060  D1 EB                    +         SHR     BX,1
0062  D1 EB                    +         SHR     BX,1
0064  D1 EB                    +         SHR     BX,1
0066  03 C3                   +         ADD     AX,BX
0068  8E D8                   +         MOV     DS,AX
006A  8E C0                   +         MOV     ES,AX
006C  8B EC                   +         MOV     BP,SP
                                ;
006E  C5 1E 0038 R                      LDS     BX,DWORD PTR EC.DDRH    ;Get address of request header
0072  32 E4                             XOR     AH,AH                   ;  and fetch operation code
0074  8A 47 02                          MOV     AL,[BX].RQopn
                                ;
0077  D0 C0                             ROL     AL,1                    ;Transfer table offset
0079  3D 0012                           CMP     AX,OFFSET DDtrte - OFFSET DDtrt
007C  7D 08                             JGE     DDxit                   ;Exit if out of range
007E  BE 002E R                         MOV     SI,OFFSET DDtrt         ;Position to transfer
0081  03 F0                             ADD     SI,AX                   ;  table entry and call
0083  2E: FF 14                         CALL    WORD PTR CS:[SI]        ;  function routine
                                ;
                                ;       Reentry point here for all device driver function routines.
                                ;       The completion bit is set in the request header status field,
                                ;       interrupts are turned back on, and the routine exits.
                                ;
0086  80 4F 03 01               DDxit:  OR      BYTE PTR [BX].RQstat,$RQdun
                                        EXIT    NOINT           ;Go back to caller
008A  07                      +         POP     ES
008B  1F                      +         POP     DS
008C  5D                      +         POP     BP
008D  5F                      +         POP     DI
008E  5E                      +         POP     SI
008F  5A                      +         POP     DX
0090  59                      +         POP     CX
0091  5B                      +         POP     BX
0092  58                      +         POP     AX
0093  CB                      +         RET
                                ;
0094                            DR_DEV_DRVR  ENDP
                                ;
```

```
0094                            DR_DEV_FNS  PROC  NEAR
                                ;
                                ;       Enter here for device driver initialization.  The EI_ONE
                                ;       one-time initialization routine is called if it has not
                                ;       previously been called.  The transporter is not initialized
                                ;       by this routine.  It will be initialized by a call to either
                                ;       of the UTIL_HOOK start entries (cold or warm).
                                ;
0094  2E: 80 3E 0042 R 01       DDinit: CMP     CS:DDIdun,$False        ;If this is not the
009A  74 01                             JE      DDini1                  ;  first entry, do not
009C  C3                                RET                             ;  take any action
                                ;
009D  E8 0000 R                 DDini1: CALL NEAR PTR EI_ONE            ;Go do first time work
00A0  2E: C6 06 0042 R 00               MOV     CS:DDIdun,$True         ;Assure no return here
00A6  C3                                RET                             ;Return from whence we came
                                ;
                                ;       Enter here for the Input function.  The caller is sent
                                ;       the full 8088 segmented address of the utility transfer
                                ;       table as the first four bytes of the input buffer
                                ;
00A7  1E                        DDUtil: PUSH    DS                      ;Save segment
                                ;
00A8  C5 7F 0E                          LDS     DI,[BX].RQrtn           ;Buffer address
00AB  BE 0012 R                         MOV     SI,OFFSET UTIL_HOOK     ;Offset of jump table
00AE  89 35                             MOV     [DI],SI                 ;Store offset, then
00B0  8C 4D 02                          MOV     2[DI],CS                ;  segment address
                                ;
00B3  1F                                POP     DS                      ;Restore segment
00B4  C7 47 12 0004                     MOV     [BX].RQct,4             ;Set return count
00B9  C3                                RET                             ;  and exit
                                ;
                                ;       Enter here for Non-destructive input function.  The caller
                                ;       is returned one byte of the address of the utility transfer
                                ;       table.  After four entries the progression starts over.
                                ;
00BA  2E: 8B 0E 0040 R          DDUtib: MOV     CX,CS:DDUctr                    ;Fetch current position
00BF  B8 0012 R                         MOV     AX,OFFSET UTIL_HOOK             ;Get offset or
00C2  F6 C1 02                          TEST    CL,2                            ;  Segment address
00C5  74 02                             JZ      DDUtic
00C7  8C C8                             MOV     AX,CS
                                ;
00C9  F6 C1 01                  DDUtic: TEST    CL,1                            ;Position byte
00CC  74 02                             JZ      DDUtid                          ;  to low order
00CE  8A C4                             MOV     AL,AH
00D0  88 47 0D                  DDUtid: MOV     [BX].RQdta,AL                   ;Store byte in request hdr
                                ;
00D3  41                                INC     CX              ;increment to next
00D4  81 E1 0003                        AND     CX,3            ;  position modulo 3
00D8  2E: 89 0E 0040 R                  MOV     CS:DDUctr,CX    ;Store new value
00DD  C3                                RET                     ;   and exit
                                ;
                                ;       Enter here for any output function.  The Non-destructive input
                                ;       position indicator (field DDUctr) is set to zero.
                                ;
00DE  2E: C7 06 0040 R 0000     DDUfx:  MOV     CS:DDUctr,0     ;Fix up
```

```
00E5  C3                        RET                     ;  and exit
                        ;
00E6                    DR_DEV_FNS  ENDP
                        ;
                        ;*****************************************************************************
                        ;Module name:  Corvus Utility Routines                                       |
                        ;                                                                            |
                        ;Version:  2.0                                                               |
                        ;                                                                            |
                        ;Last Update: 7 December 1983                                                |
                        ;                                                                            |
                        ;Function:  The Corvus Utility routines are called by those Corvus           |
                        ;           Corvus routines that require access to the Local Command         |
                        ;           I/O services.                                                    |
                        ;                                                                            |
                        ;           UTIL_IO executes the proper interrupt to call the Local          |
                        ;              Command service, and exits on return                          |
                        ;                                                                            |
                        ;           UTIL_INIT call TLD initialization to do a warm start             |
                        ;                                                                            |
                        ;           UTIL_DMMY does nothing                                           |
                        ;                                                                            |
                        ;Procedure:  See Program Logic Description, Section 8.1                      |
                        ;                                                                            |
                        ;Called by:  Corvus utilities                                                |
                        ;                                                                            |
                        ;Routines called:  TL-INIT-TLD                                               |
                        ;                                                                            |
                        ;Input:  Registers data as required by Local Command I/O                     |
                        ;                                                                            |
                        ;Output:  Register data set by Local Command I/O                             |
                        ;                                                                            |
                        ;Error procedures:  None                                                     |
                        ;                                                                            |
                        ;Written by: R.B. Talmadge, Computer Technology Ltd                          |
                        ;                                                                            |
                        ;Updates:  None                                                              |
                        ;*****************************************************************************
                        ;
00E6                    DR_DEV_UTIL  PROC  FAR
                        ;
00E6                    UTIL_IO  LABEL  NEAR
                        ;
00E6  CD 0A                     INT     $CallEM         ;Call the Emulator
                        ;
00E8                    UTIL_DMMY  LABEL  NEAR
                        ;
00E8  CB                        RET                     ;Exit
                        ;
00E9                    UTIL_INIT  LABEL  NEAR
                        ;
                                ENTER   Dta_Bgn,,Noregs         ;Set environment
00E9  1E                +       PUSH    DS
00EA  06                +       PUSH    ES
```

```
00EB  8C C8                   +          MOV     AX,CS
00ED  BB 0000 R               +          MOV     BX,OFFSET PCS:Dta_Bgn
00F0  D1 EB                   +          SHR     BX,1
00F2  D1 EB                   +          SHR     BX,1
00F4  D1 EB                   +          SHR     BX,1
00F6  D1 EB                   +          SHR     BX,1
00F8  03 C3                   +          ADD     AX,BX
00FA  8E D8                   +          MOV     DS,AX
00FC  8E C0                   +          MOV     ES,AX
                                ;
00FE  E8 013C R                 INIxt:   CALL NEAR PTR TL_INIT_TLD     ;Initialize the TLD
                                ;
0101  0A C0                              OR      AL,AL           ;Indicate NG if transporter
0103  75 0A                              JNZ     INIxt0          ;  not enabled with clear latch
0105  B8 4B4F                            MOV     AX,'KO'         ;Test value of transporter NAD
0108  80 3E 0003 E 40                    CMP     TT.TT_NAD,$NADok
010D  7E 03                              JNG     INIxt1
010F  B8 474E                   INIxt0:  MOV     AX,'GN'
0112  A3 0000 E                 INIxt1:  MOV     MND001,AX          ;Set message accordingly
                                         LOG     $MND00             ;  and output the message
0115  50                      +          PUSH    AX
0116  B0 01                   +          MOV     AL,$MND00
0118  E8 0000 E               +          CALL    DM_ERR_LOG
011B  58                      +          POP     AX
                                ;
                                         EXIT    NOINT           ;Return to caller
011C  07                      +          POP     ES
011D  1F                      +          POP     DS
011E  CB                      +          RET
                                ;
011F                            UTIL_INT_TEST  LABEL  NEAR
                                ;
                                ;-------------------------------------------------------------------------
                                ;       Enter here to call the TL_INT_DRVR at its primary interrupt      |
                                ;       entry point TL_INT_TEST.  The call will invoke the main loop     |
                                ;       of the interupt driver so that the Emulator can be run as a      |
                                ;       subroutine rather than as an interrupt driven device routine.    |
                                ;-------------------------------------------------------------------------
                                ;
                                         ENTER   Dta_Bgn,,Noregs         ;Set environment
011F  1E                      +          PUSH    DS
0120  06                      +          PUSH    ES
0121  8C C8                   +          MOV     AX,CS
0123  BB 0000 R               +          MOV     BX,OFFSET PCS:Dta_Bgn
0126  D1 EB                   +          SHR     BX,1
0128  D1 EB                   +          SHR     BX,1
012A  D1 EB                   +          SHR     BX,1
012C  D1 EB                   +          SHR     BX,1
012E  03 C3                   +          ADD     AX,BX
0130  8E D8                   +          MOV     DS,AX
0132  8E C0                   +          MOV     ES,AX
                                ;
0134  FA                                 CLI                     ;Disable interrupts
0135  E8 0000 E                          CALL    TL_INT_TEST     ;Call the interrupt driver
0138  FB                                 STI                     ;Assure interrupts enabled
```

```
                                ;
                                        EXIT    NOINT           ;Return to caller
0139  07                      +         POP     ES
013A  1F                      +         POP     DS
013B  CB                      +         RET
                                ;
013C                            DR_DEV_UTIL  ENDP
                                ;
                                ;*****************************************************************************
                                ;Module name:  TLD initialization (TL-INIT-TLD)                               |
                                ;                                                                             |
                                ;Version:  2.0                                                                |
                                ;                                                                             |
                                ;Last Update: 15 January 1984                                                 |
                                ;                                                                             |
                                ;Function:  The TLD initialization routine is one of the routines             |
                                ;           which comprise Emulator initialization.  Its task is to           |
                                ;           set up transporter memory and the TST, initialize the             |
                                ;           transporter, start transporter interrupts, and start              |
                                ;           reception of commands and Name Lookup messages.                   |
                                ;                                                                             |
                                ;Procedure:  Program Logic Description, Section 2.8                           |
                                ;                                                                             |
                                ;Called by:  EI-INIT                                                          |
                                ;                                                                             |
                                ;Routines called:  TL-DO-CMND                                                 |
                                ;                                                                             |
                                ;Input:  None                                                                 |
                                ;                                                                             |
                                ;Output:  (AL) = Successful completion indicator                              |
                                ;                                                                             |
                                ;Error procedures:  None                                                      |
                                ;                                                                             |
                                ;Written by: R.B. Talmadge, Computer Technology Ltd                           |
                                ;                                                                             |
                                ;Updates:  None                                                               |
                                ;*****************************************************************************
                                ;
013C                            TL_INIT_TLD  PROC  NEAR
                                ;
013C  FA                                CLI                     ;Disable 8088 interrupts
                                ;
                                ;       The first step is to disable transporter interrupts
                                ;       if interrupts are installed on the board
                                ;
013D  F6 06 002F R 80                   TEST    EC.EC_TBI,$TBintI
0142  74 04                             JZ      ITLgo           ;If transporter interrupts
0144  BA 024C                           MOV     DX,$DiINT       ;  installed, disable
0147  EE                                OUT     DX,AL           ;  interrupts
                                ;
                                ;       Next, the CCB and RR images and all constant data are
                                ;       inserted into transporter memory
                                ;
0148  BF 0000 E                 ITLgo:  MOV     DI,OFFSET TTbgn     ;Starting offset of TST entries
```

```
                                ;
014B  BE 0040 R                 ITLlp:  MOV     SI,OFFSET SetupT        ;Starting offset CCB image
014E  8A 45 03                          MOV     AL,[DI].TT_OP           ;Operation code
0151  88 04                             MOV     [SI].CC_Cmnd,AL         ;  to CCB image
0153  8A 45 02                          MOV     AL,[DI].SK              ;Socket number
0156  88 44 04                          MOV     [SI].Sock,AL            ;  to CCB image
0159  8B 45 0A                          MOV     AX,[DI].Dsz             ;Now data length in reverse
015C  86 C4                             XCHG    AL,AH                   ;  byte order (as are all two
015E  89 44 08                          MOV     [SI].Dat1,AX            ;  byte quantities for xporter)
                                ;
0161  8B 45 06                          MOV     AX,[DI].RR              ;Next the
0164  86 C4                             XCHG    AL,AH                   ;  result record
0166  89 44 02                          MOV     [SI].RRA,AX             ;  address
0169  8B 5D 08                          MOV     BX,[DI].Dtl             ;Then the
016C  86 FB                             XCHG    BH,BL                   ;  Data address
016E  89 5C 06                          MOV     [SI].CC_Data,BX
                                ;
0171  86 FB                             XCHG    BH,BL                   ;Compute user control
0173  86 E0                             XCHG    AH,AL                   ;   area length
0175  2B D8                             SUB     BX,AX
0177  83 EB 04                          SUB     BX,TYPE RRKD
017A  88 5C 0A                          MOV     [SI].Conln,BL           ;Store into CCB image
                                ;
017D  B9 000C                           MOV     CX,TYPE CCBLK           ;Set size of CCB
0180  8B 5D 04                          MOV     BX,[DI].CCB             ;Fetch address of CCB
0183  E8 0000 E                         CALL    TL_DTA_OUT              ;Insert CCB image into xporter
                                ;
0186  8B 5D 06                          MOV     BX,[DI].RR              ;Initialize result code
0189  8A 4D 01                          MOV     CL,[DI].TT_ST           ;  to be equal to value
018C  E8 0000 E                         CALL    TL_BYT_OUT              ;  now in TST entry
                                ;
018F  83 C7 14                          ADD     DI,TYPE TSTE            ;Position to next TST
0192  81 FF 0000 E                      CMP     DI,OFFSET TTend         ;  entry and go back
0196  72 B3                             JB      ITLlp                   ;  to process if not done
                                ;
                                ;       Initialize data area for reply to NAD requests
                                ;
0198  8B 36 0006 E                      MOV     SI,TrmNAD.RR            ;Compute location of User
019C  83 C6 04                          ADD     SI,TYPE RRKD            ;  Control area for TrmNAD
019F  8D 1C                             LEA     BX,[SI].ALofRP          ;Position to reply
01A1  B9 0001                           MOV     CX,1                    ;  length and send
01A4  E8 0000 E                         CALL    TL_WRD_OUT              ;  to transporter
01A7  8D 5C 02                          LEA     BX,[SI].Dstat           ;Position to disk status
01AA  B1 CF                             MOV     CL,$SndNAD              ;  and send invalid command
01AC  E8 0000 E                         CALL    TL_BYT_OUT              ;  to transporter
                                ;
                                ;       Initialize the transporter and save Emulator NAD
                                ;
01AF  8B 1E 0006 E                      MOV     BX,VarOP.RR     ;Fetch transporter addresses
01B3  8B 0E 0004 E                      MOV     CX,VarOP.CCB    ;  for variable operation CCB
01B7  E8 0000 E                         CALL    TL_DO_CMND      ;  and go initialize transporter
01BA  A2 0003 E                         MOV     TT.TT_NAD,AL    ;Save NAD in TST header
01BD  A2 0005 E                         MOV     BYTE PTR IDmsg.Src+1,AL   ;and NL message image
                                ;
                                ;       Enable transporter interrupts if installed and if board
```

```
                                ;       is to run enabled
                                ;
01C0  A0 002F R                         MOV     AL,EC.EC_TBI    ;Fetch transporter run type
01C3  A8 80                             TEST    AL,$TBintI
01C5  74 18                             JZ      ITLp0           ;Don't enable if not installed
01C7  A8 40                             TEST    AL,$TBrenb
01C9  74 14                             JZ      ITLp0           ;  or if not to run enabled
                                ;
01CB  E4 21                             IN      AL,$P82op1      ;Fetch 8259A interrupt
01CD  24 FA                             AND     AL,$P82enb      ;  and enable the level
01CF  E6 21                             OUT     $P82op1,AL      ;  for the transporter
01D1  BA 024E                           MOV     DX,$EnINT       ;Fetch port number
01D4  EE                                OUT     DX,AL           ;  and send the enable
01D5  BA 024D                           MOV     DX,$ClINT       ;Assure the interrupt
01D8  EE                                OUT     DX,AL           ;  latch is clear
01D9  BA 024F                           MOV     DX,$StINT       ;Now fetch status
01DC  EC                                IN      AL,DX           ;  of transporter
01DD  24 28                             AND     AL,$TRenb+$TRlat
                                ;
                                ;       Start reception of messages
                                ;
01DF  50                        ITLp0:  PUSH    AX                      ;Save transporter status
01E0  BE 0000 E                         MOV     SI,OFFSET RcvICS        ;Set offset TST entry ICS receive
01E3  B0 01                             MOV     AL,$False               ;  set for no ORB,
01E5  E8 0000 E                         CALL    TL_SET_RECV             ;  request start of reception
                                ;
01E8  BE 0000 E                         MOV     SI,OFFSET RcvNLM        ;Set offset TST entry NLM receive
01EB  B0 01                             MOV     AL,$False               ;  set for no ORB,
01ED  E8 0000 E                         CALL    TL_SET_RECV             ;  request start of reception
                                ;
                                ;       Send out a 'Hello' Name Lookup message, then reset the
                                ;       NL message image to send Identification messages
                                ;
01F0  BE 0022 R                         MOV     SI,OFFSET EC.Serv           ;Fetch Emulator name
01F3  BF 0008 E                         MOV     DI,OFFSET IDmsg.NL_Name     ;  from ECT to
01F6  B9 000A                           MOV     CX,$$Serv                   ;  the NL message image
01F9  FC                                CLD
01FA  F3/ A4                            REP MOVSB
                                ;
01FC  B1 FF                             MOV     CL,$AllNAD              ;Use broadcast NAD
01FE  BE 0000 E                         MOV     SI,OFFSET IDmsg         ;  and NL message image
0201  E8 0000 E                         CALL    TL_SEND_NLM             ;Send 'Hello' and start
                                ;                                       ;  NL message reception
0204  B8 0010                           MOV     AX,$NLIdn
0207  A3 0002 E                         MOV     IDmsg.Mgt,AX            ;Reset for Iden message
                                ;
020A  C6 06 000C R 00                   MOV BYTE PTR EC.Cold,$True      ;Indicate cold start over
020F  58                                POP     AX                      ;Fetch transporter status
0210  C3                                RET                         ;And that's all there is to do
                                ;
0211                            TL_INIT_TLD  ENDP
                                ;
0211                            PC_CODE  ENDS
                                ;
                                ;*************************************************************************
```

```
                        ;       This is the entry point for the first time initialization       |
                        ;       procedure.  This code is contained in the special segment       |
                        ;       PC_INIT, which is deleted by PC DOS after return from driver    |
                        ;       initialization.                                                 |
                        ;*******************************************************************************
                        ;
0000                    PC_INIT  SEGMENT  MEMORY 'LAST'
                        ;
0000                    EI_ONE  PROC  NEAR
                        ;
                        ;       Set code segment value into UTIL_HOOK jump table
                        ;       and offsets into DDkeys table for EI-INIT
                        ;
0000  BE 0015 R                 MOV     SI,OFFSET UTIL_CS       ;First location
0003  BF 0026 R                 MOV     DI,OFFSET UTIL_HEND     ;End location
0006  2E: 8C 0C         EI_ins: MOV     CS:[SI],CS              ;Insert segment value
0009  83 C6 05                 ADD     SI,((TYPE UTIL_CS) + OFFSET UTIL_CS - OFFSET UTIL_HOOK)
000C  3B F7                     CMP     SI,DI
000E  7C F6                     JL      EI_ins                  ;Back if not done
                        ;
0010  2E: C7 06 0026 R 0000 R   MOV     CS:DDkeys.KY_EC,OFFSET PCS:EC           ;ECT
0017  2E: C7 06 0028 R 011F R   MOV     CS:DDkeys.KY_EXT,OFFSET UTIL_INT_TEST   ;Call HW drvr
001E  2E: C7 06 002A R 0000 E   MOV     CS:DDkeys.KY_INT,OFFSET TL_INT_DRVR     ;Int Drvr
0025  2E: C7 06 002C R 0000 E   MOV     CS:DDkeys.KY_DTM,OFFSET TL_INT_DTME     ;Timer int
                        ;
                        ;       Move end location into request header for return
                        ;       information to DOS
                        ;
002C  C7 47 0E 0000 R           MOV  WORD PTR [BX].RQrtn,OFFSET PCS:EI_ONE
0031  8C 4F 10                  MOV  WORD PTR 2[BX].RQrtn,CS
                        ;
0034  C3                        RET                     ;Return from whence we came
                        ;
0035                    EI_ONE  ENDP
                        ;
0035                    PC_INIT ENDS
                        ;
                        ;*******************************************************************************
                        ;               End of the Assembly housekeeping goes here              |
                        ;*******************************************************************************
                        ;
                        ;
                                PUBLIC  EC              ;Emulator Communication table
                                PUBLIC  Dta_Bgn         ;Start of Emulator Data area
                        ;
                                END     PC_START
```

Macros:

Name	Length
COMPARE.	000F
DISABLE.	0003
ENABLE	0003
ENTER.	0013
EXIT	0007
EYE.	0001
FILLIT	0011
FIV.	0002
GEN_START.	0002
LOG.	0005
MOVEB.	000F
MOVEONE.	000E
RESTORE.	0001
SCANB.	0012
SELEXIT.	0002
SELREST.	000D
SELSAVE.	0018
SIV.	0003
START.	001A
STATUS	0001
TENTER	0003
TEXIT.	0002
TIME	0001
TRACE.	0009

Structures and records:

Name	Width / Shift	# fields / Width	Mask	Initial
CCBLK.	000C	0009		
CC_CMND.	0000			
RRA.	0002			
SOCK	0004			
CC_DATA.	0006			
DATL	0008			
CONLN.	000A			
DHOST.	000B			
CRQH	0008	0005		
QST.	0000			
DSC.	0001			
END.	0002			
TOP.	0004			
BOT.	0006			
DDH.	0012	0005		
NDEV	0000			
ATTR	0004			
STRTN.	0006			
DVRTN.	0008			
DVNME.	000A			
ECT.	0040	001A		
IDEN	0000			

```
EC_DAT . . . . . .  0008
EC_DRV . . . . . .  0009
EC_FLG . . . . . .  000A
TRCE . . . . . .  000B
COLD . . . . . .  000C
EC_TRM . . . . . .  000D
MAX. . . . . .  000E
NEXT . . . . . .  0010
DIR. . . . . .  0012
NAM. . . . . .  001A
SERV . . . . . .  0022
EC

D:PCSCH1  .LST Canceled by operator
```

```
                        ;
                        PAGE 60,132
                        ;
                        ;*****************************************************************************
                        ;                       Filename is PCSCH2A.ASM                               |
                        ;*****************************************************************************
                        ; This file is the first of two which contain the code for all of the         |
                        ; routines described in Sections 2.1 through 2.7 of the Program Logic         |
                        ; Description, OMNINET PCShare Disk Server Emulator, Version 2.0, dated       |
                        ; 7 November 1983.  The file also contains allocation statements for          |
                        ; data structures managed by the Transporter Logical Device.                  |
                        ;*****************************************************************************
                        ;
                                ENDIF
                        ;
                        ;*****************************************************************************
                        ;
                                GEN_START   <TL>        ;TLD structures
                      +         .LALL
                      +         ENDIF
                      +         ENDIF
                      +         ENDIF
                      +         IFIDN   <TRACE>,<TRACE>
= 0001                + TRACE$  =       1
                      +         ENDIF
                      +         ENDIF
                       +        ENDIF
                       +        ENDIF
                       +        ENDIF
                       +        ENDIF
                       +        ENDIF
                       +        IFIDN   <DEBUG>,<DEBUG>
= 0001                 + DEBUG$  =       1
                       +        ENDIF
                       +        ENDIF
                      C+        INCLUDE PCSTRUC.CRV
                      C+ ;*****************************************************************************
                      C+ ;                       Filename is PCSTRUC.CRV                               |
                      C+ ;*****************************************************************************
                      C+ ; This file contains all structure definitions for the implementation      |
                      C+ ; of the Corvus OMNINET PCShare Disk Server Emulator.  It is included      |
                      C+ ; in each of the assembly files when the START macro is invoked.  The      |
                      C+ ; selectors specified with that macro invocation cause the associated      |
                      C+ ; structure definitions to be included in the assembly.  Any set of       |
                      C+ ; definitions can be included with an appropriate set of selectors.       |
                      C+ ;*****************************************************************************
                      C+ ;                                                                           |
                      C+ ; Version:  2.0                                                             |
                      C+ ;                                                                           |
                      C+ ; Last Update:  14 January 1984                                             |
                      C+ ;                                                                           |
                      C+ ;*****************************************************************************
                      C+ ENDIF
                      C+ ;
```

```
                        C+ ENDIF
                        C+ ;
                        C+ ;--------------------------------------------------------------------------
                        C+ ;                    Emulator Resident Data Structures                  |
                        C+ ;--------------------------------------------------------------------------
                        C+ ;
                        C+ ;       Universal value definitions
                        C+ ;
= 0000                  C+ $True    EQU     0               ;Usually indicates OK or normal
= 0001                  C+ $False   EQU     1               ;Usually indicates NG or unusual
= 0000                  C+ $Zero    EQU     0               ;What is there to say
= 0001                  C+ $One     EQU     1               ;  about these?
= 0060                  C+ $SKnum   EQU     96              ;Number of internal stack entries
= 0001                  C+ $Vir_Drv1 EQU    1               ;Value for Virtual Drive 1
                        C+ ;
                        C+ ;--------------------------------------------------------------------------
                        C+ ;       8259A PIC usage values                                           |
                        C+ ;       Reference: Intel 8086 User's Manual                              |
                        C+ ;--------------------------------------------------------------------------
                        C+ ;
= 0021                  C+ $P82op1 EQU      21H             ;Port for OCW1 commands
= 0020                  C+ $P82op0 EQU      20H             ;Port for OCW2, OCW3 commands
                        C+ ;
= 000B                  C+ $P82Ris EQU      0BH             ;Read Interrupt Service register
= 0020                  C+ $P82Eoi EQU      20H             ;Simple end of interrupt
= 0001                  C+ $P82TML EQU      1               ;Timer interrupt level bit
= 0005                  C+ $P82EML EQU      4+$P82TML       ;Transporter and timer levels
= 00FA                  C+ $P82enb EQU      0FFH-$P82EML    ;Enable xporter and timer
                        C+ ;
                        C+ ;--------------------------------------------------------------------------
                        C+ ;       Structure Name:  Emulator Communication Table                    |
                        C+ ;       Reference: Program Logic description, Section 1.4                |
                        C+ ;--------------------------------------------------------------------------
                        C+ ;
                        C+ ;       Associated values and flags
                        C+ ;
= 0080                  C+ $Cons    EQU     80H             ;Write messages to console
= 0040                  C+ $ConsPT  EQU     40H             ;Prompt for console at initialization
= 0000                  C+ $AtrNrm  EQU     0               ;Standard attributes of Emulator files
= 000A                  C+ $TickMX  EQU     10              ;Default timer tick refresh count
                        C+ ;
= 0080                  C+ $TBintI  EQU     80H             ;Transporter interrupts installed
= 0040                  C+ $TBrenb  EQU     40H             ;Transporter to run enabled
= 0020                  C+ $TBrtck  EQU     20H             ;Transporter to run off timer tick
= 00DF                  C+ $TBnotk  EQU     0FFH-$TBrtck    ;Mask to clear timer tick on
= 0010                  C+ $TBinit  EQU     10H             ;Prompt for interrupts at init
= 0004                  C+ $TRChrd  EQU     4               ;Trace output tp hard disk
= 0002                  C+ $TRCcon  EQU     2               ;Trace output to console
= 0001                  C+ $TRCdsk  EQU     1               ;Trace output to disk
                        C+ ;
                        C+ ;       Interrupt vector numbers
                        C+ ;
= 000A                  C+ $CallEM  EQU     0AH             ;Emulator interrupt
= 001B                  C+ $CallCB  EQU     1BH             ;Control_break has occurred
= 001C                  C+ $CallTK  EQU     1CH             ;Timer interrupt appendage
```

```
= 0021                          C+ $CallDS EQU     21H             ;DOS function call
= 0023                          C+ $CallCA EQU     23H             ;Control_break appendage
= 0034                          C+ $CallKR EQU     34H             ;Gets location In-DOS flag
= 0066                          C+ $CallDI EQU     66H             ;DOS intercept
= 0067                          C+ $CallTM EQU     67H             ;Timer appendage recall
= 0001                          C+ $CallTP EQU     1               ;Trap interrupt
                                C+ ;
= 00CD                          C+ $INT$   EQU     0CDH            ;Interrupt instruction code
= 0001                          C+ $TrapFL EQU     1               ;Position of trap flag in AH
                                C+ ;
                                C+ ;       Field lengths
                                C+ ;
= 0008                          C+ $$Dir   EQU     8
= 0008                          C+ $$Nam   EQU     8
= 0004                          C+ $$Next  EQU     4
= 000A                          C+ $$Serv  EQU     10
                                C+ ;
                                C+ ;
                                C+ ECT     STRUC
0000  50 43 53 48 41 52         C+ Iden    DB      'PCSHARE2'      ;Identifier of the version
      45 32                     C+
0008  43                        C+ EC_Dat  DB      'C'             ;Drive for Emulator files (character)
0009  03                        C+ EC_Drv  DB      3               ;Same drive numeric    !NEW FIELD!
000A  40                        C+ EC_Flg  DB      $ConsPT         ;Flag for log output to console
000B  01                        C+ Trce    DB      $False          ;Trace active switch   !NEW FIELD!
000C  01                        C+ Cold    DB      $False          ;Cold start completed  !NEW FIELD!
000D  01                        C+ EC_Trm  DB      $False          ;Emulator terminated   !NEW FIELD!
000E  0064                      C+ Max     DW      100             ;Maximum number records in log file
0010  0000                      C+ Next    DW      0               ;RRN for next log record
0012  50 43 53 48 41 52         C+ Dir     DB      'PCSHARE '      ;Name of directory for Emulator files
      45 20                     C+
001A  50 43 53 48 41 52         C+ Nam     DB      'PCSHARE '      ;Filename for Emulator files
      45 20                     C+
0022     0A [                   C+ Serv    DB      $$Serv DUP(' ') ;Emulator name as a disk server
            20                  C+
               ]                C+
                                C+
002C  00                        C+ EC_AtrU DB      $AtrNrm         ;Attributes of User area file      !N F!
002D  00                        C+ EC_AtrS DB      $AtrNrm         ;Attributes of System area files   !E I!
002E  00                        C+ EC_AtrL DB      $AtrNrm         ;Attributes of Log file            !W E!
002F  C0                        C+ EC_TBI  DB      $TBintI+$TBrenb ;Transporter int flags             !  L!
0030  0A                        C+ TLtmck  DB      $TickMX         ;Current timer tick count          !  D!
0031  0A                        C+ TLtmrs  DB      $TickMX         ;Value to reset timer tick count   !  S!
0032  01                        C+ KEEPout DB      $False          ;DOS has been entered flag         !   !
0033  01                        C+ INTdfr  DB      $False          ;Deferred interrupt flag           !   !
0034  0000                      C+ DOSdkO  DW      0               ;Holds location of DOS             !   !
0036  0000                      C+ DOSdkS  DW      0               ;  Critical section (In-DOS) Flag  !   !
0038  0000                      C+ DDRH    DW      0               ;Holds location of request header when
003A  0000                      C+ DDRHS   DW      0               ;   Device Driver called    !NEW FIELD!
003C  0000                      C+ TckO    DW      0               ;Holds location of timer tick int when
003E  0000                      C+ TckS    DW      0               ;   no transporter ints     !NEW FIELD!
0040                            C+ ECT     ENDS
                                C+ ;
                                C+ ;-------------------------------------------------------------------------
                                C+ ;       Structure Name:  Log Request Block                              !
```

```
                        C+ ;       Reference: Program Logic Description, Section 4.27
                        C+ ;--------------------------------------------------------------------------
                        C+ ;
                        C+ ;       Associated values
                        C+ ;
= 0066                  C+ $MAXMSG EQU     102             ;Maximum message length
                        C+ ;
                        C+ ;       Log message numbers
                        C+ ;
= 0001                  C+ $MND00  EQU     1               ;Initialization message
= 0002                  C+ $MNT00  EQU     2
= 0003                  C+ $MNT01  EQU     3
= 0004                  C+ $MNT02  EQU     4
= 0005                  C+ $MNT03  EQU     5
= 0006                  C+ $MNT04  EQU     6
= 0007                  C+ $MNT05  EQU     7
= 0008                  C+ $MNT06  EQU     8
= 0009                  C+ $MND01  EQU     9
= 000A                  C+ $MND02  EQU     10
= 000B                  C+ $MND03  EQU     11
= 000C                  C+ $MND04  EQU     12
= 000D                  C+ $MNT07  EQU     13
= 000E                  C+ $MNT08  EQU     14
= 000F                  C+ $MNT09  EQU     15
= 0010                  C+ $MNT010 EQU     16
= 0011                  C+ $MND05  EQU     17
= 0012                  C+ $MNT011 EQU     18
= 0013                  C+ $MND06  EQU     19              ;ORB missing
= 0014                  C+ $MNT012 EQU     20              ;Bad pipes tables
                        C+ ;
                        C+ LRB     STRUC
0000  00                C+ Cde     DB      0               ;Module ID code
0001  00                C+ LR_Len  DB      0               ;Length of the message string
0002                    C+ LRB     ENDS
                        C+ ;
                        C+ IF (TL$ OR CI$ OR DM$ OR LC$)
                        C+ ;--------------------------------------------------------------------------
                        C+ ;       Structure Name:  Operation Request Block
                        C+ ;       Reference: Program Logic Description, Section 2.2.1
                        C+ ;--------------------------------------------------------------------------
                        C+ ;
                        C+ ;       Associated completion codes
                        C+ ;
= 00FF                  C+ $OPinp  EQU     0FFH            ;Operation in progress
= 0000                  C+ $OPcne  EQU     0               ;Completed without error
= 0001                  C+ $OPinl  EQU     1               ;Completed, incorrect length
= 0002                  C+ $OPtrb  EQU     2               ;Terminated, invalid ORB
= 0003                  C+ $OPtto  EQU     3               ;Terminated, receive time-out
= 0004                  C+ $OPtdf  EQU     4               ;Terminated, invalid data format
= 0005                  C+ $OPtng  EQU     5               ;Terminated, transmission failure
= 0006                  C+ $OPttf  EQU     6               ;Terminated, transporter failure
= 0007                  C+ $OPtcl  EQU     7               ;Terminated, control length mismatch
                        C+ ;
                        C+ ;       The structure itself
                        C+ ;
```

```
0000  FF                C+ ORB      STRUC
0001  00                C+ CC       DB      $OPinp          ;Completion Code
0002  0000              C+ OP       DB      0               ;Operational parameter
0004  0000              C+ Sze      DW      0               ;Number of bytes of data
0006  0000              C+ RB_Data  DW      0               ;Offset of data
0008                    C+ RB_Datas DW      0               ;Reserved for segment address
                        C+ ORB      ENDS
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ IF (DR$ OR TL$ OR LC$)
                        C+ ;-------------------------------------------------------------------------
                        C+ ;                    Transporter Usage Parameters                       |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
= 0100                  C+ $InitTA EQU      100H            ;Initial transporter CCB address
= 7FFF                  C+ $MaxDly EQU      7FFFH           ;Max loop count for operation complete
= 0010                  C+ $MaxRty EQU      10H             ;Maximum number operation retries
= 000A                  C+ $MaxWte EQU      10              ;Maximum wait time in seconds
= 01FE                  C+ $ProID  EQU      01FEH           ;protocol identifier
= 00CF                  C+ $SndNAD EQU      0CFH            ;Invalid command for NAD reply
= 0211                  C+ $MaxTBD EQU      529             ;Longest data transfer (w/o disk stat)
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;                  Transporter Socket Usage Conventions                 |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
= 00B0                  C+ $SokTD  EQU      0B0H            ;To transmit data
= 0080                  C+ $SokTM  EQU      80H             ;To transmit an NLM
= 0080                  C+ $SokRM  EQU      80H             ;To receive an NLM
= 00B0                  C+ $SokRC  EQU      0B0H            ;To receive initial command sequence
= 00A0                  C+ $SokRL  EQU      0A0H            ;To receive remainder of long command
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Name Lookup Message                              |
                        C+ ;       Reference: Program Logic Description, Section 2.2.3               |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ ;       Associated message type codes (most significant byte)
                        C+ ;
= 0000                  C+ $NLHi   EQU      0               ;Hello
= 0001                  C+ $NLAny  EQU      1               ;Any disk server (old protocol)
= 0002                  C+ $NLWho  EQU      2               ;Who are you?
= 0003                  C+ $NLWhr  EQU      3               ;Where are you?
= 0010                  C+ $NLIdn  EQU      10H             ;Identification
= 00FF                  C+ $NLBye  EQU      0FFH            ;Goodbye
                        C+ ;
                        C+ ;       Associated device types of interest to Emulator
                        C+ ;
= 0001                  C+ $NLDsk  EQU      1               ;Disk Server
= 00FF                  C+ $NLDny  EQU      0FFH            ;All Devices
                        C+ ;
                        C+ ;       Associated Name Length
                        C+ ;
= 000A                  C+ $NameLN EQU      10              ;Needed because STRUC is weak
```

```
                        C+ ;
                        C+ ;       The structure itself (arithmetic reverses byte order
                        C+ ;       for transporter memory storage)
                        C+ ;
                        C+ NLM     STRUC
0000  FE01              C+ Pid     DW      $ProID/256 + 256*($ProID AND 0FFH)
0002  0000              C+ Mgt     DW      $NLhi           ;Message Type
0004  0000              C+ Src     DW      0               ;Station Number of sender
0006  0100              C+ Dev     DW      256*$NLDsk      ;Device Type
0008  50 43 53 48 41 52 C+ NL_Name DB      'PCSHARE   '    ;Device or user name
      45 20 20 20       C+
0012                    C+ NLM     ENDS
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Receive ICS control area                      |
                        C+ ;       Reference: Corvus OMNINET Programmer' Guide                    |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ RRRCV   STRUC
0000  0000              C+ Lofcm   DW      0               ;length of the command being sent
0002  0000              C+ ELofRp  DW      0               ;Expected reply length (w/o status)
0004                    C+ RRRCV   ENDS
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Send Reply control area                       |
                        C+ ;       Reference: Corvus OMNINET Programmer' Guide                    |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ RRSND   STRUC
0000  0000              C+ ALofRp  DW      0               ;Actual length of reply (incl status)
0002  00                C+ Dstat   DB      0               ;Disk status
0003                    C+ RRSND   ENDS
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Transporter Status Table Header               |
                        C+ ;       Reference: Program Logic description, Section 2.3              |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ TSTH    STRUC
0000  00                C+ Int     DB      0               ;Interrupt depth indicator
0001  01                C+ TT_CI   DB      $False          ;Command Interpreter invocation flag
0002  00                C+ NE      DB      0               ;Number of entries in TST
0003  00                C+ TT_NAD  DB      0               ;Network address of Emulator
0004                    C+ TSTH    ENDS
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Transporter Status Table Entry                |
                        C+ ;       Reference: Program Logic Description, Section 2.3              |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ TSTE    STRUC
0000  01                C+ PN      DB      $False          ;ORB pending flag
0001  00                C+ TT_ST   DB      0               ;Status code from result record
0002  00                C+ SK      DB      0               ;Socket to be used
0003  00                C+ TT_OP   DB      0               ;Transporter commmand  !NEW USAGE!
```

```
0004  0000                C+ CCB      DW      0               ;Transporter address of the CCB
0006  0000                C+ RR       DW      0               ;Transporter address of result record
0008  0000                C+ Dtl      DW      0               ;Transporter address of data area
000A  0000                C+ Dsz      DW      0               ;Length of data area in bytes
000C  0000                C+ Isr      DW      0               ;Offset address of int service routine
000E  0000                C+ Isrs     DW      0               ;  Reserved for segment address
0010  0000                C+ TT_Orb   DW      0               ;Offset address of pending ORB
0012  0000                C+ TT_Orbs  DW      0               ;  Reserved for segment address
0014                      C+ TSTE     ENDS
                          C+ ;
                          C+ ENDIF
                          C+ ;
                          C+ IF (TL$ or LC$)
                          C+ ;-------------------------------------------------------------------------
                          C+ ;       Structure Name:  Transporter Status Table Local Command entry  |
                          C+ ;       Reference: Program Logic Description, Section 8.3              |
                          C+ ;-------------------------------------------------------------------------
                          C+ ;
                          C+ ;       Associated driver communication flag values
                          C+ ;
=                         C+ $LCnop   EQU     $False          ;No operation indication
=                         C+ $LCopn   EQU     $True           ;Operation in progress indication
= 00FF                    C+ $LCabt   EQU     0FFH            ;'Operation aborted' end indication
= 0000                    C+ $LCaok   EQU     0               ;'All ok' end indication
                          C+ ;
                          C+ ;       Asociated data values
                          C+ ;
=                         C+ $LCmaxL  EQU     $MaxTBD         ;Longest data transfer allowed
                          C+ ;
                          C+ TSTLE    STRUC
0000  00                  C+ LE_CC    DB      0               ;RB.CC or RQ.USE field
0001  00                  C+ LE_OP    DB      0               ;RB.OP field when serving as ORB
0002  0000                C+ LE_Len   DW      0               ;Length in bytes of remaining data
0004  0000                C+ Trm      DW      0               ;Offset from which to fetch data
0006  0000                C+ Trms     DW      0               ;  Segment address of data
0008  0000                C+ Rcv      DW      0               ;Offset to send reply
000A  0000                C+ Rcvs     DW      0               ;  Segment address for reply
000C  0000                C+ LE_Isr   DW      0               ;Offset of interrupt service routine
000E  0000                C+ LE_Isrs  DW      0               ;  Resderved for segment address ISR
0010  0000                C+ LE_BP    DW      0               ;Stack segment offset of inout registers
0012  FFFF                C+          DW      0FFFFH          ;Reserved
0014                      C+ TSTLE    ENDS
                          C+ ;
                          C+ ;-------------------------------------------------------------------------
                          C+ ;       Structure Name:  Command Request Queue Entry                    |
                          C+ ;       Reference: Program Logic Specification, Section 2.5             |
                          C+ ;-------------------------------------------------------------------------
                          C+ ;
                          C+ ;       Associated usage values
                          C+ ;
= 0080                    C+ $RQUseE  EQU     80H             ;Entry in queue
= 0040                    C+ $RQUseR  EQU     40H             ;Request received for command data
= 0020                    C+ $RQUseL  EQU     20H             ;Local command
= 0010                    C+ $RQUseI  EQU     10H             ;Internally generated command
                          C+ ;
```

```
= 0004                          C+ $RQCsze EQU     4               ;Number of command bytes enqueued
                                C+ ;
                                C+ ;       The structure itself
                                C+ ;
                                C+ CRQE    STRUC
0000  00                        C+ Use     DB      0               ;Usage flags
0001  00                        C+ NAD     DB      0               ;Network Address of requestor
0002  0000                      C+ Len     DW      0               ;Length in bytes of expected reply
0004     04 [                   C+ Cmnd    DB      $RQCsze Dup(0)  ;First four command bytes
               00               C+
                  ]             C+
                                C+
0008                            C+ CRQE    ENDS
                                C+ ;
                                C+ ENDIF
                                C+ ;
                                C+ IF (DR$ OR TL$)
                                C+ ;-------------------------------------------------------------------------
                                C+ ;       Structure Name:  Command Request Queue Header                   |
                                C+ ;       Reference: Program Logic Description, Section 2.5               |
                                C+ ;-------------------------------------------------------------------------
                                C+ ;
                                C+ ;       Associated queue status indicator values
                                C+ ;
= 0000                          C+ $RQmt   EQU     0               ;Queue is empty
= 0001                          C+ $RQent  EQU     1               ;Queue has entries but is not full
= 00FF                          C+ $RQful  EQU     0FFH            ;Queue is full
                                C+ ;
= 0080                          C+ $RQdse  EQU     80H             ;Disk status error flag bit
                                C+ ;
                                C+ CRQH    STRUC
0000  00                        C+ Qst     DB      $RQmt           ;Queue status indicator
0001  00                        C+ Dsc     DB      0               ;Current disk status code
0002  0000                      C+ End     DW      0               ;Offset address following CRQ
0004  0000                      C+ Top     DW      0               ;Offset address of head entry
0006  0000                      C+ Bot     DW      0               ;Offset address of tail entry
0008                            C+ CRQH    ENDS
                                C+ ;
                                C+ ENDIF
                                C+ ;
                                C+ IF TL$
                                C+ ;-------------------------------------------------------------------------
                                C+ ;       Structure Name:  Name Lookup Message Table                      |
                                C+ ;       Reference: Program Logic Description, Section 2.7               |
                                C+ ;-------------------------------------------------------------------------
                                C+ ;
                                C+ ;       Associated command modifiers
                                C+ ;
= 0003                          C+ $MTAdd  EQU     3               ;AddActive command
= 0000                          C+ $MTDel  EQU     0               ;DeleteActive command
                                C+ ;
                                C+ NLMT    STRUC
0000  10                        C+ MT_Use  DB      $RQUseI         ;Specifies internal command
0001  00                        C+ MT_NAD  DB      0               ;NAD of sender  !NEW FIELD!
0002     02 [                   C+         DB      2 DUP(0FFH)     ;Unused part of parameter block
```

```
                FF              C+
                    ]           C+
                                C+
0004  34                       C+ Opc      DB      34H             ;Internal is always hex 34
0005  00                       C+ MT_Mod   DB      0               ;Will be set to command modifier
0006      0A [                 C+ MT_Name  DB      10 DUP(' ')     ;Device or user name
                20              C+
                    ]           C+
                                C+
0010  00                       C+ MT_Src   DB      0               ;Station number of sender
0011  00                       C+ MT_Dev   DB      0               ;Device type of sender
0012      04 [                 C+ Rsvd     DB      4 DUP(' ')      ;Four byte user area
                20              C+
                    ]           C+
                                C+
0016                           C+ NLMT     ENDS
                               C+ ;
                               C+ ENDIF
                               C+ ;
                               C+ IF (DR$ OR TL$ OR LC$)
                               C+ ;-------------------------------------------------------------------------
                               C+ ;            Transporter structures and associated values             |
                               C+ ;            Reference: Corvus OMNINET Programmer's Guide             |
                               C+ ;-------------------------------------------------------------------------
                               C+ ;
                               C+ ;        Transporter Command Control Block
                               C+ ;
                               C+ CCBLK    STRUC
0000  00                       C+ CC_Cmnd  DB      0               ;Command
0001  00                       C+          DB      0               ;fill for high address byte
0002  0000                     C+ RRA      DW      0               ;address Result Record
0004  00                       C+ Sock     DB      0               ;socket to use
0005  00                       C+          DB      0               ;fill for high address byte
0006  0000                     C+ CC_Data  DW      0               ;Data address
0008  0000                     C+ Datl     DW      0               ;Data length
000A  00                       C+ Conln    DB      0               ;Control length
000B  00                       C+ DHost    DB      0               ;Destination host (if write)
000C                           C+ CCBLK    ENDS
                               C+ ;
                               C+ ;        Transporter Result Record structure
                               C+ ;
                               C+ RRKD     STRUC
0000  FF                       C+ RCode    DB      0FFH            ;Return Code
0001  00                       C+ SHost    DB      0               ;Source Host (if read, 0 write)
0002  0000                     C+ Rcvlen   DW      0               ;Length data received (if read)
0004                           C+ RRKD     ENDS
                               C+ ;
                               C+ ;        Transporter port definitions
                               C+ ;
= 0249                         C+ $RdRAM   EQU     249H            ;To read without incrementing counter
= 024B                         C+ $RdRAMI  EQU     24BH            ;To read and increment counter
= 0248                         C+ $RdStat  EQU     248H            ;To read status
                               C+ ;
= 0248                         C+ $WrCtrH  EQU     248H            ;To set high byte of counter
= 024A                         C+ $WrCtrL  EQU     24AH            ;To set low byte of counter
```

```
= 024B                  C+ $WrRAM  EQU   24BH          ;To write data and increment counter
                        C+ ;
= 0249                  C+ $WrStbe EQU   249H          ;To strobe command address
                        C+ ;
= 024C                  C+ $DiINT  EQU   24CH          ;To disable (disarm) interrupts
= 024D                  C+ $ClINT  EQU   24DH          ;To clear interrupt latch
= 024E                  C+ $EnINT  EQU   24EH          ;To enable (arm) interrupts
= 024F                  C+ $StINT  EQU   24FH          ;To read interrupt status
                        C+ ;
                        C+ ;       Transporter Status Flags
                        C+ ;
= 0080                  C+ $TRedy  EQU   80H           ;Transporter ready
= 0020                  C+ $TRenb  EQU   20H           ;Transporter interrupts enabled
= 0010                  C+ $TRpnd  EQU   10H           ;Transporter interrupt pending
= 0008                  C+ $TRlat  EQU   8             ;Interrupt latch
                        C+ ;
                        C+ ;       Transporter command codes
                        C+ ;
= 0040                  C+ $Send   EQU   40H           ;Send message
= 00F0                  C+ $Setup  EQU   0F0H          ;Setup to receive message
= 0010                  C+ $EndRcv EQU   10H           ;End receive if no message
= 0020                  C+ $InitT  EQU   20H           ;Initialize transporter
= 0001                  C+ $WhoAmI EQU   01H           ;Finds transporters NAD
= 0002                  C+ $Echo   EQU   02H           ;Test presence of specified transporter
                        C+ ;$PP    EQU   08H           ;Peek/Poke not used
                        C+ ;
                        C+ ;       Transporter result codes
                        C+ ;
    = 0000              C+ $TrOK   EQU   00H           ;Command completed successfully
    = 007F              C+ $TrST   EQU   7FH           ;Successful transmit retry maximum
    = 0080              C+ $TrNG   EQU   80H           ;Transmit failure, retry count exceeded
    = 0081              C+ $TrDL   EQU   81H           ;Data too long for receive buffer
    = 0082              C+ $TrNS   EQU   82H           ;Receiver's socket not set up
    = 0083              C+ $TrCL   EQU   83H           ;Control length mismatch
    = 0084              C+ $TrIS   EQU   84H           ;Invalid socket
    = 0085              C+ $TrNR   EQU   85H           ;receive socket in use
    = 0086              C+ $TrIT   EQU   86H           ;invalid transporter address
    = 00C0              C+ $TrEA   EQU   0C0H          ;Echo command acknowledged
    = 00FE              C+ $TrSS   EQU   0FEH          ;Receive socket setup successful
                        C+ ;
                        C+ ;       Transporter usage values
                        C+ ;
    = 0040              C+ $NADok  EQU   64            ;First invalid NAD value
    = 00FF              C+ $AllNAD EQU   0FFH          ;NAD for broadcast
    = 00FF              C+ $TrSU   EQU   0FFH          ;Initialization value for result code
                        C+ ;
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ ENDIF
                        C+ ;
```

```
                        C+ ENDIF
                        C+ ;
                        C+ ENDIF
                        C+ ENDIF
                        C+ ;
                        C+ ENDIF
                        C+
                           ;
                           ;*****************************************************************************
                           ;                       Segment Usage Definition                             |
                           ;*****************************************************************************
                           ;
                           PCS  GROUP  PC_CODE,PC_DATA
                                   ASSUME  CS:PCS,DS:PC_DATA,ES:PC_DATA
                           ;
                           ;*****************************************************************************
                           ;                       External Symbol Definitions                          |
                           ;*****************************************************************************
                           ;
                                   EXTRN   Dta_Bgn:BYTE            ;Start of data segment
                                   EXTRN   EC:BYTE                 ;Emulator Communication Table
                                   EXTRN   TT:BYTE                 ;TST header
                                   EXTRN   TTbgn:BYTE              ;Start of regular TST entries
                                   EXTRN   TTend:BYTE              ;End of regular entries
                                   EXTRN   RcvICS:BYTE             ;TST entry for ICS receive
                                   EXTRN   RcvNLM:BYTE             ;TST entry for NLM receive
                                   EXTRN   TrmNAD:BYTE             ;TST entry for sending NAD
                                   EXTRN   TL_BYT_IN:NEAR          ;Fetch byte from transporter
                                   EXTRN   TL_WRD_IN:NEAR          ;Fetch word from transporter
                                   EXTRN   TL_DTA_IN:NEAR          ;Fetch string from transporter
                                   EXTRN   TL_BYT_OUT:NEAR         ;Put byte into transporter
                                   EXTRN   CI_MAIN:NEAR            ;Command Interpreter
                                   EXTRN   TL_SET_RECV:NEAR        ;Setup message receive
                                   EXTRN   TL_SET_SEND:NEAR        ;Do message transmission
                                   EXTRN   TL_SET_RTRY:NEAR        ;Retry a command
                                   EXTRN   TL_SEND_NLM:NEAR        ;Send Name Lookup message
                                   EXTRN   LC_RCV_CMND:NEAR        ;Local Command Int Driver
                                   EXTRN   LC_INT_SRVR:NEAR        ;Local command NLM server
                                   EXTRN   DM_ERR_LOG:NEAR         ;Error log routine
                                   EXTRN   EM_TRACE:NEAR           ;Trace routine
                           ;
                           ;*****************************************************************************
                           ;                            Allocated Data                                  |
                           ;*****************************************************************************
                           ;
0000                       PC_DATA SEGMENT PUBLIC 'DATA'
                           ;
                           ;-----------------------------------------------------------------------------
                           ;       Command Request Queue
                           ;
= 003F                     $RQnum  EQU     63              ;Number of entries in the CRQ - 1
= 01F8                     $RQsze  EQU     $RQnum * TYPE CRQE
                           ;
                                   EYE     <CRQH>
0000  43 52 51 48          +            DB      "CRQH"
```

```
                                ;
0004  00                       RQ      CRQH    (,,RQend)       ;The queue header
0005  00
0006  020C R
0008  0000
000A  0000

000C  00                       RQbgn   CRQE    ()              ;First entry
000D  00
000E  0000
0010     04 [
               00
                  ]

0014  01F8 [                           DB      $RQsze DUP(0)   ;Remaining entries
               00
                  ]

= 020C                         RQend   EQU     THIS BYTE
                               ;
                               ;-----------------------------------------------------------------------
                               ;       Internal stack area
                               ;
020C  0000                     EMStak  DW      0               ;A generous stack size
= 020E                         EMstake EQU     THIS BYTE
                               ;
                               ;-----------------------------------------------------------------------
                               ;       Name Lookup Message Table
                               ;
                                       EYE     (NLMT)
020E  4E 4C 4D 54              +        DB      "NLMT"
                               ;
0212  10                       MT      NLMT    ()              ;Use STRUC initial values
0213  00
0214     02 [
               FF
                  ]
0216  34
0217  00
0218     0A [
               20
                  ]
0222  00
0223  00
0224     04 [
               20
                  ]

                               ;
                               ;-----------------------------------------------------------------------
                               ;       Identification message for WHO and WHERE replies
                               ;
                                       EYE     IDmg
0228  49 44 6D 67              +        DB      "IDmg"
                               ;
```

```
022C  FE01                    IDmsg   NLM     ()              ;Blank fields set by TL-INIT-TLD
022E  0000
0230  0000
0232  0100
0234  50 43 53 48 41 52
      45 20 20 20

                              ;
                              ;-----------------------------------------------------------------------
                              ;       Temporary storage for ICS data
                              ;
023E  00                      TmpICS  CRQE    ()              ;Form is a CRQ entry
023F  00
0240  0000
0242     04 [
               00
                  ]

                              ;
                              ;-----------------------------------------------------------------------
                              ;
0246                          PC_DATA ENDS
                              ;
                              ;***********************************************************************
                              ;                Transporter Logical Device Routines                  |
                              ;***********************************************************************
                              ;
0000                          PC_CODE SEGMENT PUBLIC 'CODE'
                              ;
                              ;***********************************************************************
                              ;Module name:  Interrupt Service Driver (TL-INT-DRVR)                  |
                              ;                                                                      |
                              ;Version:  2.0                                                         |
                              ;                                                                      |
                              ;Last Update: 15 January 1984                                          |
                              ;                                                                      |
                              ;Function:  The Interrupt Service Driver determines the cause of any   |
                              ;           transporter interrupt and calls the appropriate routine to |
                              ;           service it.  If the interrupt is due to software, the      |
                              ;           Local Command Interface interrupt routine LC-RCV-CMND is   |
                              ;           called.  If the interrupt is due to hardware, the hardware |
                              ;           interrupt depth counter (field TT.INT of the TST header)   |
                              ;           is incremented.  If the value is then 1, a new interrupt   |
                              ;           processing sequence is started.  If the value is greater   |
                              ;           than 1, reentrance has occured; immediate exit is then     |
                              ;           made as the interrupt will be serviced by the interrupt    |
                              ;           sequence already in progress.                              |
                              ;                                                                      |
                              ;Procedure:  Program Logic Description, Section 2.4                    |
                              ;                                                                      |
                              ;Called by:  8088 Interrupt A                                          |
                              ;                                                                      |
                              ;Routines called:  Interrupt service routines,                         |
                              ;                  TL-BYT-IN, CI-MAIN                                  |
                              ;                                                                      |
```

```
                        ;Input:  None
                        ;
                        ;Output:  Register data output to interrupt service routines is
                        ;
                        ;             (AL) = Completion code from result record
                        ;             (SI) = Offset of the TST entry
                        ;
                        ;         The service routines need not preserve any registers except
                        ;         the segment registers and the stack pointer
                        ;
                        ;Error procedures:  None
                        ;
                        ;Written by: R.B. Talmadge, Computer Technology Ltd
                        ;
                        ;Updates:  1: Timer tick interrupt feature, entry TL_INT_DTME, added
                        ;             30 November 1983
                        ;
                        ;          2: Separate primary test loop for external control of
                        ;             Emulator operation, entry TL_INT_TEST, implemented
                        ;             7 December 1983
                        ;
                        ;          3: Change timer tick option to normal non-assembly, done
                        ;             22 December 1983
                        ;
                        ;          4: DOS intercept driver added 28 December 1983
                        ;
                        ;          5: DOS intercept driver modified to issue control_break,
                        ;             done 11 January 1984
                        ;
                        ;          6: DOS intercept and Control_break mechanism replaced by
                        ;             DOS critical section test using In-DOS flag, done 14
                        ;             January 1984
                        ;*****************************************************************************
                        ;
0000                    TL_INT_DRVR  PROC  NEAR
                        ;
                                TENTER  EC              ;Set temporary environment
0000  50                +       PUSH    AX
0001  53                +       PUSH    BX
0002  51                +       PUSH    CX
0003  52                +       PUSH    DX
0004  BB 0000 E         +       MOV     BX,OFFSET PCS:EC
                        ;
0007  B0 0B                     MOV     AL,$P82Ris      ;Send 8259A command to
0009  E6 20                     OUT     $P82op0,AL      ;  read the interrupt service
000B  90                        NOP                     ;  register for source test
000C  E4 20                     IN      AL,$P82op0
000E  A8 05                     TEST    AL,$P82EML      ;Test for hardware
0010  75 03                     JNZ     HwDRVR          ;  interrupt
0012  E9 009F R                 JMP     SwDRVR          ;Go to software if not
                        ;
                        ;-----------------------------------------------------------------------
                        ;       For a hardware interrupt, the transporter and 8259A are reset
                        ;       and a test is made for reentrance.  If so, immediate exit is
                        ;       taken.  If not, a test is made for interrupt in DOS, when DOS
```

```
                        ;       entered from outside the Emulator.  If that has happened and   |
                        ;       DOS is in a critical section, a deferred interrupt is set up.  |
                        ;                                                                      |
                        ;       For interrupt outside DOS, the TL-INT-TEST routine is called.  |
                        ;       That routine will not return until the interrupt sequence has  |
                        ;       been completed, so TL_INT_DRVR just exits after the return.    |
                        ;-----------------------------------------------------------------------
                        ;
0015  2E: F6 47 2F 40   HwDRVR: TEST    CS:[BX].EC_TBI,$TBrenb  ;If running with
001A  74 04                     JZ      DRinc                   ;  transporter interupts
001C  BA 024D                   MOV     DX,$C1INT               ;  pick up latch port and
001F  EE                        OUT     DX,AL                   ;  Clear interrupt latch
                        ;
0020  BB 0000 E         DRinc:  MOV     BX,OFFSET PCS:TT        ;Position to TST header
0023  2E: FE 07                 INC     CS:[BX].Int             ;Increment interrupt depth
0026  2E: 80 3F 01              CMP     CS:[BX].Int,1           ;If we have not reentered
002A  74 09                     JE      DRtsds                  ;  go to process interrupt
                        ;
002C                    DRtxit  LABEL   NEAR
002C  B0 20                     MOV     AL,$P82eoi              ;If we have,
002E  E6 20                     OUT     $P82op0,AL              ;  reset 8259A
                                TEXIT                           ;  and exit
0030  5A                +       POP     DX
0031  59                +       POP     CX
0032  5B                +       POP     BX
0033  58                +       POP     AX
0034  CF                +       IRET
                        ;
                        ;       Test for interrupt within DOS; if not, call primary
                        ;       interrupt test routine and exit on return
                        ;
0035  2E: FE 0F         DRtsds: DEC     CS:[BX].Int             ;Adjust count for call or exit
0038  BB 0000 E                 MOV     BX,OFFSET PCS:EC        ;Recover ECT location
003B  1E                        PUSH    DS                      ;Set to this
003C  0E                        PUSH    CS                      ;  segment and
003D  1F                        POP     DS                      ;  fetch location
003E  C5 5F 34                  LDS     BX,DWORD PTR[BX].DOSdk0 ;  of In-DOS flag
0041  8B 07                     MOV     AX,[BX]                 ;Fetch flag itself
0043  1F                        POP     DS
                        ;
0044  0B C0                     OR      AX,AX                   ;If in DOS crtical section,
0046  75 4D                     JNZ     INTbrk                  ;  go set deferred interrupt
0048  B0 20                     MOV     AL,$P82eoi              ;If not critical, reset
004A  E6 20                     OUT     $P82op0,AL              ;  8259A and process
                        ;
                                ENTER   Dta_Bgn                 ;Set environment
004C  56                +       PUSH    SI
004D  57                +       PUSH    DI
004E  55                +       PUSH    BP
004F  1E                +       PUSH    DS
0050  06                +       PUSH    ES
0051  8C C8             +       MOV     AX,CS
0053  BB 0000 E         +       MOV     BX,OFFSET PCS:Dta_Bgn
0056  D1 EB             +       SHR     BX,1
0058  D1 EB             +       SHR     BX,1
```

```
005A  D1 EB                   +          SHR     BX,1
005C  D1 EB                   +          SHR     BX,1
005E  03 C3                   +          ADD     AX,BX
0060  8E D8                   +          MOV     DS,AX
0062  8E C0                   +          MOV     ES,AX
0064  8B EC                   +          MOV     BP,SP
                                ;
0066  F6 06 002F E 40                    TEST    EC.EC_TBI,$TBrenb       ;If running with
006B  74 05                              JZ      DRctst                  ;  transporter ints
006D  80 26 002F E DF                    AND     EC.EC_TBI,0FFH-$TBrtck  ;  clear timer
                                ;
0072  E8 00C2 R                 DRctst:  CALL    TL_INT_TEST             ;Go process interrupt
                                ;
                                         TRACE   E:TL-INT-DRVR-HDWE
0075  80 3E 000B E 00        +          CMP BYTE PTR EC.Trce,$True
007A  75 0F                   +          JNE     ??0000
007C  50                      +          PUSH    AX
007D  B0 00                   +          MOV     AL,0
007F  56                      +          PUSH    SI
0080  BE 0246 R               +          MOV     SI,OFFSET ??0001
0083  E8 0000 E               +          CALL    EM_TRACE
0086  5E                      +          POP     SI
0087  58                      +          POP     AX
0088  EB 01 90                +          JMP NEAR PTR ??0000
008B                          + PC_CODE ENDS
0246                          + PC_DATA SEGMENT PUBLIC 'DATA'
0246  00                      + ??0001  LRB     (,OFFSET ??0003 - OFFSET ??0002)
0247  12                      +
0248  45 3A 54 4C 2D 49       + ??0002  DB      "E:TL-INT-DRVR-HDWE"
025A                          + PC_DATA ENDS
008B                          + PC_CODE SEGMENT PUBLIC 'CODE'
008B                          + ??0000  LABEL   NEAR
                                ;
                                         EXIT    ,INTxit                 ;Exit on return
008B                          + INTxit  LABEL   NEAR
008B  07                      +          POP     ES
008C  1F                      +          POP     DS
008D  5D                      +          POP     BP
008E  5F                      +          POP     DI
008F  5E                      +          POP     SI
0090  5A                      +          POP     DX
0091  59                      +          POP     CX
0092  5B                      +          POP     BX
0093  58                      +          POP     AX
0094  CF                      +          IRET
                                ;
                                ;        If interrupt within DOS, exit if this has occurred previously
                                ;        before the deferred interrupt has been processed, otherwise
                                ;        force reentry by turning on timer tick entry control
                                ;
0095  BB 0000 E                 INTbrk:  MOV     BX,OFFSET PCS:EC        ;Recover ECT offset
0098  2E: 80 4F 2F 20                    OR      CS:[BX].EC_TBI,$TBrtck  ;Turn on timer
009D  EB 8D                              JMP     DRTxit                  ;  and go to exit
                                ;
                                ;--------------------------------------------------------------------------------
```

```
                        ;       For a software interrupt, call the Local Command Interface
                        ;       driver
                        ;-------------------------------------------------------------------
                        ;
009F  E8 0000 E         SwDRVR: CALL    LC_RCV_CMND     ;Call LC service routine
                        ;
                        ;       On return, test for operation in progress and call the
                        ;       primary interrupt test at CI_MAIN entry test if so
                        ;
                                TRACE   E:TL-INT-DRVR-STWE
00A2  80 3E 000B E 00   +               CMP BYTE PTR EC.Trce,$True
00A7  75 0F             +               JNE     ??0004
00A9  50                +               PUSH    AX
00AA  B0 00             +               MOV     AL,0
00AC  56                +               PUSH    SI
00AD  BE 025A R         +               MOV     SI,OFFSET ??0005
00B0  E8 0000 E         +               CALL    EM_TRACE
00B3  5E                +               POP     SI
00B4  58                +               POP     AX
00B5  EB 01 90          +               JMP NEAR PTR ??0004
00B8                    + PC_CODE ENDS
025A                    + PC_DATA SEGMENT PUBLIC 'DATA'
025A  00                + ??0005  LRB     (,OFFSET ??0007 - OFFSET ??0006)
025B  12                +
025C  45 3A 54 4C 2D 49 + ??0006  DB      "E:TL-INT-DRVR-STWE"
026E                    + PC_DATA ENDS
00B8                    + PC_CODE SEGMENT PUBLIC 'CODE'
00B8                    + ??0004  LABEL   NEAR
                        ;
00B8  FA                        CLI                     ;Disable interrupts
00B9  3C 00                     CMP     AL,$LCopn       ;If operation not started
00BB  75 CE                     JNE     INTxit          ;   go to exit, otherwise
00BD  E8 00F4 R                 CALL    DRVtci          ;   call primary routine
00C0  EB C9                     JMP     INTxit          ;Go to exit
                        ;
                        ;-------------------------------------------------------------------
                        ;       Primary interrupt test routine begins here.  This routine is
                        ;       called by TL_INT_DRVR for a transporter hardware interrupt,
                        ;       by TL_INT_DTME if timer ticks are being used to drive the
                        ;       Emulator, by the UTIL_INT_TEST device driver routine if the
                        ;       Emulator is being invoked by an external routine, and by the
                        ;       TL-INT-WAIT routine if an ORB wait occurs when the Emulator
                        ;       is being driven externally.
                        ;-------------------------------------------------------------------
                        ;
00C2                    TL_INT_TEST  LABEL  NEAR
                        ;
                        ;       Interrupt event loop tests result records associated with TST
                        ;       entries for change of status, and calls the interrupt service
                        ;       routine if a change has occurred.
                        ;
00C2  FE 06 0000 E      DRVgo:  INC     TT.Int          ;Update depth count
                        ;
00C6  FB                DRVlpe: STI                     ;Reenable interrupts
00C7  BE 0000 E                 MOV     SI,OFFSET TTbgn ;Fetch offset intial entry
```

```
                                ;
00CA  8B 5C 06                  DRVls:  MOV     BX,[SI].RR      ;Fetch result record location
00CD  E8 0000 E                         CALL    TL_BYT_IN       ;  and get current status
00D0  3A 44 01                          CMP     AL,[SI].TT_ST   ;If no change,
00D3  74 08                             JE      DRVlpn          ;  go to test for next
                                ;
00D5  56                                PUSH    SI              ;Save position
00D6  88 44 01                          MOV     [SI].TT_ST,AL   ;Record new status
00D9  FF 54 0C                          CALL WORD PTR [SI].Isr  ;Call interrupt service routine
00DC  5E                                POP     SI              ;Recover position
                                ;
00DD  83 C6 14                  DRVlpn: ADD     SI, TYPE TSTE   ;Increment to next entry
00E0  81 FE 0000 E                      CMP     SI, OFFSET TTend
00E4  7C E4                             JL      DRVls           ;Do next if not end
                                ;
                                ;       After a complete pass through the loop, it is restarted
                                ;       if the interrupt depth does not go to zero.  If it does
                                ;       go to zero the Command Interpreter is called unless the
                                ;       interrupt sequence started within it; in that case, the
                                ;       driver just exits to await return from CI.
                                ;
00E6  FA                                CLI                     ;Disable interrupts
00E7  FE 0E 0000 E                      DEC     TT.Int          ;Decrement depth count and
00EB  75 D9                             JNZ     DRVlpe          ;  restart loop if not zero
00ED  80 3E 0001 E 00                   CMP     TT.TT_CI,$True  ;Exit if we are
00F2  74 1F                             JE      DRVxit          ;  in the Command interpreter
                                ;
00F4  80 3E 000D E 00           DRVtci: CMP     EC.EC_TRM,$True ;Exit if Emulator
00F9  74 18                             JE      DRVxit          ;  operation terminated
00FB  80 3E 0004 R 00                   CMP     RQ.Qst,$RQmt    ;No need to call CI if
0100  74 11                             JE      DRVxit          ;  the CRQ is empty
0102  C6 06 0001 E 00           DRVcci: MOV     TT.TT_CI,$True  ;Turn on CI entry flag and
0107  FB                                STI                     ;  reenable interrupts
                                ;
0108  E8 0000 E                         CALL NEAR PTR CI_MAIN   ;Call the Command Interpreter
                                ;
                                ;       On return from CI-MAIN it is necessary to test the CRQ for
                                ;       a command which may have been inserted between the end of
                                ;       CI-MAIN processing and the return to this location
                                ;
010B  FA                                CLI                             ;Disable interrupts
010C  C6 06 0001 E 01                   MOV     TT.TT_CI,$False         ;Clear CI entry flag
0111  EB E1                             JMP     DRVtci                  ;Go test for exit
                                ;
0113                            DRVxit  LABEL   NEAR
                                ;
0113  C3                                RET                     ;Just a simple intra-seg return
                                ;
                                ;---------------------------------------------------------------
                                ;       When the transporter is to be run off the timer tick, this     |
                                ;       entry is invoked by each timer tick.  The ticks are counted    |
                                ;       down until the delay period expires.  The primary interupt     |
                                ;       loop is called when it does, and the delay period restarts     |
                                ;---------------------------------------------------------------
                                ;
```

```
0114                            TL_INT_DTME  LABEL  NEAR
                                ;
0114  50                               PUSH    AX                      ;Save working
0115  53                               PUSH    BX                      ;  registers
0116  BB 0000 E                        MOV     BX,OFFSET PCS:EC        ;Fetch ECT offset
                                ;
                                ;       Exit if timer tick not being used or if period not expired
                                ;
0119  2E: F6 47 2F 20                  TEST    CS:[BX].EC_TBI,$TBrtck  ;Test tick effective
011E  74 06                            JZ      TLTxit                  ;  and go out if not
0120  2E: FE 4F 30                     DEC     CS:[BX].TLtmck          ;Decrement timer tick
0124  74 05                            JZ      TLTgo                   ;Go to driver if count down
                                ;
                                ;       If not time to call the Emulator, restore registers, call
                                ;       the saved timer tick routine, then exit on return
                                ;
0126  5B                        TLTxit: POP     BX                      ;Reset to entry
0127  58                               POP     AX                      ;  conditions
0128  CD 67                            INT     $CallTM                 ;Call other timer rtn
012A  CF                               IRET                            ;  and then exit
                                ;
                                ;       If tick period has expired, reset the tick count, restore
                                ;       registers, and go to the main interrupt driver.  Note that
                                ;       in this case the saved timer tick exit routine is not called
                                ;
012B  2E: 8A 47 31              TLTgo:  MOV     AL,CS:[BX].TLtmrs       ;Reset timer tick
012F  2E: 88 47 30                      MOV     CS:[BX].TLtmck,AL       ;  count value
0133  5B                               POP     BX                      ;Reset to entry
0134  58                               POP     AX                      ;  condtions
0135  E9 0000 R                        JMP     TL_INT_DRVR             ;Go process the interrupt
                                ;
0138                            TL_INT_DRVR  ENDP
                                ;
                                ;*****************************************************************************
                                ;Module name:  Wait For Operation Complete (TL-INT-WAIT)                      |
                                ;                                                                             |
                                ;Version:  2.0                                                                |
                                ;                                                                             |
                                ;Last Update: 29 December 1983                                                |
                                ;                                                                             |
                                ;Function:  Wait For Operation Complete is called by any routine which       |
                                ;           cannot proceed until an operation started from an ORB has         |
                                ;           been completed.  If the Emulator is being driven by either        |
                                ;           its own or timer interrupts, the routine assure interrupts        |
                                ;           are enabled and waits for the interupt routine to set the         |
                                ;           ORB completion flag.   If interrupts are not being used,          |
                                ;           TL-INT-WAIT calls TL-INT-TEST to drive the interrupt test         |
                                ;           loop externally.                                                  |
                                ;                                                                             |
                                ;Procedure:  New routine; see comments in code                                |
                                ;                                                                             |
                                ;Called by:  TL-RTN-RSLT, TL-FETCH-LC                                         |
                                ;                                                                             |
                                ;Routines called:  TL-INT-TEST                                                |
                                ;
```

```
                        ;Input:  DS:(SI) = Location of ORB
                        ;
                        ;Output: ORB completion code field modified
                        ;
                        ;Error procedures:  None
                        ;
                        ;Written by: R.B. Talmadge, Computer Technology Ltd
                        ;
                        ;Updates:  None
                        ;
                        ;*************************************************************************
                        ;
0138                    TL_INT_WAIT  PROC  NEAR
                        ;
                        ;       If the transporter is being run by either interrupt type
                        ;       interrupt enable is assured and the wait is simply for a
                        ;       change of ORB completion status
                        ;
                                TRACE   S:TL-INT-WAIT
0138  80 3E 000B E 00   +       CMP BYTE PTR EC.Trce,$True
013D  75 0F             +       JNE     ??0008
013F  50                +       PUSH    AX
0140  B0 00             +       MOV     AL,0
0142  56                +       PUSH    SI
0143  BE 026E R         +       MOV     SI,OFFSET ??0009
0146  E8 0000 E         +       CALL    EM_TRACE
0149  5E                +       POP     SI
014A  58                +       POP     AX
014B  EB 01 90          +       JMP NEAR PTR ??0008
014E                    + PC_CODE ENDS
026E                    + PC_DATA SEGMENT PUBLIC 'DATA'
026E  00                + ??0009  LRB     (,OFFSET ??000B - OFFSET ??000A)
026F  0D                +
0270  53 3A 54 4C 2D 49 + ??000A  DB      "S:TL-INT-WAIT"
027D                    + PC_DATA ENDS
014E                    + PC_CODE SEGMENT PUBLIC 'CODE'
014E                    + ??0008  LABEL   NEAR
                        ;
014E  F6 06 002F E 60           TEST    EC.EC_TBI,$TBrenb+$TBrtck       ;If no interrupts
0153  74 1D                     JZ      WAITni                          ;  go to driver
                        ;
0155  FB                        STI                         ;Assure interrrupts enabled
0156  80 3C FF           WAITcc: CMP     [SI].CC,$OPinp      ;Wait around until the
0159  74 FB                     JE      WAITcc              ;  Operation code changes
                        ;
015B                    WAITxt  LABEL   NEAR
                        ;
                                TRACE   E:TL-INT-WAIT
015B  80 3E 000B E 00   +       CMP BYTE PTR EC.Trce,$True
0160  75 0F             +       JNE     ??000C
0162  50                +       PUSH    AX
0163  B0 00             +       MOV     AL,0
0165  56                +       PUSH    SI
0166  BE 027D R         +       MOV     SI,OFFSET ??000D
0169  E8 0000 E         +       CALL    EM_TRACE
```

```
016C  5E                    +          POP     SI
016D  58                    +          POP     AX
016E  EB 01 90              +          JMP NEAR PTR ??000C
0171                        + PC_CODE ENDS
027D                        + PC_DATA SEGMENT PUBLIC 'DATA'
027D  00                    + ??000D  LRB     (,OFFSET ??000F - OFFSET ??000E)
027E  0D                    +
027F  45 3A 54 4C 2D 49     + ??000E  DB      "E:TL-INT-WAIT"
028C                        + PC_DATA ENDS
0171                        + PC_CODE SEGMENT PUBLIC 'CODE'
0171                        + ??000C  LABEL   NEAR
                              ;
0171  C3                              RET                     ;Return to caller
                              ;
                              ;       If the transporter is being driven externally, call the
                              ;       primary interrupt test routine until the ORB completion
                              ;       status changes
                              ;
0172  80 3C FF                WAITni: CMP     [SI].CC,$OPinp  ;Exit if completion
0175  75 E4                           JNE     WAITxt          ;  has occurred
                                      SELSAVE (AX,BX,CX,DX,SI,DI)
0177  50                    +          PUSH  AX                ;SAVE AX
0178  53                    +          PUSH  BX                ;SAVE BX
0179  51                    +          PUSH  CX                ;SAVE CX
017A  52                    +          PUSH  DX                ;SAVE DX
017B  56                    +          PUSH  SI                ;SAVE SI
017C  57                    +          PUSH  DI                ;SAVE DI
017D  E8 00C2 R                       CALL    TL_INT_TEST     ;Call interrupt driver
                                      SELREST
0180  5F                    +          POP   DI                ;RESTORE DI
0181  5E                    +          POP   SI                ;RESTORE SI
0182  5A                    +          POP   DX                ;RESTORE DX
0183  59                    +          POP   CX                ;RESTORE CX
0184  5B                    +          POP   BX                ;RESTORE BX
0185  58                    +          POP   AX                ;RESTORE AX
0186  EB EA                           JMP     Waitni          ;  and test again
                              ;
0188                          TL_INT_WAIT  ENDP
                              ;
                              ;*****************************************************************************
                              ;Module name:  Dequeue CRQ (TL-DQ-CRQ)                                       |
                              ;                                                                            |
                              ;Version:  2.0                                                               |
                              ;                                                                            |
                              ;Last Update: 20 December 1983                                               |
                              ;                                                                            |
                              ;Function:  The Dequeue CRQ routine (TL-DQ-CRQ) is called by CI-MAIN         |
                              ;           to fetch the next command from the CRQ.  If the queue is         |
                              ;           not empty and there is a current command, the current            |
                              ;           command is deleted and the next command, if any, is made         |
                              ;           current.  If there is no current command, the first entry        |
                              ;           becomes the current command.  In either case, the content        |
                              ;           of the entry is returned to the caller.                          |
                              ;                                                                            |
                              ;Procedure:  Program Logic Description, Section 2.5                          |
```

```
                        ;
                        ;Called by:  CI-MAIN
                        ;
                        ;Routines called:  TL-SET-RECV, DM-ERR-LOG
                        ;
                        ;Input:  ES:(DI) = Location to return data
                        ;
                        ;Output:    (AL) = $True if item dequeued,
                        ;                  $Flase if queue was empty
                        ;
                        ;Error procedures:  An error is logged if the queue is not empty but
                        ;                   the queue indicator is not set for the top item
                        ;
                        ;Written by: R.B. Talmadge, Computer Technology Ltd
                        ;
                        ;Updates:  None
                        ;*****************************************************************************
                        ;
0188                    TL_DQ_CRQ  PROC  NEAR
                        ;
                                TRACE   S:TL-DQ-CRQ
0188  80 3E 000B E 00   +            CMP BYTE PTR EC.Trce,$True
018D  75 0F             +            JNE     ??0010
018F  50                +            PUSH    AX
0190  B0 00             +            MOV     AL,0
0192  56                +            PUSH    SI
0193  BE 028C R         +            MOV     SI,OFFSET ??0011
0196  E8 0000 E         +            CALL    EM_TRACE
0199  5E                +            POP     SI
019A  58                +            POP     AX
019B  EB 01 90          +            JMP NEAR PTR ??0010
019E                    + PC_CODE ENDS
028C                    + PC_DATA SEGMENT PUBLIC 'DATA'
028C  00                + ??0011  LRB     (,OFFSET ??0013 - OFFSET ??0012)
028D  0B                +
028E  53 3A 54 4C 2D 44 + ??0012  DB      "S:TL-DQ-CRQ"
0299                    + PC_DATA ENDS
019E                    + PC_CODE SEGMENT PUBLIC 'CODE'
019E                    + ??0010  LABEL   NEAR
                          ;
019E  80 3E 0004 R 00     DQs0:   CMP     RQ.Qst,$RQmt    ;Return immediately
01A3  75 19                       JNE     DQst            ;  if the queue is empty
                          ;
                                TRACE   X:TL-DQ-CRQ
01A5  80 3E 000B E 00   +            CMP BYTE PTR EC.Trce,$True
01AA  75 0F             +            JNE     ??0014
01AC  50                +            PUSH    AX
01AD  B0 00             +            MOV     AL,0
01AF  56                +            PUSH    SI
01B0  BE 0299 R         +            MOV     SI,OFFSET ??0015
01B3  E8 0000 E         +            CALL    EM_TRACE
01B6  5E                +            POP     SI
01B7  58                +            POP     AX
01B8  EB 01 90          +            JMP NEAR PTR ??0014
```

```
01BB                            + PC_CODE ENDS
0299                            + PC_DATA SEGMENT PUBLIC 'DATA'
0299  00                        + ??0015  LRB     (,OFFSET ??0017 - OFFSET ??0016)
029A  0B                        +
029B  58 3A 54 4C 2D 44         + ??0016  DB      "X:TL-DQ-CRQ"
02A6                            + PC_DATA ENDS
01BB                            + PC_CODE SEGMENT PUBLIC 'CODE'
01BB                            + ??0014  LABEL   NEAR
                                  ;
01BB  B0 01                       DQs00:  MOV     AL,$False       ;Set empty indicator
01BD  C3                                  RET                     ;Return to caller
                                  ;
01BE  56                          DQst:   PUSH    SI              ;Save working register
                                          STATUS                  ;Disable interrupts
01BF  9C                        +         PUSHF
01C0  FA                        +         CLI
                                  ;
                                  ;       If the queue is full, a call is made to the Setup Message
                                  ;       Receive routine to restart command reception
                                  ;
01C1  80 3E 0004 R FF                     CMP     RQ.Qst,$RQful   ;Test status
01C6  75 08                               JNE     DQst1
01C8  B0 01                               MOV     AL,$False       ;Call parameters are no ORB
01CA  BE 0000 E                           MOV     SI,OFFSET RcvICS   ;  and the Receive ICS TST entry
01CD  E8 0000 E                           CALL    TL_SET_RECV     ;Go restart reception
                                  ;
01D0  8B 36 0008 R                DQst1:  MOV     SI,RQ.Top       ;Offset of queue head item
01D4  F6 04 80                            TEST    [SI].Use,$RQUseE
01D7  75 10                               JNZ     DQok            ;OK, entry is queued
                                  ;
                                          RESTORE                 ;If entry is not queued
01D9  9D                        +         POPF
                                          LOG     $MND01          ;  log an error message,
01DA  50                        +         PUSH    AX
01DB  B0 09                     +         MOV     AL,$MND01
01DD  E8 0000 E                 +         CALL    DM_ERR_LOG
01E0  58                        +         POP     AX
01E1  C6 06 0004 R 00                     MOV     RQ.Qst,$RQmt    ;  set queue empty,
01E6  5E                                  POP     SI              ;  restore regsiter
01E7  EB D2                               JMP     DQs00           ;  and go to empty exit
                                  ;
                                  ;       Test entry for current item.  If not, go to fetch its
                                  ;       data.  If so, move to next item and set it current, then
                                  ;       go back and pretend we have just started the routine
                                  ;
01E9  F6 04 40                    DQok:   TEST    [SI].Use,$RQUseR
01EC  74 22                               JZ      DQfrst          ;First reference returns data
                                  ;
01EE  3B 36 000A R                        CMP     SI,RQ.Bot       ;If current entry is only
01F2  75 08                               JNE     DQok1           ;  one in queue, set empty
01F4  C6 06 0004 R 00                     MOV     RQ.Qst,$RQmt    ;  indicator and go
01F9  EB 0D 90                            JMP     DQok2           ;  back to process
                                  ;
01FC  83 C6 08                    DQok1:  ADD     SI,TYPE CRQE    ;Position to next entry
01FF  3B 36 0006 R                        CMP     SI,RQ.End       ;Reset position
```

```
0203  7C 03                          JL      DQok2           ;  if we wrap around
0205  BE 000C R                      MOV     SI,OFFSET RQbgn
                                    ;
0208  89 36 0008 R                  DQok2:  MOV     RQ.Top,SI       ;Reset top pointer
                                            RESTORE                 ;Restore interrupt status
020C  9D                          +         POPF
020D  5E                                    POP     SI              ;  and go back to process
020E  EB 8E                                 JMP     DQs0
                                    ;
                                    ;       Reenter here to send data from new current item
                                    ;
0210                                DQfrst: RESTORE                 ;Restore interrupt status
0210  9D                          +         POPF
0211  80 0C 40                             OR      [SI].Use,$RQUseR        ;Record passage
0214  57                                    PUSH    DI              ;Save original location
0215  51                                    PUSH    CX              ;Set up
0216  B9 0004                             MOV     CX,$RQCsze      ; fetch count
0219  8D 74 04                             LEA     SI,[SI].Cmnd    ;Set fetch position
021C  F3/ A4                               REP  MOVSB              ;Move the data
021E  59                                    POP     CX
021F  5F                                    POP     DI              ;Restore registers
                                    ;
0220  B0 00                                 MOV     AL,$True        ;Set return data
0222  5E                                    POP     SI              ;Restore work register
                                    ;
                                            TRACE   E:TL-DQ-CRQ
0223  80 3E 000B E 00            +         CMP BYTE PTR EC.Trce,$True
0228  75 0F                      +         JNE     ??0018
022A  50                          +         PUSH    AX
022B  B0 00                       +         MOV     AL,0
022D  56                          +         PUSH    SI
022E  BE 02A6 R                   +         MOV     SI,OFFSET ??0019
0231  E8 0000 E                   +         CALL    EM_TRACE
0234  5E                          +         POP     SI
0235  58                          +         POP     AX
0236  EB 01 90                    +         JMP NEAR PTR ??0018
0239                              + PC_CODE ENDS
02A6                              + PC_DATA SEGMENT PUBLIC 'DATA'
02A6  00                          + ??0019  LRB     (,OFFSET ??001B - OFFSET ??001A)
02A7  0B                          +
02A8  45 3A 54 4C 2D 44           + ??001A  DB      "E:TL-DQ-CRQ"
02B3                              + PC_DATA ENDS
0239                              + PC_CODE SEGMENT PUBLIC 'CODE'
0239                              + ??0018  LABEL   NEAR
                                    ;
0239  C3                                    RET                     ;Return to caller
                                    ;
023A                                TL_DQ_CRQ  ENDP
                                    ;
                                    ;*************************************************************************
                                    ;Module name:  Enqueue CRQ (TL-NQ-CRQ)                                  |
                                    ;                                                                       |
                                    ;Version:  2.0                                                          |
                                    ;                                                                       |
                                    ;Last Update: 20 December 1983                                          |
```

```
                        ;
                        ;Function:  The Enqueue CRQ routine (TL-NQ-CRQ) is called by service
                        ;           routines to place a new command into the queue.  The
                        ;           queue status (full or not full) is returned to the caller
                        ;
                        ;Procedure:  Program Logic Description, Section 2.5
                        ;
                        ;Called by:  TL-RCV-ICS, TL-RCV-HI, TL-RCV-BYE,
                        ;            TL-RCV-IDEN, LC-RECV-DISK, LC-SEND-DISK
                        ;
                        ;Routines called:  None
                        ;
                        ;Input:  DS:(SI) = Location of pro forma CRQ entry containing
                        ;                  data to be enqueued
                        ;
                        ;Output:    (AL) = Completion code
                        ;
                        ;Error procedures:  None
                        ;
                        ;Written by: R.B. Talmadge, Computer Technology Ltd
                        ;
                        ;Updates:  None
                        ;
                        ;*****************************************************************
                        ;
                        ;
023A                      TL_NQ_CRQ  PROC  NEAR
                          ;
                                   TRACE   S:TL-NQ-CRQ
023A  80 3E 000B E 00    +         CMP BYTE PTR EC.Trce,$True
023F  75 0F              +         JNE     ??001C
0241  50                 +         PUSH    AX
0242  B0 00              +         MOV     AL,0
0244  56                 +         PUSH    SI
0245  BE 02B3 R          +         MOV     SI,OFFSET ??001D
0248  E8 0000 E          +         CALL    EM_TRACE
024B  5E                 +         POP     SI
024C  58                 +         POP     AX
024D  EB 01 90           +         JMP NEAR PTR ??001C
0250                     + PC_CODE ENDS
02B3                     + PC_DATA SEGMENT PUBLIC 'DATA'
02B3  00                 + ??001D  LRB     <,OFFSET ??001F - OFFSET ??001E>
02B4  0B                 +
02B5  53 3A 54 4C 2D 4E  + ??001E  DB      "S:TL-NQ-CRQ"
02C0                     + PC_DATA ENDS
0250                     + PC_CODE SEGMENT PUBLIC 'CODE'
0250                     + ??001C  LABEL   NEAR
                          ;
0250  A0 0004 R                    MOV     AL,RQ.Qst       ;Return at once
0253  3C FF                       CMP     AL,$RQful       ;  if the queue is full
0255  75 17                       JNE     NQok
                          ;
                                   TRACE   X;TL-NQ-CRQ
0257  80 3E 000B E 00    +         CMP BYTE PTR EC.Trce,$True
025C  75 0F              +         JNE     ??0020
```

```
025E  50                        +           PUSH    AX
025F  B0 00                     +           MOV     AL,0
0261  56                        +           PUSH    SI
0262  BE 02C0 R                 +           MOV     SI,OFFSET ??0021
0265  E8 0000 E                 +           CALL    EM_TRACE
0268  5E                        +           POP     SI
0269  58                        +           POP     AX
026A  EB 01 90                  +           JMP NEAR PTR ??0020
026D                            + PC_CODE   ENDS
02C0                            + PC_DATA   SEGMENT PUBLIC 'DATA'
02C0  00                        + ??0021    LRB     (,OFFSET ??0023 - OFFSET ??0022)
02C1  01                        +
02C2  58                        + ??0022    DB      "X"
02C3                            + PC_DATA   ENDS
026D                            + PC_CODE   SEGMENT PUBLIC 'CODE'
026D                            + ??0020    LABEL   NEAR
                                  ;
026D  C3                                    RET
                                  ;
026E  57                          NQok:     PUSH    DI              ;Save working register
                                            STATUS                  ;Disable interrupts
026F  9C                        +           PUSHF
0270  FA                        +           CLI
0271  80 3E 0004 R 00                       CMP     RQ.Qst,$RQmt    ;Normal processing
0276  75 13                               JNE     NQst            ;  if the queue is not empty
                                  ;
                                  ;         If the queue is empty, set to use first entry and
                                  ;         go to fetch the data
                                  ;
0278  BF 000C R                           MOV     DI,OFFSET RQbgn
027B  89 3E 0008 R                        MOV     RQ.Top,DI       ;Set header pointers
027F  89 3E 000A R                        MOV     RQ.Bot,DI
0283  C6 06 0004 R 01                     MOV     RQ.Qst,$RQent   ;Set status indicator
0288  EB 20 90                            JMP     NQmov
                                  ;
                                  ;         Reenter here if queue neither empty or full
                                  ;
028B  8B 3E 000A R                NQst:   MOV     DI,RQ.Bot       ;Fetch offset tail entry
028F  83 C7 08                            ADD     DI,TYPE CRQE    ;Position to next
0292  3B 3E 0006 R                        CMP     DI,RQ.End       ;Reset position if
0296  7C 03                               JL      NQst1           ;  we wrap around
0298  BF 000C R                           MOV     DI,OFFSET RQbgn
                                  ;
029B  89 3E 000A R                NQst1:  MOV     RQ.Bot,DI       ;Reset bottom pointer
029F  3B 3E 0008 R                        CMP     DI,RQ.Top
02A3  75 05                               JNE     NQmov           ;Set indicator if
02A5  C6 06 0004 R FF                     MOV     RQ.Qst,$RQful   ;  queue now full
                                  ;
                                  ;         Reenter here to fetch data and set up new entry
                                  ;
02AA                              NQmov:  RESTORE                 ;Restore interrupt status
02AA  9D                        +           POPF
02AB  56                                    PUSH    SI              ;Save original locations
02AC  57                                    PUSH    DI
02AD  51                                    PUSH    CX              ;Set up
```

```
02AE  B9 0008                           MOV     CX,TYPE CRQE    ;  fetch count
02B1  F3/ A4                            REP MOVSB               ;Move the data
02B3  59                                POP     CX
02B4  5F                                POP     DI              ;Restore registers
02B5  5E                                POP     SI
02B6  80 0D 80                          OR      [DI].Use,$RQUseE  ;Indicate entry queued
02B9  A0 0004 R                         MOV     AL,RQ.Qst         ;Set return data
                                ;
02BC  5F                        NQxit:  POP     DI              ;Restore working register
                                ;
                                        TRACE   E:TL-NQ-CRQ
02BD  80 3E 000B E 00           +       CMP BYTE PTR EC.Trce,$True
02C2  75 0F                     +       JNE     ??0024
02C4  50                        +       PUSH    AX
02C5  B0 00                     +       MOV     AL,0
02C7  56                        +       PUSH    SI
02C8  BE 02C3 R                 +       MOV     SI,OFFSET ??0025
02CB  E8 0000 E                 +       CALL    EM_TRACE
02CE  5E                        +       POP     SI
02CF  58                        +       POP     AX
02D0  EB 01 90                  +       JMP NEAR PTR ??0024
02D3                            + PC_CODE ENDS
02C3                            + PC_DATA SEGMENT PUBLIC 'DATA'
02C3  00                        + ??0025  LRB     <,OFFSET ??0027 - OFFSET ??0026>
02C4  0B                        +
02C5  45 3A 54 4C 2D 4E         + ??0026  DB      "E:TL-NQ-CRQ"
02D0                            + PC_DATA ENDS
02D3                            + PC_CODE SEGMENT PUBLIC 'CODE'
02D3                            + ??0024  LABEL   NEAR
                                ;
02D3  C3                                RET                     ;Return to caller
                                ;
02D4                            TL_NQ_CRQ  ENDP
                                ;
                                ;*****************************************************************************
                                ;Module name:  Set Disk Status (TL-SET-ERR)                                   |
                                ;                                                                            |
                                ;Version:  2.0                                                               |
                                ;                                                                            |
                                ;Last Update: 14 November 1983                                               |
                                ;                                                                            |
                                ;Function:  The Set Disk Status routine (TL-SET-ERR) is called by            |
                                ;           any routine which wants to set the current disk status.          |
                                ;           Field RQ.DSC of the CRQ header is set to the value of the        |
                                ;           byte supplied as input.                                          |
                                ;                                                                            |
                                ;Procedure:  Program Logic Description, Section 2.5                          |
                                ;                                                                            |
                                ;Called by:  See function description                                        |
                                ;                                                                            |
                                ;Routines called:  None                                                      |
                                ;                                                                            |
                                ;Input:  (AL) = Byte to be inserted                                          |
                                ;                                                                            |
                                ;Output:  None                                                               |
```

```
                        ;                                                                       |
                        ;Error procedures:  None                                                |
                        ;                                                                       |
                        ;Written by: R.B. Talmadge, Computer Technology Ltd                     |
                        ;                                                                       |
                        ;Updates:  None                                                         |
                        ;                                                                       |
                        ;***************************************************************************
                        ;
02D4                    TL_SET_ERR  PROC  NEAR
                        ;
                                TRACE   S+E:TL-SET-ERR
02D4  80 3E 000B E 00   +       CMP BYTE PTR EC.Trce,$True
02D9  75 0F             +       JNE     ??0028
02DB  50                +       PUSH    AX
02DC  B0 00             +       MOV     AL,0
02DE  56                +       PUSH    SI
02DF  BE 02D0 R         +       MOV     SI,OFFSET ??0029
02E2  E8 0000 E         +       CALL    EM_TRACE
02E5  5E                +       POP     SI
02E6  58                +       POP     AX
02E7  EB 01 90          +       JMP NEAR PTR ??0028
02EA                    + PC_CODE ENDS
02D0                    + PC_DATA SEGMENT PUBLIC 'DATA'
02D0  00                + ??0029  LRB     (,OFFSET ??002B - OFFSET ??002A)
02D1  0E                +
02D2  53 2B 45 3A 54 4C + ??002A  DB      "S+E:TL-SET-ERR"
02E0                    + PC_DATA ENDS
02EA                    + PC_CODE SEGMENT PUBLIC 'CODE'
02EA                    + ??0028  LABEL   NEAR
                        ;
02EA  A2 0005 R                 MOV     RQ.Dsc,AL       ;Set new value
02ED  C3                        RET                     ;Return to caller
                        ;
02EE                    TL_SET_ERR  ENDP
                        ;
                        ;***************************************************************************
                        ;Module name:  Identify Requestor (TL-GET-NAD)                          |
                        ;                                                                       |
                        ;Version:  2.0                                                          |
                        ;                                                                       |
                        ;Last Update: 14 November 1983                                          |
                        ;                                                                       |
                        ;Function:  The Identify Requestor routine is called by any Emulator    |
                        ;           routine which wants to know the NAD of the requestor of     |
                        ;           the current command.  The value is extracted from field     |
                        ;           RQ.NAD of the CRQ entry for the current command             |
                        ;                                                                       |
                        ;Procedure:  Program Logic Description, Section 2.6.3                   |
                        ;                                                                       |
                        ;Called by:  See function description                                   |
                        ;                                                                       |
                        ;Routines called:  None                                                 |
                        ;                                                                       |
                        ;Input:  None                                                           |
```

```
                        ;
                        ;Output:  (AL) = NAD.  The value $FF (any or all machines) is
                        ;                      returned if there is no current entry in the CRQ
                        ;
                        ;Error procedures:  None
                        ;
                        ;Written by: R.B. Talmadge, Computer Technology Ltd
                        ;
                        ;Updates:  None
                        ;
                        ;*****************************************************************************
                        ;
02EE                    TL_GET_NAD  PROC  NEAR
                        ;
                                TRACE   S:TL-GET-NAD
02EE  80 3E 000B E 00   +               CMP BYTE PTR EC.Trce,$True
02F3  75 0F             +               JNE     ??002C
02F5  50                +               PUSH    AX
02F6  B0 00             +               MOV     AL,0
02F8  56                +               PUSH    SI
02F9  BE 02E0 R         +               MOV     SI,OFFSET ??002D
02FC  E8 0000 E         +               CALL    EM_TRACE
02FF  5E                +               POP     SI
0300  58                +               POP     AX
0301  EB 01 90          +               JMP NEAR PTR ??002C
0304                    + PC_CODE ENDS
02E0                    + PC_DATA SEGMENT PUBLIC 'DATA'
02E0  00                + ??002D  LRB     (,OFFSET ??002F - OFFSET ??002E)
02E1  0C                +
02E2  53 3A 54 4C 2D 47 + ??002E  DB      "S:TL-GET-NAD"
02EE                    + PC_DATA ENDS
0304                    + PC_CODE SEGMENT PUBLIC 'CODE'
0304                    + ??002C  LABEL   NEAR
                          ;
0304  B0 FF                     MOV     AL,$AllNAD      ;Assume CRQ is empty
0306  80 3E 0004 R 00           CMP     RQ.Qst,$RQmt    ;Go test top entry
030B  75 17                     JNE     NADgo           ;  if it is not
                          ;
                                TRACE   X:TL-GET-NAD
030D  80 3E 000B E 00   +               CMP BYTE PTR EC.Trce,$True
0312  75 0F             +               JNE     ??0030
0314  50                +               PUSH    AX
0315  B0 00             +               MOV     AL,0
0317  56                +               PUSH    SI
0318  BE 02EE R         +               MOV     SI,OFFSET ??0031
031B  E8 0000 E         +               CALL    EM_TRACE
031E  5E                +               POP     SI
031F  58                +               POP     AX
0320  EB 01 90          +               JMP NEAR PTR ??0030
0323                    + PC_CODE ENDS
02EE                    + PC_DATA SEGMENT PUBLIC 'DATA'
02EE  00                + ??0031  LRB     (,OFFSET ??0033 - OFFSET ??0032)
02EF  0C                +
02F0  58 3A 54 4C 2D 47 + ??0032  DB      "X:TL-GET-NAD"
02FC                    + PC_DATA ENDS
```

```
0323                            + PC_CODE SEGMENT PUBLIC 'CODE'
0323                            + ??0030  LABEL   NEAR
                                  ;
0323  C3                               RET
                                  ;
0324  53                        NADgo:  PUSH    BX                ;Save working register
0325  8B 1E 0008 R                     MOV     BX,RQ.Top         ;Fetch offset top entry
0329  F6 07 80                        TEST    [BX].Use,$RQUseE
032C  74 03                            JZ      NADxt             ;Exit if not queued
032E  8A 47 01                         MOV     AL,[BX].NAD       ;Fetch NAD if queued
                                  ;
0331  5B                        NADxt:  POP     BX                ;Restore working register
                                  ;
                                       TRACE   E:TL-GET-NAD
0332  80 3E 000B E 00          +       CMP BYTE PTR EC.Trce,$True
0337  75 0F                    +       JNE     ??0034
0339  50                       +       PUSH    AX
033A  B0 00                    +       MOV     AL,0
033C  56                       +       PUSH    SI
033D  BE 02FC R                +       MOV     SI,OFFSET ??0035
0340  E8 0000 E                +       CALL    EM_TRACE
0343  5E                       +       POP     SI
0344  58                       +       POP     AX
0345  EB 01 90                +       JMP NEAR PTR ??0034
0348                            + PC_CODE ENDS
02FC                            + PC_DATA SEGMENT PUBLIC 'DATA'
02FC  00                        + ??0035  LRB     (,OFFSET ??0037 - OFFSET ??0036)
02FD  0C                        +
02FE  45 3A 54 4C 2D 47         + ??0036  DB      "E:TL-GET-NAD"
030A                            + PC_DATA ENDS
0348                            + PC_CODE SEGMENT PUBLIC 'CODE'
0348                            + ??0034  LABEL   NEAR
                                  ;
0348  C3                               RET                       ;Return to caller
                                  ;
0349                              TL_GET_NAD  ENDP
                                  ;
                                  ;*****************************************************************************
                                  ;Module name:  Receive Initial Command Sequence (TL-RCV-ICS)                |
                                  ;                                                                           |
                                  ;Version:  2.0                                                              |
                                  ;                                                                           |
                                  ;Last Update: 14 January 1984                                               |
                                  ;                                                                           |
                                  ;Function:  The Receive Initial Command Sequence routine is the             |
                                  ;           interrupt service routine called whenever there is an           |
                                  ;           interrupt for socket B0 receive (TST entry RcvICS).  If         |
                                  ;           the interrupt is completion of a setup receive, no action       |
                                  ;           is required.  If a message has been received, TL-NQ-CRQ         |
                                  ;           is called to enter the command in the CRQ.  If the return       |
                                  ;           indicates the CRQ is not full, TL-SET-RECV is called to         |
                                  ;           restart command reception.                                      |
                                  ;                                                                           |
                                  ;Procedure:  Program Logic Description, Section 2.6                         |
                                  ;                                                                           |
```

```
                        ;Called by:  TL-INT-DRVR
                        ;
                        ;Routines called:  TL-NQ-CRQ, TL-SET-RECV, DM-ERR-LOG,
                        ;                  TL-BYT-IN, TL-WRD-IN, TL-DTA-IN
                        ;
                        ;Input:     (SI) = Offset TST entry
                        ;           (AL) = Result code from transporter
                        ;
                        ;Output:    None
                        ;
                        ;Error procedures:  An error message is logged if the completion code
                        ;                   from the transporter shows setup not sucessful
                        ;
                        ;Written by: R.B. Talmadge, Computer Technology Ltd
                        ;
                        ;Updates:  None
                        ;
                        ;*****************************************************************************
                        ;
0349                    TL_RCV_ICS  PROC  NEAR
                        ;
                                TRACE   S:TL-RCV-ICS
0349  80 3E 000B E 00     +          CMP BYTE PTR EC.Trce,$True
034E  75 0F               +          JNE     ??0038
0350  50                  +          PUSH    AX
0351  B0 00               +          MOV     AL,0
0353  56                  +          PUSH    SI
0354  BE 030A R           +          MOV     SI,OFFSET ??0039
0357  E8 0000 E           +          CALL    EM_TRACE
035A  5E                  +          POP     SI
035B  58                  +          POP     AX
035C  EB 01 90            +          JMP NEAR PTR ??0038
035F                      + PC_CODE ENDS
030A                      + PC_DATA SEGMENT PUBLIC 'DATA'
030A  00                  + ??0039  LRB     (,OFFSET ??003B - OFFSET ??003A)
030B  0C                  +
030C  53 3A 54 4C 2D 52   + ??003A  DB      "S:TL-RCV-ICS"
0318                      + PC_DATA ENDS
035F                      + PC_CODE SEGMENT PUBLIC 'CODE'
035F                      + ??0038  LABEL   NEAR
                            ;
035F  3C 00                         CMP     AL,$TrOK        ;Go to process if
0361  74 2B                         JE      RCVmsg          ;  a message has arrived
0363  3C FE                         CMP     AL,$TrSS        ;Just exit if this
0365  75 17                         JNE     RCVerr          ;  is a setup completion
                            ;
                                TRACE   X:TL-RCV-ICS
0367  80 3E 000B E 00     +          CMP BYTE PTR EC.Trce,$True
036C  75 0F               +          JNE     ??003C
036E  50                  +          PUSH    AX
036F  B0 00               +          MOV     AL,0
0371  56                  +          PUSH    SI
0372  BE 0318 R           +          MOV     SI,OFFSET ??003D
0375  E8 0000 E           +          CALL    EM_TRACE
0378  5E                  +          POP     SI
```

```
0379  58                      +          POP     AX
037A  EB 01 90                +          JMP NEAR PTR ??003C
037D                          + PC_CODE ENDS
0318                          + PC_DATA SEGMENT PUBLIC 'DATA'
0318  00                      + ??003D  LRB     (,OFFSET ??003F - OFFSET ??003E)
0319  0C                      +
031A  58 3A 54 4C 2D 52       + ??003E  DB      "X:TL-RCV-ICS"
0326                          + PC_DATA ENDS
037D                          + PC_CODE SEGMENT PUBLIC 'CODE'
037D                          + ??003C  LABEL   NEAR
                                ;
037D  C3                                RET
                                ;
                                ;       If the previous setup failed, decrement the retry count
                                ;       and retry the setup.  If the count goes to zero, log an
                                ;       error message and exit; command reception will then cease.
                                ;
037E  FE 4C 10                  RCVerr: DEC     BYTE PTR [SI].TT_Orb
0381  74 03                            JZ      RCVstp          ;Issue message if failure
0383  EB 47 90                          JMP     RCVxit          ;  else go to retry
                                ;
0386                            RCVstp: LOG     $MND02          ;Log the message
0386  50                      +         PUSH    AX
0387  B0 0A                  +         MOV     AL,$MND02
0389  E8 0000 E               +         CALL    DM_ERR_LOG
038C  58                      +         POP     AX
038D  C3                                RET                     ;Return to the Driver
                                ;
                                ;       Reenter here if a message has been received to
                                ;       fetch data from transporter memory and store in CRQ
                                ;
038E  8B 7C 06                  RCVmsg: MOV     DI,[SI].RR      ;Fetch location result record
0391  8D 5D 01                          LEA     BX,[DI].Shost   ;Position to Host data
0394  E8 0000 E                         CALL    TL_BYT_IN       ;Fetch the NAD
0397  A2 023F R                         MOV     TmpICS.NAD,AL     ; and store in temp area
                                ;
039A  83 C7 04                          ADD     DI,TYPE RRKD    ;Position to control area
039D  8D 5D 02                          LEA     BX,[DI].ELofRP  ;Fetch the length
03A0  E8 0000 E                         CALL    TL_WRD_IN       ;  of the expected reply
03A3  3D 0211                           CMP     AX,$MaxTBD      ;Assure value not
03A6  7E 03                             JLE     RCVmsh          ;  not larger than
03A8  B8 0211                           MOV     AX,$MaxTBD      ;  we can transmit
03AB  A3 0240 R                 RCVmsh: MOV     TmpICS.Len,AX   ;Store in temp area
                                ;
03AE  BF 023E R                         MOV     DI,OFFSET TmpICS  ;Fetch tmp stg offset
03B1  8D 7D 04                          LEA     DI,[DI].Cmnd      ;Point to command field
03B4  B9 0004                           MOV     CX,$RQCsze        ;Set count
03B7  8B 5C 08                          MOV     BX,[SI].Dtl       ;Transporter address data
03BA  E8 0000 E                         CALL    TL_DTA_IN         ;Fetch the command data
                                ;
03BD  8B FE                             MOV     DI,SI               ;Save offset to TST entry
03BF  BE 023E R                         MOV     SI,OFFSET TmpICS    ;Replace by tmp stg offset
03C2  E8 023A R                         CALL    TL_NQ_CRQ           ;Request enqueue of command
                                ;
                                ;       Restart reception of commands if CRQ is not full
```

```
                                ;
03C5  3C FF                             CMP     AL,$RQful       ;Just exit if
03C7  75 01                             JNE     RCVrst          ;  the queue is full
03C9  C3                                RET
                                ;
03CA  8B F7                     RCVrst: MOV     SI,DI           ;If not, restore TST offset,
03CC  B0 01                     RCVxit: MOV     AL,$False       ;  set for no ORB,
03CE  E8 0000 E                         CALL    TL_SET_RECV     ;Request restart of reception
                                ;
                                        TRACE   TL-RCV-ICS
03D1  80 3E 000B E 00           +       CMP BYTE PTR EC.Trce,$True
03D6  75 0F                     +       JNE     ??0040
03D8  50                        +       PUSH    AX
03D9  B0 00                     +       MOV     AL,0
03DB  56                        +       PUSH    SI
03DC  BE 0326 R                 +       MOV     SI,OFFSET ??0041
03DF  E8 0000 E                 +       CALL    EM_TRACE
03E2  5E                        +       POP     SI
03E3  58                        +       POP     AX
03E4  EB 01 90                  +       JMP NEAR PTR ??0040
03E7                            + PC_CODE ENDS
0326                            + PC_DATA SEGMENT PUBLIC 'DATA'
0326  00                        + ??0041  LRB     (,OFFSET ??0043 - OFFSET ??0042)
0327  0A                        +
0328  54 4C 2D 52 43 56         + ??0042  DB      "TL-RCV-ICS"
0332                            + PC_DATA ENDS
03E7                            + PC_CODE SEGMENT PUBLIC 'CODE'
03E7                            + ??0040  LABEL   NEAR
                                ;
03E7  C3                                RET                     ;Return to caller
                                ;
03E8                            TL_RCV_ICS  ENDP
                                ;
                                ;*****************************************************************************
                                ;Module name:  Send Network Address (TL-SEND-NAD)                            |
                                ;                                                                           |
                                ;Version:  2.0                                                              |
                                ;                                                                           |
                                ;Last Update: 14 November 1983                                              |
                                ;                                                                           |
                                ;Function:  The Send Network Address routine is called by the Name          |
                                ;           Lookup interrupt service driver when a request is received      |
                                ;           for the NAD of a disk server.  TL-SEND-NAD sends back an        |
                                ;           'invalid commmand' indication, as specified by the disk         |
                                ;           server protocol.  No action is taken to assure that the         |
                                ;           reply is received.                                              |
                                ;                                                                           |
                                ;Procedure:  Program Logic Description, Section 2.6.4                       |
                                ;                                                                           |
                                ;Called By:  TL-RCV-NLM                                                     |
                                ;                                                                           |
                                ;Routines called:  TL-SET-SEND, TL-WRD-OUT, TL-BYT-OUT                      |
                                ;                                                                           |
                                ;Input:  None                                                               |
                                ;                                                                           |
```

```
                        ;Output:  None                                                          |
                        ;                                                                       |
                        ;Error procedures:  None                                                |
                        ;                                                                       |
                        ;Written by: R.B. Talmadge, Computer Technology Ltd                     |
                        ;                                                                       |
                        ;Updates:  None                                                         |
                        ;                                                                       |
                        ;***************************************************************************
                        ;
03E8                    TL_SEND_NAD  PROC  NEAR
                        ;
                                TRACE   S:TL-SEND-NAD
03E8  80 3E 000B E 00   +               CMP BYTE PTR EC.Trce,$True
03ED  75 0F             +               JNE     ??0044
03EF  50                +               PUSH    AX
03F0  B0 00             +               MOV     AL,0
03F2  56                +               PUSH    SI
03F3  BE 0332 R         +               MOV     SI,OFFSET ??0045
03F6  E8 0000 E         +               CALL    EM_TRACE
03F9  5E                +               POP     SI
03FA  58                +               POP     AX
03FB  EB 01 90          +               JMP NEAR PTR ??0044
03FE                    + PC_CODE ENDS
0332                    + PC_DATA SEGMENT PUBLIC 'DATA'
0332  00                + ??0045  LRB     (,OFFSET ??0047 - OFFSET ??0046)
0333  0D                +
0334  53 3A 54 4C 2D 53 + ??0046  DB      "S:TL-SEND-NAD"
0341                    + PC_DATA ENDS
03FE                    + PC_CODE SEGMENT PUBLIC 'CODE'
03FE                    + ??0044  LABEL   NEAR
                        ;
                        ;       ---------------------------------------------------------
                        ;       | Note: The data to be transmitted, which is constant,  |
                        ;       |       was inserted into transporter memory by the     |
                        ;       |       initialization routine TL-INIT-TLD (Section 2.8)|
                        ;       ---------------------------------------------------------
                        ;
03FE  8A 0E 0213 R              MOV     CL,MT.NAD       ;Fetch requestor network address
0402  8B 1E 0004 E              MOV     BX,TrmNAD.CCB   ;Compute transporter location
0406  8D 5F 0B                  LEA     BX,[BX].Dhost   ;  to place the address
0409  E8 0000 E                 CALL    TL_BYT_OUT      ;Put into CCB
                        ;
040C  B8 0001                   MOV     AX,$False       ;Indicate there is no ORB
040F  BE 0000 E                 MOV     SI,OFFSET TrmNAD  ;Fetch TST entry offset
0412  E8 0000 E                 CALL    TL_SET_SEND       ;  and go do transmission
                        ;
                                TRACE   E:TL-SEND-NAD
0415  80 3E 000B E 00   +               CMP BYTE PTR EC.Trce,$True
041A  75 0F             +               JNE     ??0048
041C  50                +               PUSH    AX
041D  B0 00             +               MOV     AL,0
041F  56                +               PUSH    SI
0420  BE 0341 R         +               MOV     SI,OFFSET ??0049
0423  E8 0000 E         +               CALL    EM_TRACE
```

```
0426  5E                        +          POP     SI
0427  58                        +          POP     AX
0428  EB 01 90                  +          JMP NEAR PTR ??0048
042B                            + PC_CODE ENDS
0341                            + PC_DATA SEGMENT PUBLIC 'DATA'
0341  00                        + ??0049  LRB     (,OFFSET ??004B - OFFSET ??004A)
0342  0D                        +
0343  45 3A 54 4C 2D 53         + ??004A  DB      "E:TL-SEND-NAD"
0350                            + PC_DATA ENDS
042B                            + PC_CODE SEGMENT PUBLIC 'CODE'
042B                            + ??0048  LABEL   NEAR
                                ;
042B  C3                                  RET                       ;Go back to driver
                                ;
                                ;       The following routine is the TL-INT-NAD interrupt service routine,
                                ;       which is an integral part of the TL-SEND-NAD routine.  It is called
                                ;       by the interrupt service driver to handle the interrupts generated
                                ;       by the TL-SEND-NAD transmissions.
                                ;
042C                            TL_INT_NAD  LABEL  NEAR
                                ;
                                          TRACE   S+E:TL-INT-NAD
042C  80 3E 000B E 00           +         CMP BYTE PTR EC.Trce,$True
0431  75 0F                     +         JNE     ??004C
0433  50                        +         PUSH    AX
0434  B0 00                     +         MOV     AL,0
0436  56                        +         PUSH    SI
0437  BE 0350 R                 +         MOV     SI,OFFSET ??004D
043A  E8 0000 E                 +         CALL    EM_TRACE
043D  5E                        +         POP     SI
043E  58                        +         POP     AX
043F  EB 01 90                  +         JMP NEAR PTR ??004C
0442                            + PC_CODE ENDS
0350                            + PC_DATA SEGMENT PUBLIC 'DATA'
0350  00                        + ??004D  LRB     (,OFFSET ??004F - OFFSET ??004E)
0351  0E                        +
0352  53 2B 45 3A 54 4C         + ??004E  DB      "S+E:TL-INT-NAD"
0360                            + PC_DATA ENDS
0442                            + PC_CODE SEGMENT PUBLIC 'CODE'
0442                            + ??004C  LABEL   NEAR
                                ;
0442  C3                                  RET                       ;Do nothing (Well, that was easy)
                                ;
0443                            TL_SEND_NAD  ENDP
                                ;
                                ;*****************************************************************************
                                ;Module name:  Name Lookup Message Processing Routines                       |
                                ;                                                                           |
                                ;Version:  2.0                                                              |
                                ;                                                                           |
                                ;Last Update: 14 November 1983                                              |
                                ;                                                                           |
                                ;Function:  The Name Lookup Message Processing routines are called by       |
                                ;           the Name Lookup Receive driver to carry out the response        |
                                ;           required for the message.  There are five routines:             |
```

```
                        ;
                        ;           TL-RCV-HI responds to a Hello message by generating an
                        ;              AddActive command
                        ;           TL-RCV-BYE responds to a Goodby message by generating a
                        ;              DeleteActive command
                        ;           TL-RCV-WHO responds to a Who are you? message by sending
                        ;              an Identification message in response
                        ;           TL-RCV-WHERE responds to a Where are you? message by
                        ;              an Identification message in response if the specified
                        ;              name corresponds to the Emulator
                        ;           TL-RCV-IDEN responds to an Identification message by
                        ;              generating an AddActive command
                        ;
                        ;Procedure:  See Program Logic Description, Sections 2.7.1 - 2.7.5
                        ;
                        ;Called by:  TL-RCV-NLM
                        ;
                        ;Routines called:  TL-NQ-CRQ, TL-SEND-NLM
                        ;
                        ;Input:  (SI) = Offset of TST entry for socket 80 receive (RcvNLM)
                        ;
                        ;Output:  (MT.MOD) = Operation start indicator
                        ;
                        ;Error procedures:  None
                        ;
                        ;Written by: R.B. Talmadge, Computer Technology Ltd
                        ;
                        ;Updates:  None
                        ;
                        ;*****************************************************************************
                        ;
0443                    TL_NLM_SRVC  PROC  NEAR
                        ;
0443                    TL_RCV_HI  LABEL NEAR
                        ;
0443                    TL_RCV_IDEN  LABEL  NEAR
                        ;
                                TRACE   S+E:TL-RCV-HI/DEN
0443  80 3E 000B E 00   +            CMP BYTE PTR EC.Trce,$True
0448  75 0F             +            JNE     ??0050
044A  50                +            PUSH    AX
044B  B0 00             +            MOV     AL,0
044D  56                +            PUSH    SI
044E  BE 0360 R         +            MOV     SI,OFFSET ??0051
0451  E8 0000 E         +            CALL    EM_TRACE
0454  5E                +            POP     SI
0455  58                +            POP     AX
0456  EB 01 90          +            JMP NEAR PTR ??0050
0459                    + PC_CODE ENDS
0360                    + PC_DATA SEGMENT PUBLIC 'DATA'
0360  00                + ??0051  LRB     (,OFFSET ??0053 - OFFSET ??0052)
0361  11                +
0362  53 2B 45 3A 54 4C + ??0052  DB      "S+E:TL-RCV-HI/DEN"
0373                    + PC_DATA ENDS
0459                    + PC_CODE SEGMENT PUBLIC 'CODE'
```

```
0459                            + ??0050  LABEL   NEAR
                                  ;
0459  C6 06 0217 R 03             HInq:   MOV     MT.MT_Mod,$MTadd   ;Insert AddActive Command
045E  BE 0212 R                           MOV     SI,OFFSET MT     ;Point to ORB in NLMT
0461  E8 023A R                           CALL    TL_NQ_CRQ        ;Insert command into CRQ
0464  C3                                  RET                      ;Return to driver
                                  ;
0465                              TL_RCV_BYE  LABEL  NEAR
                                  ;
                                          TRACE   S+E:TL-RCV-BYE
0465  80 3E 000B E 00            +         CMP BYTE PTR EC.Trce,$True
046A  75 0F                      +         JNE     ??0054
046C  50                         +         PUSH    AX
046D  B0 00                      +         MOV     AL,0
046F  56                         +         PUSH    SI
0470  BE 0373 R                  +         MOV     SI,OFFSET ??0055
0473  E8 0000 E                  +         CALL    EM_TRACE
0476  5E                         +         POP     SI
0477  58                         +         POP     AX
0478  EB 01 90                   +         JMP NEAR PTR ??0054
047B                             + PC_CODE ENDS
0373                             + PC_DATA SEGMENT PUBLIC 'DATA'
0373  00                         + ??0055  LRB     (,OFFSET ??0057 - OFFSET ??0056)
0374  0E                         +
0375  53 2B 45 3A 54 4C          + ??0056  DB      "S+E:TL-RCV-BYE"
0383                             + PC_DATA ENDS
047B                             + PC_CODE SEGMENT PUBLIC 'CODE'
047B                             + ??0054  LABEL   NEAR
                                  ;
047B  C6 06 0217 R 00                     MOV     MT.MT_Mod,$MTDel   ;Insert DeleteActive command
0480  EB DC                               JMP     HInq             ;go put into CRQ
                                  ;
0482                              TL_RCV_WHO  LABEL  NEAR
                                  ;
                                          TRACE   S+E:TL-RCV-WHO
0482  80 3E 000B E 00            +         CMP BYTE PTR EC.Trce,$True
0487  75 0F                      +         JNE     ??0058
0489  50                         +         PUSH    AX
048A  B0 00                      +         MOV     AL,0
048C  56                         +         PUSH    SI
048D  BE 0383 R                  +         MOV     SI,OFFSET ??0059
0490  E8 0000 E                  +         CALL    EM_TRACE
0493  5E                         +         POP     SI
0494  58                         +         POP     AX
0495  EB 01 90                   +         JMP NEAR PTR ??0058
0498                             + PC_CODE ENDS
0383                             + PC_DATA SEGMENT PUBLIC 'DATA'
0383  00                         + ??0059  LRB     (,OFFSET ??005B - OFFSET ??005A)
0384  0E                         +
0385  53 2B 45 3A 54 4C          + ??005A  DB      "S+E:TL-RCV-WHO"
0393                             + PC_DATA ENDS
0498                             + PC_CODE SEGMENT PUBLIC 'CODE'
0498                             + ??0058  LABEL   NEAR
                                  ;
0498  80 3E 0223 R 01                     CMP     MT.MT_Dev,$NLDsk         ;If disk server wanted
```

```
049D  74 08                             JE      WHOsnd                  ;  send back ID
049F  80 3E 0223 R FF                   CMP     MT.MT_Dev,$NLDny        ;Do same if anybody
04A4  74 01                             JE      WHOsnd                  ;  will do
04A6  C3                                RET                             ;Otherwise exit
                                ;
04A7  8A 0E 0213 R              WHOsnd: MOV     CL,MT.MT_NAD            ;Fetch NAD
04AB  BE 022C R                         MOV     SI,OFFSET IDmsg         ;Point to Iden message
04AE  E8 0000 E                         CALL    TL_SEND_NLM             ;Send the reply
04B1  C3                                RET                             ;Go back to driver
                                ;
                                ;
04B2                            TL_RCV_WHERE  LABEL  NEAR
                                ;
                                        TRACE   S+E:TL-RCV-WHERE
04B2  80 3E 000B E 00           +       CMP BYTE PTR EC.Trce,$True
04B7  75 0F                     +       JNE     ??005C
04B9  50                        +       PUSH    AX
04BA  B0 00                     +       MOV     AL,0
04BC  56                        +       PUSH    SI
04BD  BE 0393 R                 +       MOV     SI,OFFSET ??005D
04C0  E8 0000 E                 +       CALL    EM_TRACE
04C3  5E                        +       POP     SI
04C4  58                        +       POP     AX
04C5  EB 01 90                  +       JMP NEAR PTR ??005C
04C8                            + PC_CODE ENDS
0393                            + PC_DATA SEGMENT PUBLIC 'DATA'
0393  00                        + ??005D  LRB     (,OFFSET ??005F - OFFSET ??005E)
0394  10                        +
0395  53 2B 45 3A 54 4C         + ??005E  DB      "S+E:TL-RCV-WHERE"
03A5                            + PC_DATA ENDS
04C8                            + PC_CODE SEGMENT PUBLIC 'CODE'
04C8                            + ??005C  LABEL   NEAR
                                ;
04C8  B9 000A                           MOV     CX,$NameLN      ;Set count to length of name
04CB  BF 0022 E                         MOV     DI,OFFSET EC.Serv       ;Offset to Emulator name
04CE  BE 0218 R                         MOV     SI,OFFSET MT.MT_Name    ;Offset to name sought
04D1  FC                                CLD
04D2  F3/ A6                            REPE    CMPSB           ;If the names are the
04D4  74 AC                             JE      TL_RCV_WHO      ;  same, go test for Iden msg
04D6  C3                                RET                     ;If not, return to driver
                                ;
04D7                            TL_NLM_SRVC  ENDP
                                ;
                                ;*****************************************************************************
                                ;Module name:  Receive Name Lookup Message (TL-RCV-NLM)                       |
                                ;                                                                            |
                                ;Version:  2.0                                                               |
                                ;                                                                            |
                                ;Last Update: 14 November 1983                                               |
                                ;                                                                            |
                                ;Function:  The Receive Name Lookup Message routine is the interrupt         |
                                ;           service routine called whenever there is an interrupt for        |
                                ;           socket 80 receive (TST entry RcvNLM).  If the interrupt is       |
                                ;           completion of a setup receive, no action action is taken.        |
                                ;           If a message has been received, TL-RCV-NLM determines the        |
```

```
                        ;           message type and invokes an appropriate messare processing
                        ;           routines.
                        ;
                        ;Procedure:  Program Logic Description, Section 2.7
                        ;
                        ;Called by:  TL-INT-DRVR
                        ;
                        ;Routines called:  TL-SET-RECV, DM-ERR-LOG, TL-BYT-IN,
                        ;                  TL-WRD-IN, TL-DTA-IN, LC-INT-SRVR,
                        ;                  Message Processing routines
                        ;
                        ;Input:     (SI) = Offset TST entry
                        ;           (AL) = Result code from transporter
                        ;
                        ;Output:    None
                        ;
                        ;Error procedures:  An error message is logged if the completion code
                        ;                   from the transporter shows setup not sucessful
                        ;
                        ;Written by: R.B. Talmadge, Computer Technology Ltd
                        ;
                        ;Updates:  None
                        ;
                        ;*****************************************************************
                        ;
04D7                    TL_RCV_NLM  PROC  NEAR
                        ;
                                    TRACE   S:TL-RCV-NLM
04D7  80 3E 000B E 00   +           CMP BYTE PTR EC.Trce,$True
04DC  75 0F             +           JNE     ??0060
04DE  50                +           PUSH    AX
04DF  B0 00             +           MOV     AL,0
04E1  56                +           PUSH    SI
04E2  BE 03A5 R         +           MOV     SI,OFFSET ??0061
04E5  E8 0000 E         +           CALL    EM_TRACE
04E8  5E                +           POP     SI
04E9  58                +           POP     AX
04EA  EB 01 90          +           JMP NEAR PTR ??0060
04ED                    + PC_CODE ENDS
03A5                    + PC_DATA SEGMENT PUBLIC 'DATA'
03A5  00                + ??0061  LRB     (,OFFSET ??0063 - OFFSET ??0062)
03A6  0C                +
03A7  53 3A 54 4C 2D 52 + ??0062  DB      "S:TL-RCV-NLM"
03B3                    + PC_DATA ENDS
04ED                    + PC_CODE SEGMENT PUBLIC 'CODE'
04ED                    + ??0060  LABEL   NEAR
                        ;
04ED  80 3C 00                    CMP     [SI].PN,$True   ;If message being solicited
04F0  75 03                       JNE     NLMmsg          ;  transfer to Local Command
04F2  E9 0000 E                   JMP     LC_INT_SRVR     ;  interrupt service rtn
                        ;
04F5  3C 00             NLMmsg:   CMP     AL,$TrOK        ;Go to process if
04F7  74 2C                       JE      NLMmg1          ;  a message has arrived
04F9  3C FE                       CMP     AL,$TrSS        ;Just exit if this
04FB  75 17                       JNE     NLMerr          ;  is a setup completion
```

```
                                ;
                                        TRACE   X:TL-RCV-NLM
04FD  80 3E 000B E 00         +         CMP BYTE PTR EC.Trce,$True
0502  75 0F                   +         JNE     ??0064
0504  50                      +         PUSH    AX
0505  B0 00                   +         MOV     AL,0
0507  56                      +         PUSH    SI
0508  BE 03B3 R               +         MOV     SI,OFFSET ??0065
050B  E8 0000 E               +         CALL    EM_TRACE
050E  5E                      +         POP     SI
050F  58                      +         POP     AX
0510  EB 01 90                +         JMP NEAR PTR ??0064
0513                          + PC_CODE ENDS
03B3                          + PC_DATA SEGMENT PUBLIC 'DATA'
03B3  00                      + ??0065  LRB     <,OFFSET ??0067 - OFFSET ??0066>
03B4  0C                      +
03B5  58 3A 54 4C 2D 52       + ??0066  DB      "X:TL-RCV-NLM"
03C1                          + PC_DATA ENDS
0513                          + PC_CODE SEGMENT PUBLIC 'CODE'
0513                          + ??0064  LABEL   NEAR
                                ;
0513  C3                               RET
                                ;
                                ;       If the previous setup failed, decrement the retry count
                                ;       and retry the setup.  If the count goes to zero, log an
                                ;       error message and exit; message reception will then cease.
                                ;
0514  FE 4C 10                  NLMerr: DEC     BYTE PTR [SI].TT_Orb
0517  74 04                           JZ      NLMstp          ;Issue message if failure
0519  E8 0000 E                       CALL    TL_SET_RTRY     ;If not, request retry
051C  C3                              RET                     ;  and return to caller
                                ;
051D                            NLMstp: LOG     $MND04              ;Log the message
051D  50                      +         PUSH    AX
051E  B0 0C                   +         MOV     AL,$MND04
0520  E8 0000 E               +         CALL    DM_ERR_LOG
0523  58                      +         POP     AX
0524  C3                               RET                         ;Return to the Driver
                                ;
                                ;       Reenter here if a message has been received.  The
                                ;       protocol ID is tested and the message discarded if
                                ;       the ID does not match that of a Name Lookup message.
                                ;
0525  8B 5C 08                 NLMmg1: MOV     BX,[SI].Dtl     ;Transporter address of data
0528  8D 1F                           LEA     BX,[BX].Pid     ;Position to protocol ID
052A  E8 0000 E                       CALL    TL_WRD_IN       ;Fetch the ID
052D  3D 01FE                         CMP     AX,$ProID       ;If not Name Lookup
0530  74 03                           JE      NLMmg2          ;  go restart reception
0532  E9 05C3 R                       JMP     NLMrst
                                ;
                                ;       If Name Lookup message, the message data is placed into
                                ;       the Name Lookup Table, and the appropriate processing
                                ;       routine is invoked.
                                ;
0535  8B 7C 06                 NLMmg2: MOV     DI,[SI].RR      ;Transporter address res rkd
```

```
0538  8D 5D 01                        LEA     BX,[DI].Shost   ;Position to sender NAD
053B  E8 0000 E                       CALL    TL_BYT_IN       ;Fetch NAD and
053E  A2 0213 R                       MOV     MT.MT_NAD,AL    ;  save in NLMT
                                ;
0541  8B 7C 08                        MOV     DI,[SI].Dtl     ;Transporter address of data
0544  8D 5D 04                        LEA     BX,[DI].Src     ;Position to source
0547  E8 0000 E                       CALL    TL_WRD_IN       ;Fetch source and
054A  A2 0222 R                       MOV     MT.MT_Src,AL    ;  place into NLMT
054D  8D 5D 06                        LEA     BX,[DI].Dev     ;Position to device type
0550  E8 0000 E                       CALL    TL_WRD_IN       ;Fetch type and
0553  A2 0223 R                       MOV     MT.MT_Dev,AL    ;  place into NLMT
                                ;
0556  8D 5D 08                        LEA     BX,[DI].NL_Name     ;Position to Name
0559  BF 0218 R                       MOV     DI,OFFSET MT.MT_Name
055C  B9 000A                         MOV     CX,$NameLN          ;Number characters
055F  E8 0000 E                       CALL    TL_DTA_IN       ;Fetch name to NLMT
                                ;
0562  8B 7C 08                        MOV     DI,[SI].Dtl     ;Compute transporter
0565  8D 5D 02                        LEA     BX,[DI].Mgt     ;  address message type
0568  E8 0000 E                       CALL    TL_WRD_IN       ;Bring it in
                                ;
                                ;       Because the message type codes are irregular and few in
                                ;       number, dispatching is carried out by a series of tests
                                ;       rather than by a transfer table.
                                ;
056B  C6 06 0217 R FF                  MOV     MT.MT_Mod,$TrSU     ;Initialize Operation code
                                ;
0570  BF 03E8 R                        MOV     DI,OFFSET TL_SEND_NAD
0573  80 FC 01                         CMP     AH,$NLany       ;Any disk server?
0576  74 2B                            JE      NLMcal          ;  If so, send NAD
                                ;
0578  BF 0443 R                        MOV     DI,OFFSET TL_RCV_HI
057B  80 FC 00                         CMP     AH,$NLHi        ;Hello message?
057E  74 23                            JE      NLMcal          ;  if so, add active
                                ;
0580  BF 0465 R                        MOV     DI,OFFSET TL_RCV_BYE
0583  80 FC FF                         CMP     AH,$NLBye       ;Goodbye message?
0586  74 1B                            JE      NLMcal          ;  if so, delete active
                                ;
0588  8B 3E 0443 R                     MOV     DI,TL_RCV_IDEN
058C  80 FC 10                         CMP     AH,$NLIdn       ;Identification?
058F  74 12                            JE      NLMcal          ;  if so, add active
                                ;
0591  8B 3E 0482 R                     MOV     DI,TL_RCV_WHO
0595  80 FC 02                         CMP     AH,$NLWho       ;Who are you?
0598  74 09                            JE      NLMcal          ;  if so, send ID
                                ;
059A  8B 3E 04B2 R                     MOV     DI,TL_RCV_WHERE
059E  80 FC 03                         CMP     AH,$NLWhr       ;Where are you?
05A1  75 02                            JNE     NLMrtn          ;Ignore message if not
                                ;
05A3  FF D7                     NLMcal: CALL    DI              ;Call processing routine
                                ;
                                ;       Return here from all message processing routines
                                ;
```

offset

offset

offset

```
05A5  80 3E 0217 R FF          NLMrtn: CMP     MT.MT_MOD,$TrSU  ;Restart reception if
05AA  74 17                            JE      NLMrst           ;  no command or message,
                                       ;
                                       TRACE   E1:TL-RCV-NLM
05AC  80 3E 000B E 00        +         CMP BYTE PTR EC.Trce,$True
05B1  75 0F                  +         JNE     ??0068
05B3  50                     +         PUSH    AX
05B4  B0 00                  +         MOV     AL,0
05B6  56                     +         PUSH    SI
05B7  BE 03C1 R              +         MOV     SI,OFFSET ??0069
05BA  E8 0000 E              +         CALL    EM_TRACE
05BD  5E                     +         POP     SI
05BE  58                     +         POP     AX
05BF  EB 01 90               +         JMP NEAR PTR ??0068
05C2                         + PC_CODE ENDS
03C1                         + PC_DATA SEGMENT PUBLIC 'DATA'
03C1  00                     + ??0069  LRB     (,OFFSET ??006B - OFFSET ??006A)
03C2  0D                     +
03C3  45 31 3A 54 4C 2D      + ??006A  DB      "E1:TL-RCV-NLM"
03D0                         + PC_DATA ENDS
05C2                         + PC_CODE SEGMENT PUBLIC 'CODE'
05C2                         + ??0068  LABEL   NEAR
                                       ;
05C2  C3                               RET                      ;Otherwise just exit
                                       ;
                                       ;       Reentry point to restart reception of messages
                                       ;
05C3  B0 01                    NLMrst: MOV     AL,$False        ;Set for no ORB,
05C5  BE 0000 E                        MOV     SI,OFFSET RcvNLM ;  assure TST pointer,
05C8  E8 0000 E                        CALL    TL_SET_RECV      ;  request restart
                                       ;
                                       TRACE   E2:TL-RCV-NLM
05CB  80 3E 000B E 00        +         CMP BYTE PTR EC.Trce,$True
05D0  75 0F                  +         JNE     ??006C
05D2  50                     +         PUSH    AX
05D3  B0 00                  +         MOV     AL,0
05D5  56                     +         PUSH    SI
05D6  BE 03D0 R              +         MOV     SI,OFFSET ??006D
05D9  E8 0000 E              +         CALL    EM_TRACE
05DC  5E                     +         POP     SI
05DD  58                     +         POP     AX
05DE  EB 01 90               +         JMP NEAR PTR ??006C
05E1                         + PC_CODE ENDS
03D0                         + PC_DATA SEGMENT PUBLIC 'DATA'
03D0  00                     + ??006D  LRB     (,OFFSET ??006F - OFFSET ??006E)
03D1  0D                     +
03D2  45 32 3A 54 4C 2D      + ??006E  DB      "E2:TL-RCV-NLM"
03DF                         + PC_DATA ENDS
05E1                         + PC_CODE SEGMENT PUBLIC 'CODE'
05E1                         + ??006C  LABEL   NEAR
                                       ;
05E1  C3                               RET                      ;Return to caller
                                       ;
05E2                           TL_RCV_NLM  ENDP
                                       ;
```

```
                        ;*****************************************************************************
                        ;Module name:  Fetch Internal Command (TL-FETCH-IC)                          |
                        ;                                                                            |
                        ;Version:  2.0                                                               |
                        ;                                                                            |
                        ;Last Update: 14 November 1983                                               |
                        ;                                                                            |
                        ;Function:  The Fetch Internal Command routine is entered when the           |
                        ;           TL-FETCH-LC routine encounters an internal commmand.  It         |
                        ;           fetches the data from the NLMT to the location specified         |
                        ;           by the input ORB.  Socket 80 reception is then restarted         |
                        ;           if a Local Command receive is not pending.                       |
                        ;                                                                            |
                        ;Procedure:  Program Logic Description, Section 2.7.7                        |
                        ;                                                                            |
                        ;Called By:  TL-FETCH-LC (via a long jump)                                   |
                        ;                                                                            |
                        ;Routines called:  TL-SET-RECV                                               |
                        ;                                                                            |
                        ;Input:  DS:(SI) = Location of the ORB                                       |
                        ;                                                                            |
                        ;Output:  Completion code returned in ORB                                    |
                        ;                                                                            |
                        ;Error procedures:  None                                                     |
                        ;                                                                            |
                        ;Written by: R.B. Talmadge, Computer Technology Ltd                          |
                        ;                                                                            |
                        ;Updates:  None                                                              |
                        ;                                                                            |
                        ;*****************************************************************************
                        ;
05E2                    TL_FETCH_IC  PROC  NEAR
                        ;
                                TRACE   S:TL-FETCH-IC
05E2  80 3E 000B E 00   +       CMP BYTE PTR EC.Trce,$True
05E7  75 0F             +       JNE     ??0070
05E9  50                +       PUSH    AX
05EA  B0 00             +       MOV     AL,0
05EC  56                +       PUSH    SI
05ED  BE 03DF R         +       MOV     SI,OFFSET ??0071
05F0  E8 0000 E         +       CALL    EM_TRACE
05F3  5E                +       POP     SI
05F4  58                +       POP     AX
05F5  EB 01 90          +       JMP NEAR PTR ??0070
05F8                    + PC_CODE ENDS
03DF                    + PC_DATA SEGMENT PUBLIC 'DATA'
03DF  00                + ??0071  LRB     (,OFFSET ??0073 - OFFSET ??0072)
03E0  0D                +
03E1  53 3A 54 4C 2D 46 + ??0072  DB      "S:TL-FETCH-IC"
03EE                    + PC_DATA ENDS
05F8                    + PC_CODE SEGMENT PUBLIC 'CODE'
05F8                    + ??0070  LABEL   NEAR
                        ;
05F8  56                        PUSH    SI              ;Save working registers
05F9  57                        PUSH    DI
```

```
05FA  51                              PUSH    CX
                                ;
05FB  C6 04 00                        MOV     [SI].CC,$OPcne  ;Set completion OK
05FE  8B 4C 02                        MOV     CX,[SI].Sze     ;Fetch count from ORB
0601  8B 7C 04                        MOV     DI,[SI].RB_Data    ;Location to place data
0604  BE 0212 R                       MOV     SI,OFFSET MT    ;Source starts at NLMT plus
0607  83 C6 08                        ADD     SI,TYPE CRQE    ;  offset of initial CRQE
                                ;
060A  FC                              CLD                     ;Set direction and
060B  F3/ A4                          REP     MOVSB           ;  move the data
                                ;
060D  BE 0000 E                       MOV     SI,OFFSET RcvNLM   ;Fetch TST entry receive NLM
0610  80 3C 01                        CMP     [SI].PN,$False     ;Go restart reception
0613  74 1A                           JE      ICrst              ;  if no local command
                                ;
0615  59                        ICxit: POP     CX              ;Restore registers
0616  5F                              POP     DI
0617  5E                              POP     SI
                                ;
                                       TRACE   E:TL-FETCH-IC
0618  80 3E 000B E 00           +         CMP BYTE PTR EC.Trce,$True
061D  75 0F                     +         JNE     ??0074
061F  50                        +         PUSH    AX
0620  B0 00                     +         MOV     AL,0
0622  56                        +         PUSH    SI
0623  BE 03EE R                 +         MOV     SI,OFFSET ??0075
0626  E8 0000 E                 +         CALL    EM_TRACE
0629  5E                        +         POP     SI
062A  58                        +         POP     AX
062B  EB 01 90                  +         JMP NEAR PTR ??0074
062E                            + PC_CODE ENDS
03EE                            + PC_DATA SEGMENT PUBLIC 'DATA'
03EE  00                        + ??0075  LRB     (,OFFSET ??0077 - OFFSET ??0076)
03EF  0D                        +
03F0  45 3A 54 4C 2D 46         + ??0076  DB      "E:TL-FETCH-IC"
03FD                            + PC_DATA ENDS
062E                            + PC_CODE SEGMENT PUBLIC 'CODE'
062E                            + ??0074  LABEL   NEAR
                                ;
062E  C3                              RET                     ;Return to caller
                                ;
                                ;     Reenter here to restart socket 80 reception
                                ;
062F  B0 01                     ICrst: MOV     AL,$False          ;Set for no ORB,
0631  E8 0000 E                       CALL    TL_SET_RECV        ;  request restart,
0634  EB DF                           JMP     ICxit              ;  and exit
                                ;
0636                            TL_FETCH_IC  ENDP
                                ;
                                ;*****************************************************************************
                                ;               End of the Assembly housekeeping goes here                  |
                                ;*****************************************************************************
                                ;
0636                            PC_CODE ENDS
                                ;
```

```
                PUBLIC  RQ              ;Command request queue
                PUBLIC  IDmsg           ;NL message image
                PUBLIC  MT              ;Name Lookup Message Table
                PUBLIC  EMstake         ;Initial stack location
                PUBLIC  TL_INT_DRVR     ;Interrupt Service Driver
                PUBLIC  TL_INT_DTME     ;Interrupt driver off timer tick
                PUBLIC  TL_DQ_CRQ       ;CRQ dequeue
                PUBLIC  TL_NQ_CRQ       ;CRQ enqueue
                PUBLIC  TL_SET_ERR      ;Set disk error code
                PUBLIC  TL_GET_NAD      ;Get NAD of requestor
                PUBLIC  TL_RCV_ICS      ;Receive ICS
                PUBLIC  TL_INT_NAD      ;Int svc routine for send NAD
                PUBLIC  TL_RCV_NLM      ;Receive NL message
                PUBLIC  TL_FETCH_IC     ;Internal command fetch
                PUBLIC  TL_INT_TEST     ;Primary interrupt test loop
                PUBLIC  TL_INT_WAIT     ;Wait for Operation complete
;
                END
```

Macros:

Name	Length
COMPARE.	000F
DISABLE.	0005
ENABLE	0004
ENTER.	0013
EXIT	0007
EYE.	0001
FILLIT	0011
FIV.	0002
GEN_START.	0002
LOG.	0005
MOVEB.	000F
MOVEONE.	000E
RESTORE.	0001
SCANB.	0012
SELEXIT.	0002
SELREST.	000D
SELSAVE.	0018
SIV.	0003
START.	001A
STATUS	0001
TENTER	0003
TEXIT.	0002
TIME	0001
TRACE.	0009

Structures and records:

Name	Width / Shift	# fields / Width	Mask	Initial
CCBLK.	000C	0009		
CC_CMND.	0000			
RRA.	0002			
SOCK	0004			
CC_DATA.	0006			
DATL	0008			
CONLN.	000A			
DHOST.	000B			
CRQE	0008	0004		
USE.	0000			
NAD.	0001			
LEN.	0002			
CMND	0004			
CRQH	0008	0005		
QST.	0000			
DSC.	0001			
END.	0002			
TOP.	0004			
BOT.	0006			
ECT.	0040	001A		
IDEN	0000			
EC_DAT	0008			

```
  EC_DRV . . . . . .  0009
  EC_FLG . . . . . .  000A
  TRCE . . . . . .  000B
  COLD . . . . . .  000C
  EC_TRM . . . . . .  000D
  MAX. . . . . .  000E
  NEXT . . . . . .  0010
  DIR. . . . . .  0012
  NAM. . . . . .  001A
  SERV . . . . . .  0022
  EC_ATRU. . . . . .  002C
  EC_ATRS. . . . . .  002D
  EC_ATRL. . . . . .  002E
  EC_TBI . . . . . .  002F
  TLTMCK . . . . . .  0030
  TLTMRS . . . . . .  0031
  KEEPOUT. . . . . .  0032
  INTDFR . . . . . .  0033
  DOSDKO . . . . . .  0034
  DOSDKS . . . . . .  0036
  DDRH . . . . . .  0038
  DDRHS. . . . . .  003A
  TCKO . . . . . .  003C
  TCKS . . . . . .  003E
LRB. . . . . .  0002  0002
  CDE. . . . . .  0000
  LR_LEN . . . . . .  0001
NLM. . . . . .  0012  0005
  PID. . . . . .  0000
  MGT. . . . . .  0002
  SRC. . . . . .  0004
  DEV. . . . . .  0006
  NL_NAME. . . . . .  0008
NLMT . . . . . .  0016  0009
  MT_USE . . . . . .  0000
  MT_NAD . . . . . .  0001
  OPC. . . . . .  0004
  MT_MOD . . . . . .  0005
  MT_NAME. . . . . .  0006
  MT_SRC . . . . . .  0010
  MT_DEV . . . . . .  0011
  RSVD . . . . . .  0012
ORB. . . . . .  0008  0005
  CC . . . . . .  0000
  OP . . . . . .  0001
  SZE. . . . . .  0002
  RB_DATA. . . . . .  0004
  RB_DATAS . . . . . .  0006
RRKD . . . . . .  0004  0003
  RCODE. . . . . .  0000
  SHOST. . . . . .  0001
  RCVLEN . . . . . .  0002
RRRCV. . . . . .  0004  0002
  LOFCM. . . . . .  0000
  ELOFRP . . . . . .  0002
```

```
RRSND. . . . . .          0003    0002
  ALOFRP . . . . . .      0000
  DSTAT. . . . . .        0002
TSTE . . . . . .          0014    000C
  PN . . . . . .          0000
  TT_ST. . . . . .        0001
  SK . . . . . .          0002
  TT_OP. . . . . .        0003
  CCB. . . . . .          0004
  RR . . . . . .          0006
  DTL. . . . . .          0008
  DSZ. . . . . .          000A
  ISR. . . . . .          000C
  ISRS . . . . . .        000E
  TT_ORB . . . . . .      0010
  TT_ORBS. . . . . .      0012
TSTH . . . . . .          0004    0004
  INT. . . . . .          0000
  TT_CI. . . . . .        0001
  NE . . . . . .          0002
  TT_NAD . . . . . .      0003
TSTLE. . . . . .          0014    000B
  LE_CC. . . . . .        0000
  LE_OP. . . . . .        0001
  LE_LEN . . . . . .      0002
  TRM. . . . . .          0004
  TRMS . . . . . .        0006
  RCV. . . . . .          0008
  RCVS . . . . . .        000A
  LE_ISR . . . . . .      000C
  LE_ISRS. . . . . .      000E
  LE_BP. . . . . .        0010
```

Segments and groups:

N a m e	Size	align	combine	class
PCS.	GROUP			
PC_CODE.	0636	PARA	PUBLIC	'CODE'
PC_DATA.	03FD	PARA	PUBLIC	'DATA'

Symbols:

N a m e	Type	Value	Attr	
CI$.	Number	0000		
CI_MAIN.	L NEAR	0000		External
DEBUG$	Number	0001		
DM$.	Number	0000		
DM_ERR_LOG	L NEAR	0000		External
DQFRST	L NEAR	0210	PC_CODE	
DQOK	L NEAR	01E9	PC_CODE	
DQOK1.	L NEAR	01FC	PC_CODE	
DQOK2.	L NEAR	0208	PC_CODE	
DQS0	L NEAR	019E	PC_CODE	

DQS00.	L NEAR	01BB	PC_CODE	
DQST	L NEAR	01BE	PC_CODE	
DQST1.	L NEAR	01D0	PC_CODE	
DR$.	Number	0000		
DRCTST	L NEAR	0072	PC_CODE	
DRINC.	L NEAR	0020	PC_CODE	
DRTSDS	L NEAR	0035	PC_CODE	
DRTXIT	L NEAR	002C	PC_CODE	
DRVCCI	L NEAR	0102	PC_CODE	
DRVGO.	L NEAR	00C2	PC_CODE	
DRVLPE	L NEAR	00C6	PC_CODE	
DRVLPN	L NEAR	00DD	PC_CODE	
DRVLS.	L NEAR	00CA	PC_CODE	
DRVTCI	L NEAR	00F4	PC_CODE	
DRVXIT	L NEAR	0113	PC_CODE	
DTA_BGN.	V BYTE	0000		External
EC	V BYTE	0000		External
EI$.	Number	0000		
EMSTAK	L WORD	020C	PC_DATA	
EMSTAKE.	E BYTE	020E	PC_DATA	Global
EM_TRACE	L NEAR	0000		External
EYE$	Number	0001		
HINQ	L NEAR	045E	PC_CODE	
HWDRVR	L NEAR	0015	PC_CODE	
ICRST.	L NEAR	062F	PC_CODE	
ICXIT.	L NEAR	0615	PC_CODE	
IDMSG.	L 0012	022C	PC_DATA	Global
INTBRK	L NEAR	0095	PC_CODE	
INTXIT	L NEAR	008B	PC_CODE	
IS$.	Number	0000		
LC$.	Number	0000		
LC_INT_SRVR.	L NEAR	0000		External
LC_RCV_CMND.	L NEAR	0000		External
MT	L 0016	0212	PC_DATA	Global
NA$.	Number	0000		
NADGO.	L NEAR	0324	PC_CODE	
NADXT.	L NEAR	0331	PC_CODE	
NLMCAL	L NEAR	05A3	PC_CODE	
NLMERR	L NEAR	0514	PC_CODE	
NLMMG1	L NEAR	0525	PC_CODE	
NLMMG2	L NEAR	0535	PC_CODE	
NLMMSG	L NEAR	04F5	PC_CODE	
NLMRST	L NEAR	05C3	PC_CODE	
NLMRTN	L NEAR	05A5	PC_CODE	
NLMSTP	L NEAR	051D	PC_CODE	
NOCOM$	Number	0000		
NQMOV.	L NEAR	02AA	PC_CODE	
NQOK	L NEAR	026E	PC_CODE	
NQST	L NEAR	028B	PC_CODE	
NQST1.	L NEAR	029B	PC_CODE	
NQXIT.	L NEAR	02BC	PC_CODE	
PI$.	Number	0000		
RAX$	Number	0001		
RBP$	Number	0000		
RBX$	Number	0001		

```
RCVERR . . . . . .  L NEAR  037E  PC_CODE
RCVICS . . . . . .  V BYTE  0000          External
RCVMSG . . . . . .  L NEAR  038E  PC_CODE
RCVMSH . . . . . .  L NEAR  03AB  PC_CODE
RCVNLM . . . . . .  V BYTE  0000          External
RCVRST . . . . . .  L NEAR  03CA  PC_CODE
RCVSTP . . . . . .  L NEAR  0386  PC_CODE
RCVXIT . . . . . .  L NEAR  03CC  PC_CODE
RCX$ . . . . . .  Number  0001
RDI$ . . . . . .  Number  0001
RDS$ . . . . . .  Number  0000
RDX$ . . . . . .  Number  0001
RES$ . . . . . .  Number  0000
RQ . . . . . .  L QWORD  0004  PC_DATA Global
RQBGN. . . . . .  L QWORD  000C  PC_DATA
RQEND. . . . . .  E BYTE  020C  PC_DATA
RSI$ . . . . . .  Number  0001
SM$. . . . . .  Number  0000
SSWCH$ . . . . . .  Number  0000
SWDRVR . . . . . .  L NEAR  009F  PC_CODE
TBINTA$. . . . . .  Number  0001
TENV$. . . . . .  Number  0001
TL$. . . . . .  Number  0001
TLTGO. . . . . .  L NEAR  012B  PC_CODE
TLTXIT . . . . . .  L NEAR  0126  PC_CODE
TL_BYT_IN. . . . . .  L NEAR  0000          External
TL_BYT_OUT . . . . . .  L NEAR  0000          External
TL_DQ_CRQ. . . . . .  N PROC  0188  PC_CODE Global  Length =00B2
TL_DTA_IN. . . . . .  L NEAR  0000          External
TL_FETCH_IC. . . . . .  N PROC  05E2  PC_CODE Global  Length =0054
TL_GET_NAD . . . . . .  N PROC  02EE  PC_CODE Global  Length =005B
TL_INT_DRVR. . . . . .  N PROC  0000  PC_CODE Global  Length =0138
TL_INT_DTME. . . . . .  L NEAR  0114  PC_CODE Global
TL_INT_NAD . . . . . .  L NEAR  042C  PC_CODE Global
TL_INT_TEST. . . . . .  L NEAR  00C2  PC_CODE Global
TL_INT_WAIT. . . . . .  N PROC  0138  PC_CODE Global  Length =0050
TL_NLM_SRVC. . . . . .  N PROC  0443  PC_CODE Length =0094
TL_NQ_CRQ. . . . . .  N PROC  023A  PC_CODE Global  Length =009A
TL_RCV_BYE . . . . . .  L NEAR  0465  PC_CODE
TL_RCV_HI. . . . . .  L NEAR  0443  PC_CODE
TL_RCV_ICS . . . . . .  N PROC  0349  PC_CODE Global  Length =009F
TL_RCV_IDEN. . . . . .  L NEAR  0443  PC_CODE
TL_RCV_NLM . . . . . .  N PROC  04D7  PC_CODE Global  Length =010B
TL_RCV_WHERE . . . . . .  L NEAR  04B2  PC_CODE
TL_RCV_WHO . . . . . .  L NEAR  0482  PC_CODE
TL_SEND_NAD. . . . . .  N PROC  03E8  PC_CODE Length =005B
TL_SEND_NLM. . . . . .  L NEAR  0000          External
TL_SET_ERR . . . . . .  N PROC  02D4  PC_CODE Global  Length =001A
TL_SET_RECV. . . . . .  L NEAR  0000          External
TL_SET_RTRY. . . . . .  L NEAR  0000          External
TL_SET_SEND. . . . . .  L NEAR  0000          External
TL_WRD_IN. . . . . .  L NEAR  0000          External
TMPICS . . . . . .  L QWORD  023E  PC_DATA
TRACE$ . . . . . .  Number  0001
TRMNAD . . . . . .  V BYTE  0000          External
```

```
TT . . . . . . . . . . . . . . . .  V BYTE  0000            External
TTBGN. . . . . . . . . . . . . . .  V BYTE  0000            External
TTEND. . . . . . . . . . . . . . .  V BYTE  0000            External
WAITCC . . . . . . . . . . . . . .  L NEAR  0156    PC_CODE
WAITNI . . . . . . . . . . . . . .  L NEAR  0172    PC_CODE
WAITXT . . . . . . . . . . . . . .  L NEAR  015B    PC_CODE
WHOSND . . . . . . . . . . . . . .  L NEAR  04A7    PC_CODE
$$DIR. . . . . . . . . . . . . . .  Number  0008
$$NAM. . . . . . . . . . . . . . .  Number  0008
$$NEXT . . . . . . . . . . . . . .  Number  0004
$$SERV . . . . . . . . . . . . . .  Number  000A
$ALLNAD. . . . . . . . . . . . . .  Number  00FF
$ATRNRM. . . . . . . . . . . . . .  Number  0000
$CALLCA. . . . . . . . . . . . . .  Number  0023
$CALLCB. . . . . . . . . . . . . .  Number  001B
$CALLDI. . . . . . . . . . . . . .  Number  0066
$CALLDS. . . . . . . . . . . . . .  Number  0021
$CALLEM. . . . . . . . . . . . . .  Number  000A
$CALLKR. . . . . . . . . . . . . .  Number  0034
$CALLTK. . . . . . . . . . . . . .  Number  001C
$CALLTM. . . . . . . . . . . . . .  Number  0067
$CALLTP. . . . . . . . . . . . . .  Number  0001
$CLINT . . . . . . . . . . . . . .  Number  024D
$CONS. . . . . . . . . . . . . . .  Number  0080
$CONSPT. . . . . . . . . . . . . .  Number  0040
$DIINT . . . . . . . . . . . . . .  Number  024C
$ECHO. . . . . . . . . . . . . . .  Number  0002
$ENDRCV. . . . . . . . . . . . . .  Number  0010
$ENINT . . . . . . . . . . . . . .  Number  024E
$FALSE . . . . . . . . . . . . . .  Number  0001
$INITT . . . . . . . . . . . . . .  Number  0020
$INITTA. . . . . . . . . . . . . .  Number  0100
$INT$. . . . . . . . . . . . . . .  Number  00CD
$LCABT . . . . . . . . . . . . . .  Number  00FF
$LCAOK . . . . . . . . . . . . . .  Number  0000
$LCMAXL. . . . . . . . . . . . . .  Alias   $MAXTBD
$LCNOP . . . . . . . . . . . . . .  Alias   $FALSE
$LCOPN . . . . . . . . . . . . . .  Alias   $TRUE
$MAXDLY. . . . . . . . . . . . . .  Number  7FFF
$MAXMSG. . . . . . . . . . . . . .  Number  0066
$MAXRTY. . . . . . . . . . . . . .  Number  0010
$MAXTBD. . . . . . . . . . . . . .  Number  0211
$MAXWTE. . . . . . . . . . . . . .  Number  000A
$MND00 . . . . . . . . . . . . . .  Number  0001
$MND01 . . . . . . . . . . . . . .  Number  0009
$MND02 . . . . . . . . . . . . . .  Number  000A
$MND03 . . . . . . . . . . . . . .  Number  000B
$MND04 . . . . . . . . . . . . . .  Number  000C
$MND05 . . . . . . . . . . . . . .  Number  0011
$MND06 . . . . . . . . . . . . . .  Number  0013
$MNT00 . . . . . . . . . . . . . .  Number  0002
$MNT01 . . . . . . . . . . . . . .  Number  0003
$MNT010. . . . . . . . . . . . . .  Number  0010
$MNT011. . . . . . . . . . . . . .  Number  0012
$MNT012. . . . . . . . . . . . . .  Number  0014
```

```
$MNT02 . . . . . Number  0004
$MNT03 . . . . . Number  0005
$MNT04 . . . . . Number  0006
$MNT05 . . . . . Number  0007
$MNT06 . . . . . Number  0008
$MNT07 . . . . . Number  000D
$MNT08 . . . . . Number  000E
$MNT09 . . . . . Number  000F
$MTADD . . . . . Number  0003
$MTDEL . . . . . Number  0000
$NADOK . . . . . Number  0040
$NAMELN. . . . . Number  000A
$NLANY . . . . . Number  0001
$NLBYE . . . . . Number  00FF
$NLDNY . . . . . Number  00FF
$NLDSK . . . . . Number  0001
$NLHI. . . . . Number  0000
$NLIDN . . . . . Number  0010
$NLWHO . . . . . Number  0002
$NLWHR . . . . . Number  0003
$ONE . . . . . Number  0001
$OPCNE . . . . . Number  0000
$OPINL . . . . . Number  0001
$OPINP . . . . . Number  00FF
$OPTCL . . . . . Number  0007
$OPTDF . . . . . Number  0004
$OPTNG . . . . . Number  0005
$OPTRB . . . . . Number  0002
$OPTTF . . . . . Number  0006
$OPTTO . . . . . Number  0003
$P82EML. . . . . Number  0005
$P82ENB. . . . . Number  00FA
$P82EOI. . . . . Number  0020
$P82OP0. . . . . Number  0020
$P82OP1. . . . . Number  0021
$P82RIS. . . . . Number  000B
$P82TML. . . . . Number  0001
$PROID . . . . . Number  01FE
$RDRAM . . . . . Number  0249
$RDRAMI. . . . . Number  024B
$RDSTAT. . . . . Number  0248
$RQCSZE. . . . . Number  0004
$RQDSE . . . . . Number  0080
$RQENT . . . . . Number  0001
$RQFUL . . . . . Number  00FF
$RQMT. . . . . Number  0000
$RQNUM . . . . . Number  003F
$RQSZE . . . . . Number  01F8
$RQUSEE. . . . . Number  0080
$RQUSEI. . . . . Number  0010
$RQUSEL. . . . . Number  0020
$RQUSER. . . . . Number  0040
$SEND. . . . . Number  0040
$SETUP . . . . . Number  00F0
$SKNUM . . . . . Number  0060
```

```
$SNDNAD. . . . . .  Number  00CF
$SOKRC . . . . . .  Number  00B0
$SOKRL . . . . . .  Number  00A0
$SOKRM . . . . . .  Number  0080
$SOKTD . . . . . .  Number  00B0
$SOKTM . . . . . .  Number  0080
$STAK. . . . . .  Number  0016
$STAKK . . . . . .  Number  0001
$STINT . . . . . .  Number  024F
$TBINIT. . . . . .  Number  0010
$TBINTI. . . . . .  Number  0080
$TBNOTK. . . . . .  Number  00DF
$TBRENB. . . . . .  Number  0040
$TBRTCK. . . . . .  Number  0020
$TICKMX. . . . . .  Number  000A
$TRAPFL. . . . . .  Number  0001
$TRCCON. . . . . .  Number  0002
$TRCDSK. . . . . .  Number  0001
$TRCHRD. . . . . .  Number  0004
$TRCL. . . . . .  Number  0083
$TRDL. . . . . .  Number  0081
$TREA. . . . . .  Number  00C0
$TREDY . . . . . .  Number  0080
$TRENB . . . . . .  Number  0020
$TRIS. . . . . .  Number  0084
$TRIT. . . . . .  Number  0086
$TRLAT . . . . . .  Number  0008
$TRNG. . . . . .  Number  0080
$TRNR. . . . . .  Number  0085
$TRNS. . . . . .  Number  0082
$TROK. . . . . .  Number  0000
$TRPND . . . . . .  Number  0010
$TRSS. . . . . .  Number  00FE
$TRST. . . . . .  Number  007F
$TRSU. . . . . .  Number  00FF
$TRUE. . . . . .  Number  0000
$VIR_DRV1. . . . . .  Number  0001
$WHOAMI. . . . . .  Number  0001
$WRCTRH. . . . . .  Number  0248
$WRCTRL. . . . . .  Number  024A
$WRRAM . . . . . .  Number  024B
$WRSTBE. . . . . .  Number  0249
$ZERO. . . . . .  Number  0000
??0000 . . . . . .  L NEAR  008B    PC_CODE
??0001 . . . . . .  L WORD  0246    PC_DATA
??0002 . . . . . .  L BYTE  0248    PC_DATA
??0003 . . . . . .  E BYTE  025A    PC_DATA
??0004 . . . . . .  L NEAR  00B8    PC_CODE
??0005 . . . . . .  L WORD  025A    PC_DATA
??0006 . . . . . .  L BYTE  025C    PC_DATA
??0007 . . . . . .  E BYTE  026E    PC_DATA
??0008 . . . . . .  L NEAR  014E    PC_CODE
??0009 . . . . . .  L WORD  026E    PC_DATA
??000A . . . . . .  L BYTE  0270    PC_DATA
??000B . . . . . .  E BYTE  027D    PC_DATA
```

??000C	L NEAR	0171	PC_CODE
??000D	L WORD	027D	PC_DATA
??000E	L BYTE	027F	PC_DATA
??000F	E BYTE	028C	PC_DATA
??0010	L NEAR	019E	PC_CODE
??0011	L WORD	028C	PC_DATA
??0012	L BYTE	028E	PC_DATA
??0013	E BYTE	0299	PC_DATA
??0014	L NEAR	01BB	PC_CODE
??0015	L WORD	0299	PC_DATA
??0016	L BYTE	029B	PC_DATA
??0017	E BYTE	02A6	PC_DATA
??0018	L NEAR	0239	PC_CODE
??0019	L WORD	02A6	PC_DATA
??001A	L BYTE	02A8	PC_DATA
??001B	E BYTE	02B3	PC_DATA
??001C	L NEAR	0250	PC_CODE
??001D	L WORD	02B3	PC_DATA
??001E	L BYTE	02B5	PC_DATA
??001F	E BYTE	02C0	PC_DATA
??0020	L NEAR	026D	PC_CODE
??0021	L WORD	02C0	PC_DATA
??0022	L BYTE	02C2	PC_DATA
??0023	E BYTE	02C3	PC_DATA
??0024	L NEAR	02D3	PC_CODE
??0025	L WORD	02C3	PC_DATA
??0026	L BYTE	02C5	PC_DATA
??0027	E BYTE	02D0	PC_DATA
??0028	L NEAR	02EA	PC_CODE
??0029	L WORD	02D0	PC_DATA
??002A	L BYTE	02D2	PC_DATA
??002B	E BYTE	02E0	PC_DATA
??002C	L NEAR	0304	PC_CODE
??002D	L WORD	02E0	PC_DATA
??002E	L BYTE	02E2	PC_DATA
??002F	E BYTE	02EE	PC_DATA
??0030	L NEAR	0323	PC_CODE
??0031	L WORD	02EE	PC_DATA
??0032	L BYTE	02F0	PC_DATA
??0033	E BYTE	02FC	PC_DATA
??0034	L NEAR	0348	PC_CODE
??0035	L WORD	02FC	PC_DATA
??0036	L BYTE	02FE	PC_DATA
??0037	E BYTE	030A	PC_DATA
??0038	L NEAR	035F	PC_CODE
??0039	L WORD	030A	PC_DATA
??003A	L BYTE	030C	PC_DATA
??003B	E BYTE	0318	PC_DATA
??003C	L NEAR	037D	PC_CODE
??003D	L WORD	0318	PC_DATA
??003E	L BYTE	031A	PC_DATA
??003F	E BYTE	0326	PC_DATA
??0040	L NEAR	03E7	PC_CODE
??0041	L WORD	0326	PC_DATA
??0042	L BYTE	0328	PC_DATA

```
??0043 . . . . . .  E BYTE  0332  PC_DATA
??0044 . . . . . .  L NEAR  03FE  PC_CODE
??0045 . . . . . .  L WORD  0332  PC_DATA
??0046 . . . . . .  L BYTE  0334  PC_DATA
??0047 . . . . . .  E BYTE  0341  PC_DATA
??0048 . . . . . .  L NEAR  042B  PC_CODE
??0049 . . . . . .  L WORD  0341  PC_DATA
??004A . . . . . .  L BYTE  0343  PC_DATA
??004B . . . . . .  E BYTE  0350  PC_DATA
??004C . . . . . .  L NEAR  0442  PC_CODE
??004D . . . . . .  L WORD  0350  PC_DATA
??004E . . . . . .  L BYTE  0352  PC_DATA
??004F . . . . . .  E BYTE  0360  PC_DATA
??0050 . . . . . .  L NEAR  0459  PC_CODE
??0051 . . . . . .  L WORD  0360  PC_DATA
??0052 . . . . . .  L BYTE  0362  PC_DATA
??0053 . . . . . .  E BYTE  0373  PC_DATA
??0054 . . . . . .  L NEAR  047B  PC_CODE
??0055 . . . . . .  L WORD  0373  PC_DATA
??0056 . . . . . .  L BYTE  0375  PC_DATA
??0057 . . . . . .  E BYTE  0383  PC_DATA
??0058 . . . . . .  L NEAR  0498  PC_CODE
??0059 . . . . . .  L WORD  0383  PC_DATA
??005A . . . . . .  L BYTE  0385  PC_DATA
??005B . . . . . .  E BYTE  0393  PC_DATA
??005C . . . . . .  L NEAR  04C8  PC_CODE
??005D . . . . . .  L WORD  0393  PC_DATA
??005E . . . . . .  L BYTE  0395  PC_DATA
??005F . . . . . .  E BYTE  03A5  PC_DATA
??0060 . . . . . .  L NEAR  04ED  PC_CODE
??0061 . . . . . .  L WORD  03A5  PC_DATA
??0062 . . . . . .  L BYTE  03A7  PC_DATA
??0063 . . . . . .  E BYTE  03B3  PC_DATA
??0064 . . . . . .  L NEAR  0513  PC_CODE
??0065 . . . . . .  L WORD  03B3  PC_DATA
??0066 . . . . . .  L BYTE  03B5  PC_DATA
??0067 . . . . . .  E BYTE  03C1  PC_DATA
??0068 . . . . . .  L NEAR  05C2  PC_CODE
??0069 . . . . . .  L WORD  03C1  PC_DATA
??006A . . . . . .  L BYTE  03C3  PC_DATA
??006B . . . . . .  E BYTE  03D0  PC_DATA
??006C . . . . . .  L NEAR  05E1  PC_CODE
??006D . . . . . .  L WORD  03D0  PC_DATA
??006E . . . . . .  L BYTE  03D2  PC_DATA
??006F . . . . . .  E BYTE  03DF  PC_DATA
??0070 . . . . . .  L NEAR  05F8  PC_CODE
??0071 . . . . . .  L WORD  03DF  PC_DATA
??0072 . . . . . .  L BYTE  03E1  PC_DATA
??0073 . . . . . .  E BYTE  03EE  PC_DATA
??0074 . . . . . .  L NEAR  062E  PC_CODE
??0075 . . . . . .  L WORD  03EE  PC_DATA
??0076 . . . . . .  L BYTE  03F0  PC_DATA
??0077 . . . . . .  E BYTE  03FD  PC_DATA

Warning Severe
```

Errors Errors
0   0

```
                        ;
                        PAGE 60,132
                        ;
                        ;*****************************************************************************
                        ;                       Filename is PCSCH2B.ASM                               |
                        ;*****************************************************************************
                        ; This file is the second of two which contain the code for all of the        |
                        ; routines described in Sections 2.1 through 2.7 of the Program Logic         |
                        ; Description, OMNINET PCShare Disk Server Emulator, Version 2.0, dated       |
                        ; 7 November 1983.  The file also contains allocation statements for          |
                        ; data structures managed by the Transporter Logical Device.                  |
                        ;*****************************************************************************
                        ;
                                ENDIF
                        ;
                        ;*****************************************************************************
                        ;
                                GEN_START   <TL>        ;TLD structures
                      +         .LALL
                      +         ENDIF
                      +         ENDIF
                      +         ENDIF
                      +         IFIDN   <TRACE>,<TRACE>
= 0001                + TRACE$  =       1
                      +         ENDIF
                      +         ENDIF
                       +        ENDIF
                       +        ENDIF
                       +        ENDIF
                       +        ENDIF
                       +        ENDIF
                       +        IFIDN   <DEBUG>,<DEBUG>
= 0001                 + DEBUG$  =       1
                       +        ENDIF
                       +        ENDIF
                      C+        INCLUDE PCSTRUC.CRV
                      C+ ;*****************************************************************************
                      C+ ;                       Filename is PCSTRUC.CRV                               |
                      C+ ;*****************************************************************************
                      C+ ; This file contains all structure definitions for the implementation      |
                      C+ ; of the Corvus OMNINET PCShare Disk Server Emulator.  It is included      |
                      C+ ; in each of the assembly files when the START macro is invoked.  The      |
                      C+ ; selectors specified with that macro invocation cause the associated      |
                      C+ ; structure definitions to be included in the assembly.  Any set of       |
                      C+ ; definitions can be included with an appropriate set of selectors.        |
                      C+ ;*****************************************************************************
                      C+ ;                                                                             |
                      C+ ; Version:  2.0                                                               |
                      C+ ;                                                                             |
                      C+ ; Last Update:  14 January 1984                                               |
                      C+ ;                                                                             |
                      C+ ;*****************************************************************************
                      C+ ENDIF
                      C+ ;
```

```
                        C+ ENDIF
                        C+ ;
                        C+ ;--------------------------------------------------------------------------
                        C+ ;                    Emulator Resident Data Structures                  |
                        C+ ;--------------------------------------------------------------------------
                        C+ ;
                        C+ ;       Universal value definitions
                        C+ ;
= 0000                  C+ $True    EQU     0               ;Usually indicates OK or normal
= 0001                  C+ $False   EQU     1               ;Usually indicates NG or unusual
= 0000                  C+ $Zero    EQU     0               ;What is there to say
= 0001                  C+ $One     EQU     1               ;  about these?
= 0060                  C+ $SKnum   EQU     96              ;Number of internal stack entries
= 0001                  C+ $Vir_Drv1 EQU    1               ;Value for Virtual Drive 1
                        C+ ;
                        C+ ;--------------------------------------------------------------------------
                        C+ ;       8259A PIC usage values                                           |
                        C+ ;       Reference: Intel 8086 User's Manual                              |
                        C+ ;--------------------------------------------------------------------------
                        C+ ;
= 0021                  C+ $P82op1 EQU      21H             ;Port for OCW1 commands
= 0020                  C+ $P82op0 EQU      20H             ;Port for OCW2, OCW3 commands
                        C+ ;
= 000B                  C+ $P82Ris EQU      0BH             ;Read Interrupt Service register
= 0020                  C+ $P82Eoi EQU      20H             ;Simple end of interrupt
= 0001                  C+ $P82TML EQU      1               ;Timer interrupt level bit
= 0005                  C+ $P82EML EQU      4+$P82TML       ;Transporter and timer levels
= 00FA                  C+ $P82enb EQU      0FFH-$P82EML    ;Enable xporter and timer
                        C+ ;
                        C+ ;--------------------------------------------------------------------------
                        C+ ;       Structure Name:  Emulator Communication Table                    |
                        C+ ;       Reference: Program Logic description, Section 1.4                |
                        C+ ;--------------------------------------------------------------------------
                        C+ ;
                        C+ ;       Associated values and flags
                        C+ ;
= 0080                  C+ $Cons    EQU     80H             ;Write messages to console
= 0040                  C+ $ConsPT  EQU     40H             ;Prompt for console at initialization
= 0000                  C+ $AtrNrm  EQU     0               ;Standard attributes of Emulator files
= 000A                  C+ $TickMX  EQU     10              ;Default timer tick refresh count
                        C+ ;
= 0080                  C+ $TBintI  EQU     80H             ;Transporter interrupts installed
= 0040                  C+ $TBrenb  EQU     40H             ;Transporter to run enabled
= 0020                  C+ $TBrtck  EQU     20H             ;Transporter to run off timer tick
= 00DF                  C+ $TBnotk  EQU     0FFH-$TBrtck    ;Mask to clear timer tick on
= 0010                  C+ $TBinit  EQU     10H             ;Prompt for interrupts at init
= 0004                  C+ $TRChrd  EQU     4               ;Trace output tp hard disk
= 0002                  C+ $TRCcon  EQU     2               ;Trace output to console
= 0001                  C+ $TRCdsk  EQU     1               ;Trace output to disk
                        C+ ;
                        C+ ;       Interrupt vector numbers
                        C+ ;
= 000A                  C+ $CallEM  EQU     0AH             ;Emulator interrupt
= 001B                  C+ $CallCB  EQU     1BH             ;Control_break has occurred
= 001C                  C+ $CallTK  EQU     1CH             ;Timer interrupt appendage
```

```
= 0021                          C+ $CallDS EQU     21H             ;DOS function call
= 0023                          C+ $CallCA EQU     23H             ;Control_break appendage
= 0034                          C+ $CallKR EQU     34H             ;Gets location In-DOS flag
= 0066                          C+ $CallDI EQU     66H             ;DOS intercept
= 0067                          C+ $CallTM EQU     67H             ;Timer appendage recall
= 0001                          C+ $CallTP EQU     1               ;Trap interrupt
                                C+ ;
= 00CD                          C+ $INT$   EQU     0CDH            ;Interrupt instruction code
= 0001                          C+ $TrapFL EQU     1               ;Position of trap flag in AH
                                C+ ;
                                C+ ;       Field lengths
                                C+ ;
= 0008                          C+ $$Dir   EQU     8
= 0008                          C+ $$Nam   EQU     8
= 0004                          C+ $$Next  EQU     4
= 000A                          C+ $$Serv  EQU     10
                                C+ ;
                                C+ ;
                                C+ ECT     STRUC
0000  50 43 53 48 41 52         C+ Iden    DB      'PCSHARE2'      ;Identifier of the version
      45 32                     C+
0008  43                        C+ EC_Dat  DB      'C'             ;Drive for Emulator files (character)
0009  03                        C+ EC_Drv  DB      3               ;Same drive numeric    !NEW FIELD!
000A  40                        C+ EC_Flg  DB      $ConsPT         ;Flag for log output to console
000B  01                        C+ Trce    DB      $False          ;Trace active switch   !NEW FIELD!
000C  01                        C+ Cold    DB      $False          ;Cold start completed  !NEW FIELD!
000D  01                        C+ EC_Trm  DB      $False          ;Emulator terminated   !NEW FIELD!
000E  0064                      C+ Max     DW      100             ;Maximum number records in log file
0010  0000                      C+ Next    DW      0               ;RRN for next log record
0012  50 43 53 48 41 52         C+ Dir     DB      'PCSHARE '      ;Name of directory for Emulator files
      45 20                     C+
001A  50 43 53 48 41 52         C+ Nam     DB      'PCSHARE '      ;Filename for Emulator files
      45 20                     C+
0022     0A [                   C+ Serv    DB      $$Serv DUP(' ') ;Emulator name as a disk server
               20               C+
                  ]             C+
                                C+
002C  00                        C+ EC_AtrU DB      $AtrNrm         ;Attributes of User area file      !N F!
002D  00                        C+ EC_AtrS DB      $AtrNrm         ;Attributes of System area files   !E I!
002E  00                        C+ EC_AtrL DB      $AtrNrm         ;Attributes of Log file            !W E!
002F  C0                        C+ EC_TBI  DB      $TBintI+$TBrenb ;Transporter int flags             !  L!
0030  0A                        C+ TLtmck  DB      $TickMX         ;Current timer tick count          !  D!
0031  0A                        C+ TLtmrs  DB      $TickMX         ;Value to reset timer tick count   !  S!
0032  01                        C+ KEEPout DB      $False          ;DOS has been entered flag         !   !
0033  01                        C+ INTdfr  DB      $False          ;Deferred interrupt flag           !   !
0034  0000                      C+ DOSdkO  DW      0               ;Holds location of DOS             !   !
0036  0000                      C+ DOSdkS  DW      0               ;  Critical section (In-DOS) Flag  !   !
0038  0000                      C+ DDRH    DW      0               ;Holds location of request header when
003A  0000                      C+ DDRHS   DW      0               ;   Device Driver called     !NEW FIELD!
003C  0000                      C+ TckO    DW      0               ;Holds location of timer tick int when
003E  0000                      C+ TckS    DW      0               ;   no transporter ints      !NEW FIELD!
0040                            C+ ECT     ENDS
                                C+ ;
                                C+ ;-------------------------------------------------------------------------
                                C+ ;       Structure Name:  Log Request Block                             !
```

```
                        C+ ;       Reference: Program Logic Description, Section 4.27
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ ;       Associated values
                        C+ ;
= 0066                  C+ $MAXMSG EQU     102             ;Maximum message length
                        C+ ;
                        C+ ;       Log message numbers
                        C+ ;
= 0001                  C+ $MND00  EQU     1               ;Initialization message
= 0002                  C+ $MNT00  EQU     2
= 0003                  C+ $MNT01  EQU     3
= 0004                  C+ $MNT02  EQU     4
= 0005                  C+ $MNT03  EQU     5
= 0006                  C+ $MNT04  EQU     6
= 0007                  C+ $MNT05  EQU     7
= 0008                  C+ $MNT06  EQU     8
= 0009                  C+ $MND01  EQU     9
= 000A                  C+ $MND02  EQU     10
= 000B                  C+ $MND03  EQU     11
= 000C                  C+ $MND04  EQU     12
= 000D                  C+ $MNT07  EQU     13
= 000E                  C+ $MNT08  EQU     14
= 000F                  C+ $MNT09  EQU     15
= 0010                  C+ $MNT010 EQU     16
= 0011                  C+ $MND05  EQU     17
= 0012                  C+ $MNT011 EQU     18
= 0013                  C+ $MND06  EQU     19              ;ORB missing
= 0014                  C+ $MNT012 EQU     20              ;Bad pipes tables
                        C+ ;
                        C+ LRB     STRUC
0000  00                C+ Cde     DB      0               ;Module ID code
0001  00                C+ LR_Len  DB      0               ;Length of the message string
0002                    C+ LRB     ENDS
                        C+ ;
                        C+ IF (TL$ OR CI$ OR DM$ OR LC$)
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Operation Request Block
                        C+ ;       Reference: Program Logic Description, Section 2.2.1
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ ;       Associated completion codes
                        C+ ;
= 00FF                  C+ $OPinp  EQU     0FFH            ;Operation in progress
= 0000                  C+ $OPcne  EQU     0               ;Completed without error
= 0001                  C+ $OPinl  EQU     1               ;Completed, incorrect length
= 0002                  C+ $OPtrb  EQU     2               ;Terminated, invalid ORB
= 0003                  C+ $OPtto  EQU     3               ;Terminated, receive time-out
= 0004                  C+ $OPtdf  EQU     4               ;Terminated, invalid data format
= 0005                  C+ $OPtng  EQU     5               ;Terminated, transmission failure
= 0006                  C+ $OPttf  EQU     6               ;Terminated, transporter failure
= 0007                  C+ $OPtcl  EQU     7               ;Terminated, control length mismatch
                        C+ ;
                        C+ ;       The structure itself
                        C+ ;
```

```
0000  FF                C+ ORB      STRUC
0001  00                C+ CC       DB      $OPinp          ;Completion Code
0002  0000              C+ OP       DB      0               ;Operational parameter
0004  0000              C+ Sze      DW      0               ;Number of bytes of data
0006  0000              C+ RB_Data  DW      0               ;Offset of data
0008                    C+ RB_Datas DW      0               ;Reserved for segment address
                        C+ ORB      ENDS
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ IF (DR$ OR TL$ OR LC$)
                        C+ ;--------------------------------------------------------------------------
                        C+ ;                    Transporter Usage Parameters                        |
                        C+ ;--------------------------------------------------------------------------
                        C+ ;
= 0100                  C+ $InitTA EQU      100H            ;Initial transporter CCB address
= 7FFF                  C+ $MaxDly EQU      7FFFH           ;Max loop count for operation complete
= 0010                  C+ $MaxRty EQU      10H             ;Maximum number operation retries
= 000A                  C+ $MaxWte EQU      10              ;Maximum wait time in seconds
= 01FE                  C+ $ProID  EQU      01FEH           ;protocol identifier
= 00CF                  C+ $SndNAD EQU      0CFH            ;Invalid command for NAD reply
= 0211                  C+ $MaxTBD EQU      529             ;Longest data transfer (w/o disk stat)
                        C+ ;
                        C+ ;--------------------------------------------------------------------------
                        C+ ;                 Transporter Socket Usage Conventions                   |
                        C+ ;--------------------------------------------------------------------------
                        C+ ;
= 00B0                  C+ $SokTD  EQU      0B0H            ;To transmit data
= 0080                  C+ $SokTM  EQU      80H             ;To transmit an NLM
= 0080                  C+ $SokRM  EQU      80H             ;To receive an NLM
= 00B0                  C+ $SokRC  EQU      0B0H            ;To receive initial command sequence
= 00A0                  C+ $SokRL  EQU      0A0H            ;To receive remainder of long command
                        C+ ;
                        C+ ;--------------------------------------------------------------------------
                        C+ ;       Structure Name:  Name Lookup Message                               |
                        C+ ;       Reference: Program Logic Description, Section 2.2.3                |
                        C+ ;--------------------------------------------------------------------------
                        C+ ;
                        C+ ;       Associated message type codes (most significant byte)
                        C+ ;
= 0000                  C+ $NLHi   EQU      0               ;Hello
= 0001                  C+ $NLAny  EQU      1               ;Any disk server (old protocol)
= 0002                  C+ $NLWho  EQU      2               ;Who are you?
= 0003                  C+ $NLWhr  EQU      3               ;Where are you?
= 0010                  C+ $NLIdn  EQU      10H             ;Identification
= 00FF                  C+ $NLBye  EQU      0FFH            ;Goodbye
                        C+ ;
                        C+ ;       Associated device types of interest to Emulator
                        C+ ;
= 0001                  C+ $NLDsk  EQU      1               ;Disk Server
= 00FF                  C+ $NLDny  EQU      0FFH            ;All Devices
                        C+ ;
                        C+ ;       Associated Name Length
                        C+ ;
= 000A                  C+ $NameLN EQU      10              ;Needed because STRUC is weak
```

```
                        C+ ;
                        C+ ;       The structure itself (arithmetic reverses byte order
                        C+ ;       for transporter memory storage)
                        C+ ;
                        C+ NLM     STRUC
0000  FE01              C+ Pid     DW      $ProID/256 + 256*($ProID AND 0FFH)
0002  0000              C+ Mgt     DW      $NLhi           ;Message Type
0004  0000              C+ Src     DW      0               ;Station Number of sender
0006  0100              C+ Dev     DW      256*$NLDsk      ;Device Type
0008  50 43 53 48 41 52 C+ NL_Name DB      'PCSHARE   '    ;Device or user name
      45 20 20 20       C+
0012                    C+ NLM     ENDS
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Receive ICS control area                       |
                        C+ ;       Reference: Corvus OMNINET Programmer' Guide                     |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ RRRCV   STRUC
0000  0000              C+ Lofcm   DW      0               ;length of the command being sent
0002  0000              C+ ELofRp  DW      0               ;Expected reply length (w/o status)
0004                    C+ RRRCV   ENDS
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Send Reply control area                        |
                        C+ ;       Reference: Corvus OMNINET Programmer' Guide                     |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ RRSND   STRUC
0000  0000              C+ ALofRp  DW      0               ;Actual length of reply (incl status)
0002  00                C+ Dstat   DB      0               ;Disk status
0003                    C+ RRSND   ENDS
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Transporter Status Table Header                |
                        C+ ;       Reference: Program Logic description, Section 2.3               |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ TSTH    STRUC
0000  00                C+ Int     DB      0               ;Interrupt depth indicator
0001  01                C+ TT_CI   DB      $False          ;Command Interpreter invocation flag
0002  00                C+ NE      DB      0               ;Number of entries in TST
0003  00                C+ TT_NAD  DB      0               ;Network address of Emulator
0004                    C+ TSTH    ENDS
                        C+ ;
                        C+ ;-------------------------------------------------------------------------
                        C+ ;       Structure Name:  Transporter Status Table Entry                 |
                        C+ ;       Reference: Program Logic Description, Section 2.3               |
                        C+ ;-------------------------------------------------------------------------
                        C+ ;
                        C+ TSTE    STRUC
0000  01                C+ PN      DB      $False          ;ORB pending flag
0001  00                C+ TT_ST   DB      0               ;Status code from result record
0002  00                C+ SK      DB      0               ;Socket to be used
0003  00                C+ TT_OP   DB      0               ;Transporter commmand  !NEW USAGE!
```

```
0004  0000                C+ CCB      DW      0               ;Transporter address of the CCB
0006  0000                C+ RR       DW      0               ;Transporter address of result record
0008  0000                C+ Dtl      DW      0               ;Transporter address of data area
000A  0000                C+ Dsz      DW      0               ;Length of data area in bytes
000C  0000                C+ Isr      DW      0               ;Offset address of int service routine
000E  0000                C+ Isrs     DW      0               ;  Reserved for segment address
0010  0000                C+ TT_Orb   DW      0               ;Offset address of pending ORB
0012  0000                C+ TT_Orbs  DW      0               ;  Reserved for segment address
0014                      C+ TSTE     ENDS
                          C+ ;
                          C+ ENDIF
                          C+ ;
                          C+ IF (TL$ or LC$)
                          C+ ;-------------------------------------------------------------------------
                          C+ ;       Structure Name:  Transporter Status Table Local Command entry  |
                          C+ ;       Reference: Program Logic Description, Section 8.3              |
                          C+ ;-------------------------------------------------------------------------
                          C+ ;
                          C+ ;       Associated driver communication flag values
                          C+ ;
=                         C+ $LCnop   EQU     $False          ;No operation indication
=                         C+ $LCopn   EQU     $True           ;Operation in progress indication
= 00FF                    C+ $LCabt   EQU     0FFH            ;'Operation aborted' end indication
= 0000                    C+ $LCaok   EQU     0               ;'All ok' end indication
                          C+ ;
                          C+ ;       Asociated data values
                          C+ ;
=                         C+ $LCmaxL  EQU     $MaxTBD         ;Longest data transfer allowed
                          C+ ;
                          C+ TSTLE    STRUC
0000  00                  C+ LE_CC    DB      0               ;RB.CC or RQ.USE field
0001  00                  C+ LE_OP    DB      0               ;RB.OP field when serving as ORB
0002  0000                C+ LE_Len   DW      0               ;Length in bytes of remaining data
0004  0000                C+ Trm      DW      0               ;Offset from which to fetch data
0006  0000                C+ Trms     DW      0               ;  Segment address of data
0008  0000                C+ Rcv      DW      0               ;Offset to send reply
000A  0000                C+ Rcvs     DW      0               ;  Segment address for reply
000C  0000                C+ LE_Isr   DW      0               ;Offset of interrupt service routine
000E  0000                C+ LE_Isrs  DW      0               ;  Resderved for segment address ISR
0010  0000                C+ LE_BP    DW      0               ;Stack segment offset of inout registers
0012  FFFF                C+          DW      0FFFFH          ;Reserved
0014                      C+ TSTLE    ENDS
                          C+ ;
                          C+ ;-------------------------------------------------------------------------
                          C+ ;       Structure Name:  Command Request Queue Entry                   |
                          C+ ;       Reference: Program Logic Specification, Section 2.5            |
                          C+ ;-------------------------------------------------------------------------
                          C+ ;
                          C+ ;       Associated usage values
                          C+ ;
= 0080                    C+ $RQUseE  EQU     80H             ;Entry in queue
= 0040                    C+ $RQUseR  EQU     40H             ;Request received for command data
= 0020                    C+ $RQUseL  EQU     20H             ;Local command
= 0010                    C+ $RQUseI  EQU     10H             ;Internally generated command
                          C+ ;
```

```
= 0004                          C+ $RQCsze EQU     4               ;Number of command bytes enqueued
                                C+ ;
                                C+ ;       The structure itself
                                C+ ;
                                C+ CRQE    STRUC
0000  00                        C+ Use     DB      0               ;Usage flags
0001  00                        C+ NAD     DB      0               ;Network Address of requestor
0002  0000                      C+ Len     DW      0               ;Length in bytes of expected reply
0004     04 [                   C+ Cmnd    DB      $RQCsze Dup(0)  ;First four command bytes
               00               C+
                  ]             C+
                                C+
0008                            C+ CRQE    ENDS
                                C+ ;
                                C+ ENDIF
                                C+ ;
                                C+ IF (DR$ OR TL$)
                                C+ ;-------------------------------------------------------------------------
                                C+ ;       Structure Name:  Command Request Queue Header                   |
                                C+ ;       Reference: Program Logic Description, Section 2.5               |
                                C+ ;-------------------------------------------------------------------------
                                C+ ;
                                C+ ;       Associated queue status indicator values
                                C+ ;
= 0000                          C+ $RQmt   EQU     0               ;Queue is empty
= 0001                          C+ $RQent  EQU     1               ;Queue has entries but is not full
= 00FF                          C+ $RQful  EQU     0FFH            ;Queue is full
                                C+ ;
= 0080                          C+ $RQdse  EQU     80H             ;Disk status error flag bit
                                C+ ;
                                C+ CRQH    STRUC
0000  00                        C+ Qst     DB      $RQmt           ;Queue status indicator
0001  00                        C+ Dsc     DB      0               ;Current disk status code
0002  0000                      C+ End     DW      0               ;Offset address following CRQ
0004  0000                      C+ Top     DW      0               ;Offset address of head entry
0006  0000                      C+ Bot     DW      0               ;Offset address of tail entry
0008                            C+ CRQH    ENDS
                                C+ ;
                                C+ ENDIF
                                C+ ;
                                C+ IF TL$
                                C+ ;-------------------------------------------------------------------------
                                C+ ;       Structure Name:  Name Lookup Message Table                      |
                                C+ ;       Reference: Program Logic Description, Section 2.7               |
                                C+ ;-------------------------------------------------------------------------
                                C+ ;
                                C+ ;       Associated command modifiers
                                C+ ;
= 0003                          C+ $MTAdd  EQU     3               ;AddActive command
= 0000                          C+ $MTDel  EQU     0               ;DeleteActive command
                                C+ ;
                                C+ NLMT    STRUC
0000  10                        C+ MT_Use  DB      $RQUseI         ;Specifies internal command
0001  00                        C+ MT_NAD  DB      0               ;NAD of sender  !NEW FIELD!
0002     02 [                   C+         DB      2 DUP(0FFH)     ;Unused part of parameter block
```

```
                FF              C+
                    ]           C+
                                C+
0004  34                       C+ Opc      DB      34H             ;Internal is always hex 34
0005  00                       C+ MT_Mod   DB      0               ;Will be set to command modifier
0006      0A [                 C+ MT_Name  DB      10 DUP(' ')     ;Device or user name
                20              C+
                    ]           C+
                                C+
0010  00                       C+ MT_Src   DB      0               ;Station number of sender
0011  00                       C+ MT_Dev   DB      0               ;Device type of sender
0012      04 [                 C+ Rsvd     DB      4 DUP(' ')      ;Four byte user area
                20              C+
                    ]           C+
                                C+
0016                           C+ NLMT     ENDS
                               C+ ;
                               C+ ENDIF
                               C+ ;
                               C+ IF (DR$ OR TL$ OR LC$)
                               C+ ;-----------------------------------------------------------------------
                               C+ ;            Transporter structures and associated values           |
                               C+ ;            Reference: Corvus OMNINET Programmer's Guide           |
                               C+ ;-----------------------------------------------------------------------
                               C+ ;
                               C+ ;        Transporter Command Control Block
                               C+ ;
                               C+ CCBLK    STRUC
0000  00                       C+ CC_Cmnd  DB      0               ;Command
0001  00                       C+          DB      0               ;fill for high address byte
0002  0000                     C+ RRA      DW      0               ;address Result Record
0004  00                       C+ Sock     DB      0               ;socket to use
0005  00                       C+          DB      0               ;fill for high address byte
0006  0000                     C+ CC_Data  DW      0               ;Data address
0008  0000                     C+ Datl     DW      0               ;Data length
000A  00                       C+ Conln    DB      0               ;Control length
000B  00                       C+ DHost    DB      0               ;Destination host (if write)
000C                           C+ CCBLK    ENDS
                               C+ ;
                               C+ ;        Transporter Result Record structure
                               C+ ;
                               C+ RRKD     STRUC
0000  FF                       C+ RCode    DB      0FFH            ;Return Code
0001  00                       C+ SHost    DB      0               ;Source Host (if read, 0 write)
0002  0000                     C+ Rcvlen   DW      0               ;Length data received (if read)
0004                           C+ RRKD     ENDS
                               C+ ;
                               C+ ;        Transporter port definitions
                               C+ ;
= 0249                         C+ $RdRAM   EQU     249H            ;To read without incrementing counter
= 024B                         C+ $RdRAMI  EQU     24BH            ;To read and increment counter
= 0248                         C+ $RdStat  EQU     248H            ;To read status
                               C+ ;
= 0248                         C+ $WrCtrH  EQU     248H            ;To set high byte of counter
= 024A                         C+ $WrCtrL  EQU     24AH            ;To set low byte of counter
```

```
= 024B                  C+ $WrRAM  EQU   24BH          ;To write data and increment counter
                        C+ ;
= 0249                  C+ $WrStbe EQU   249H          ;To strobe command address
                        C+ ;
= 024C                  C+ $DiINT  EQU   24CH          ;To disable (disarm) interrupts
= 024D                  C+ $ClINT  EQU   24DH          ;To clear interrupt latch
= 024E                  C+ $EnINT  EQU   24EH          ;To enable (arm) interrupts
= 024F                  C+ $StINT  EQU   24FH          ;To read interrupt status
                        C+ ;
                        C+ ;       Transporter Status Flags
                        C+ ;
= 0080                  C+ $TRedy  EQU   80H           ;Transporter ready
= 0020                  C+ $TRenb  EQU   20H           ;Transporter interrupts enabled
= 0010                  C+ $TRpnd  EQU   10H           ;Transporter interrupt pending
                        C+ ;
                        C+ ;       Transporter command codes
                        C+ ;
= 0040                  C+ $Send   EQU   40H           ;Send message
= 00F0                  C+ $Setup  EQU   0F0H          ;Setup to receive message
= 0010                  C+ $EndRcv EQU   10H           ;End receive if no message
= 0020                  C+ $InitT  EQU   20H           ;Initialize transporter
= 0001                  C+ $WhoAmI EQU   01H           ;Finds transporters NAD
= 0002                  C+ $Echo   EQU   02H           ;Test presence of specified transporter
                        C+ ;$PP    EQU   08H           ;Peek/Poke not used
                        C+ ;
                        C+ ;       Transporter result codes
                        C+ ;
= 0000                  C+ $TrOK   EQU   00H           ;Command completed successfully
= 007F                  C+ $TrST   EQU   7FH           ;Successful transmit retry maximum
= 0080                  C+ $TrNG   EQU   80H           ;Transmit failure, retry count exceeded
= 0081                  C+ $TrDL   EQU   81H           ;Data too long for receive buffer
= 0082                  C+ $TrNS   EQU   82H           ;Receiver's socket not set up
= 0083                  C+ $TrCL   EQU   83H           ;Control length mismatch
= 0084                  C+ $TrIS   EQU   84H           ;Invalid socket
= 0085                  C+ $TrNR   EQU   85H           ;receive socket in use
= 0086                  C+ $TrIT   EQU   86H           ;invalid transporter address
= 00C0                  C+ $TrEA   EQU   0C0H          ;Echo command acknowledged
= 00FE                  C+ $TrSS   EQU   0FEH          ;Receive socket setup successful
                        C+ ;
                        C+ ;       Transporter usage values
                        C+ ;
= 0040                  C+ $NADok  EQU   64            ;First invalid NAD value
= 00FF                  C+ $AllNAD EQU   0FFH          ;NAD for broadcast
= 00FF                  C+ $TrSU   EQU   0FFH          ;Initialization value for result code
                        C+ ;
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ ENDIF
                        C+ ;
                        C+ ENDIF
```

```
C+ ;
C+ ENDIF
C+ ENDIF
C+ ;
C+ ENDIF
C+
    ;
    ;*****************************************************************************
    ;                       Segment Usage Definition                               |
    ;*****************************************************************************
    ;
    PCS  GROUP  PC_CODE,PC_DATA
            ASSUME  CS:PCS,DS:PC_DATA,ES:PC_DATA
    ;
    ;*****************************************************************************
    ;                       External Symbol Definitions                            |
    ;*****************************************************************************
    ;
            EXTRN   EC:BYTE                 ;Emulator Communication Table
            EXTRN   MT:BYTE                 ;Name Lookup Message Table
            EXTRN   RQ:BYTE                 ;CRQ Header
            EXTRN   DM_ERR_LOG:NEAR         ;Error log routine
            EXTRN   EM_TRACE:NEAR           ;Trace routine
            EXTRN   LC_RCV_CMND:NEAR        ;Local Command receive
            EXTRN   TL_RCV_NLM:NEAR         ;Receive NLM routine
            EXTRN   TL_RCV_ICS:NEAR         ;Receive ICS routine
            EXTRN   LC_RTN_RSLT:NEAR        ;Local Command result
            EXTRN   LC_FETCH_LC:NEAR        ;Local Command fetch long
            EXTRN   TL_INT_NAD:NEAR         ;Int svc rtn for send NAD
            EXTRN   TL_FETCH_IC:NEAR        ;Fetch internal command
            EXTRN   LC_INT_NLM:NEAR         ;Local Command solicited NLM
            EXTRN   TL_INT_WAIT:NEAR        ;Wait for Operation Complete
    ;
    ;*****************************************************************************
    ;                             Allocated Data                                   |
    ;*****************************************************************************
    ;
0000        PC_DATA SEGMENT PUBLIC 'DATA'
    ;
    ;-----------------------------------------------------------------------------
    ;
    ;       Transporter Data Area lengths
    ;
= 0012      $DtaNL  EQU     TYPE NLM        ;Name Lookup Message
= 0004      $DtaIS  EQU     4               ;Initial Command Sequence
=           $DtaLC  EQU     $MaxTBD         ;Remainder of long command
=           $DtaRP  EQU     $MaxTBD         ;Reply to any command
= 0002      $DtaGO  EQU     2               ;Long command GO message
= 0000      $DtaND  EQU     0               ;Return NAD message
    ;
    ;       Transporter Control area lengths
    ;
= 0004      $Ctl1S  EQU     TYPE RRRCV      ;Initial Command Sequence
= 0003      $Ctl1P  EQU     TYPE RRSND      ;All replies
= 0000      $Ctl1O  EQU     0               ;Long command GO message
```

```
                                ;
                                ;       Transporter Status Table
                                ;
                                        EYE     <TSTH>
0000  54 53 54 48             +         DB      "TSTH"
                                ;
0004  00                        TT      TSTH    (,$False,((OFFSET TTEND - OFFSET TTbgn) / (TYPE TSTE)))
0005  01
0006  08
0007  00

                                ;
= 0008                          TTbgn   EQU     THIS BYTE       ;Start non-local entries
                                ;
= 0100                          $TACCB  =       $InitTA                 ;Initial transporter
= 010C                          $TARR   =       $TACCB + TYPE CCBLK     ;  addresses for CCB,
= 0110                          $TADtl  =       $TARR + TYPE RRKD       ;  RR, and data
                                ;
0008  01                        RcvNLM  TSTE    (,,$SokRM,$Setup,$TACCB,$TARR,$TADtl,$DtaNL,TL_RCV_NLM)
0009  00
000A  80
000B  F0
000C  0100
000E  010C
0010  0110
0012  0012
0014  0000 E
0016  0000
0018  0000
001A  0000

                                ;
= 0122                          $TACCB  =       $TADtl + $DtaNL         ;Addresses for
= 012E                          $TARR   =       $TACCB + TYPE CCBLK     ;  RcvICS group
= 0136                          $TADtl  =       $TARR + TYPE RRKD + $CtlIS
                                ;
001C  01                        RcvICS  TSTE    (,,$SokRC,$Setup,$TACCB,$TARR,$TADtl,$DtaIS,TL_RCV_ICS)
001D  00
001E  B0
001F  F0
0020  0122
0022  012E
0024  0136
0026  0004
0028  0000 E
002A  0000
002C  0000
002E  0000

                                ;
= 013A                          $TACCB  =       $TADtl + $DtaIS         ;Addresses for
= 0146                          $TARR   =       $TACCB + TYPE CCBLK     ;  RcvGO group
= 014A                          $TADtl  =       $TARR + TYPE RRKD
                                ;
0030  01                        RcvGO   TSTE    (,,$SokRL,$Setup,$TACCB,$TARR,$TADtl,$DtaLC,TL_INT_LCR)
```

```
0031  00
0032  A0
0033  F0
0034  013A
0036  0146
0038  014A
003A  0211
003C  0435 R
003E  0000
0040  0000
0042  0000

                                ;
= 035B                          $TACCB  =       $TADtl + $DtaLC         ;Addresses for
= 0367                          $TARR   =       $TACCB + TYPE CCBLK     ;  TrmNLM group
= 036B                          $TADtl  =       $TARR + TYPE RRKD
                                ;
0044  01                        TrmNLM  TSTE    (,,$SokTM,$Send,$TACCB,$TARR,$TADtl,$DtaNL,TL_INT_NLM)
0045  00
0046  80
0047  40
0048  035B
004A  0367
004C  036B
004E  0012
0050  056F R
0052  0000
0054  0000
0056  0000

                                ;
= 037D                          $TACCB  =       $TADtl + $DtaNL         ;Addresses for
= 0389                          $TARR   =       $TACCB + TYPE CCBLK     ;  TrmNAD group
= 0390                          $TADtl  =       $TARR + TYPE RRKD + $CtlRP
                                ;
0058  01                        TrmNAD  TSTE    (,,$SokTD,$Send,$TACCB,$TARR,$TADtl,$DtaND,TL_INT_NAD)
0059  00
005A  B0
005B  40
005C  037D
005E  0389
0060  0390
0062  0000
0064  0000 E
0066  0000
0068  0000
006A  0000

                                ;
= 0390                          $TACCB  =       $TADtl + $DtaND         ;Addresses for
= 039C                          $TARR   =       $TACCB + TYPE CCBLK     ;  TrmRPY group
= 03A3                          $TADtl  =       $TARR + TYPE RRKD + $CtlRP
                                ;
006C  01                        TrmRPY  TSTE    (,,$SokTD,$Send,$TACCB,$TARR,$TADtl,$DtaRP,TL_INT_RSLT)
006D  00
```

```
006E  B0
006F  40
0070  0390
0072  039C
0074  03A3
0076  0211
0078  034D R
007A  0000
007C  0000
007E  0000

                                ;
= 05B4                          $TACCB  =       $TADtl + $DtaRP         ;Addresses for
= 05C0                          $TARR   =       $TACCB + TYPE CCBLK     ;  TrmGO group
= 05C4                          $TADtl  =       $TARR + TYPE RRKD + $CtlGO
                                ;
0080  01                        TrmGO   TSTE    (,,$SokTD,$Send,$TACCB,$TARR,$TADtl,$DtaGO,TL_INT_LCT)
0081  00
0082  B0
0083  40
0084  05B4
0086  05C0
0088  05C4
008A  0002
008C  04C4 R
008E  0000
0090  0000
0092  0000

                                ;
= 05C6                          $TACCB  =       $TADtl + $DtaGO         ;Addresses for
= 05D2                          $TARR   =       $TACCB + TYPE CCBLK     ;  VarOP group
= 05D6                          $TADtl  =       $TARR + TYPE RRKD
                                ;
0094  01                        VarOP   TSTE    (,,,$initT,$TACCB,$TARR,$TADtl,,TL_INT_TIME)
0095  00
0096  00
0097  20
0098  05C6
009A  05D2
009C  05D6
009E  0000
00A0  01EE R
00A2  0000
00A4  0000
00A6  0000

                                ;
= 00A8                          TTend   EQU     THIS BYTE       ;End non-local entries
                                ;
                                ;--------------------------------------------------------------------------
                                ;
00A8                            PC_DATA ENDS
                                ;
                                ;****************************************************************************
```

```
                        ;                Transporter Logical Device Routines
                        ;*****************************************************************
                        ;
0000                    PC_CODE SEGMENT PUBLIC 'CODE'
                        ;
                        ;*****************************************************************
                        ;Module name:  Transporter Interface Routines
                        ;
                        ;Version:  2.0
                        ;
                        ;Last Update: 15 December 1983
                        ;
                        ;Function:  The Transporter Interface routines provide services that
                        ;           insert data into transporter memory, extract data from it,
                        ;           and initiate commands.  There are eight routines:
                        ;
                        ;           TL-SET-CTR sets the transporter location counter
                        ;           TL-BYT-OUT, TL-WRD-OUT, TL-DTA-OUT put one byte, one word,
                        ;             or a string of bytes into transporter memory
                        ;           TL-BYT-IN, TL-WRD-IN, TL-DTA-IN fetch one byte, one word,
                        ;             or a sring of bytes from transporter memory
                        ;           TL-DO-CMND causes a transporter command to be executed
                        ;
                        ;Procedure:  See Program Logic Description, Section 2.3
                        ;
                        ;Called by:  TLD service routines
                        ;
                        ;Routines called:  None
                        ;
                        ;Input:  See individual routine descriptions
                        ;
                        ;Output:  See individual routine descriptions
                        ;
                        ;Error procedures:  None
                        ;
                        ;Written by: R.B. Talmadge, Computer Technology Ltd
                        ;
                        ;Updates:  None
                        ;
                        ;*****************************************************************
                        ;
                        ;---------------------------------------------------------------
                        ;=======TL-SET-CTR: Sets transporter address counter to value in
                        ;                  register AX.  Registers AX, DX not preserved.
                        ;
0000                    TL_SET_CTR  PROC  NEAR
0000  BA 024A                   MOV     DX,$WrCtrL      ;Port for low byte
0003  EE                       OUT     DX,AL           ;  which goes out first
0004  BA 0248                   MOV     DX,$WrCtrH      ;Port for high byte
0007  86 E0                     XCHG    AH,AL           ;  which goes
0009  EE                       OUT     DX,AL           ;  out next
000A  C3                       RET
000B                    TL_SET_CTR  ENDP
                        ;
                        ;---------------------------------------------------------------
```

```
                        ;=======TL-BYT-OUT:  Outputs one byte to a specified location
                        ;                   in transporter memory
                        ;
                        ;               Input: BX=Transporter memory location
                        ;                      CL=Byte to output
                        ;
                        ;               Output: None
                        ;
                        ;               Registers AX, DX not preserved
                        ;
000B                    TL_BYT_OUT  PROC  NEAR
                                STATUS                  ;Disable interrupts
000B  9C              +         PUSHF
000C  FA              +         CLI
000D  8B C3                     MOV     AX,BX           ;Set up transporter
000F  E8 0000 R                 CALL    TL_SET_CTR      ;  location counter
0012  BA 024B                   MOV     DX,$WrRAM       ;Port for  output
0015  8A C1                     MOV     AL,CL           ;  and out
0017  EE                        OUT     DX,AL           ;  it goes
                                RESTORE                 ;Restore interrupt status
0018  9D              +         POPF
0019  C3                       RET
001A                    TL_BYT_OUT  ENDP
                        ;
                        ;-----------------------------------------------------------------
                        ;=======TL-WRD-OUT:  Outputs one word to a specified location
                        ;                   in transporter memory
                        ;
                        ;               Input: BX=transporter memory location
                        ;                      CX=Word to output
                        ;
                        ;               Register AX, DX not preserved
                        ;
                        ;               Reentry at TL-OUTa before any other call to
                        ;               a device routine will output another word from
                        ;               CX to transporter memory at next location
                        ;
001A                    TL_WRD_OUT  PROC  NEAR
                                STATUS                  ;Disable interrupts
001A  9C              +         PUSHF
001B  FA              +         CLI
001C  8B C3                     MOV     AX,BX           ;Set up transporter
001E  E8 0000 R                 CALL    TL_SET_CTR      ;  location counter
0021  BA 024B                   MOV     DX,$WrRAM       ;Port for output
                        ;
0024  8A C5             TL_OUTa: MOV    AL,CH           ;Output high
0026  EE                        OUT     DX,AL           ;  order byte
0027  8A C1                     MOV     AL,CL
0029  EE                        OUT     DX,AL           ;  then low order
                                RESTORE                 ;Restore interrupt status
002A  9D              +         POPF
002B  C3                       RET
002C                    TL_WRD_OUT  ENDP
                        ;
                        ;-----------------------------------------------------------------
```

```
                        ;=======TL_BYT_IN:  Inputs one byte from a specified location
                        ;                  in transporter memory
                        ;
                        ;               Input:  BX=Transporter memory location
                        ;
                        ;               Output: AL=Byte returned
                        ;
                        ;               Register DX not preserved
                        ;
002C                    TL_BYT_IN  PROC  NEAR
                                STATUS                  ;Disable interrupts
002C  9C              +         PUSHF
002D  FA              +         CLI
002E  8B C3                     MOV     AX,BX           ;Set up transporter
0030  E8 0000 R                 CALL    TL_SET_CTR      ;  location counter
0033  BA 024B                   MOV     DX,$RdRAMI      ;Port to read from (it increments)
0036  EC                        IN      AL,DX           ;Get the byte
                                RESTORE                 ;Restore interrupt status
0037  9D              +         POPF
0038  C3                        RET                     ;  and exit
0039                    TL_BYT_IN  ENDP
                        ;
                        ;-------------------------------------------------------------------------
                        ;=======TL-WRD-IN:  Inputs one word from a specified location in
                        ;                  transporter memory
                        ;
                        ;               Same as TL-BYT-IN except AX=Word returned
                        ;
0039                    TL_WRD_IN  PROC  NEAR
                                STATUS                  ;Disable interrupts
0039  9C              +         PUSHF
003A  FA              +         CLI
003B  E8 002C R                 CALL    TL_BYT_IN       ;Get high order byte
003E  8A E0                     MOV     AH,AL           ;  and set it high
0040  EC                        IN      AL,DX           ;Now low order
                                RESTORE                 ;Restore interrupt status
0041  9D              +         POPF
0042  C3                        RET
0043                    TL_WRD_IN  ENDP
                        ;
                        ;-------------------------------------------------------------------------
                        ;=======TL-DO-CMND: Initiates transporter command and waits for result
                        ;                   code to change indicating command completion.
                        ;
                        ;               Input: BX=Transporter adddress of result code
                        ;                      CX=Transporter address of CCB
                        ;
                        ;               Returns: AL=Result code value
                        ;                        AH=0
                        ;
                        ;               Registers CX, DX not preserved.
                        ;
0043                    TL_DO_CMND  PROC  NEAR
                                STATUS                  ;Disable interrrupts
0043  9C              +         PUSHF
```

```
0044  FA                    +         CLI
                            ;
0045  8B C3                          MOV     AX,BX           ;Set counter to point to
0047  E8 0000 R                      CALL    TL_SET_CTR      ;  result code
004A  B0 FF                          MOV     AL,$TrSU        ;Initialize result
004C  BA 024B                        MOV     DX,$WrRAM       ;  code value
004F  EE                             OUT     DX,AL
                            ;
0050  32 E4                          XOR     AH,AH           ;High byte of command location
0052  E8 0076 R                      CALL    StrbC           ;  is always zero for this board
0055  8A E5                          MOV     AH,CH           ;Other bytes are
0057  E8 0076 R                      CALL    StrbC           ;  then output to
005A  8A E1                          MOV     AH,CL           ;  start the operation
005C  E8 0076 R                      CALL    StrbC
                            ;
005F  8B C3                          MOV     AX,BX           ;Now set counter back to
0061  E8 0000 R                      CALL    TL_SET_CTR      ;  point to result code
0064  BA 0249                        MOV     DX,$RdRAM       ;  and setup for read
0067  33 C0                          XOR     AX,AX
0069  B9 7FFF                        MOV     CX,$MaxDly      ;Don't wait forever
                            ;
006C  EC                    DoRes:    IN      AL,DX           ;get result (no ctr increment)
006D  3C FF                          CMP     AL,$TrSU        ;Keep testing until it
006F  75 02                          JNE     DoRes1          ;  changes from initial value
0071  E1 F9                          LOOPE   DoRes           ;  or time runs out
                            ;
0073  EC                    DoRes1:   IN      AL,DX           ;Then read again to be sure
                                      RESTORE                 ;Restore interrupt status
0074  9D                    +         POPF
0075  C3                              RET                     ;  and exit
                              ;
                              ;               Subroutine internal to TL_DO_CMND which outputs
                              ;               command byte when transporter is ready to receive
                              ;
0076  BA 0248               StrbC:   MOV     DX,$RdStat      ;Port to read ready status
0079  EC                    StrbCr:  IN      AL,DX           ;Get status byte and
007A  A8 80                           TEST    AL,$TRedy       ;  test flag until
007C  74 FB                           JZ      StrbCr          ;  transporter ready
007E  8A C4                           MOV     AL,AH           ;Then output
0080  BA 0249                         MOV     DX,$WrStbe      ;  the command byte
0083  EE                              OUT     DX,AL
0084  C3                              RET
                              ;
0085                        TL_DO_CMND  ENDP
                              ;
                              ;-------------------------------------------------------------------------
                              ;=======TL-DTA-OUT:  Outputs data from memory to transporter board       |
                              ;                                                                       |
                              ;               Input:  DS:SI = memory location of first byte           |
                              ;                       CX = Number of bytes                            |
                              ;                       BX = Transporter address for data               |
                              ;                                                                       |
                              ;               Returns: None                                           |
                              ;                                                                       |
                              ;               Registers AX, SI, CX, DX not preserved                  |
```

```
                                ;
                                TL_DTA_OUT  PROC  NEAR
0085                                    JCXZ    DtaO            ;Just exit if count is zero
0085  E3 10                             STATUS                  ;Disable interrupts
                              +         PUSHF
0087  9C                     +         CLI
0088  FA                              ;
                                        MOV     AX,BX           ;Setup transporter
0089  8B C3                            CALL    TL_SET_CTR      ;  location counter
008B  E8 0000 R                         MOV     DX,$WrRAM       ;Port for data output
008E  BA 024B                           CLD                     ;Data addresses increment
0091  FC                              ;
                                DtaOUa: LODSB                   ;Get current byte
0092  AC                                OUT     DX,AL           ;Store it and go
0093  EE                                LOOP    DtaOUa          ;  back to get next
0094  E2 FC                           ;
                                        RESTORE                 ;Restore interrupt status
                              +         POPF
0096  9D                       DtaO:  RET                     ;  and exit when no more
0097  C3                              TL_DTA_OUT  ENDP
0098                                  ;
                                      ;-----------------------------------------------------------------------
                                      ;=======TL-DTA-IN:  Inputs data from transporter board to memory
                                      ;
                                      ;               Input:  ES:DI = Memory location of first byte
                                      ;                       CX = Number of bytes
                                      ;                       BX = Transporter address of data
                                      ;
                                      ;               Returns: None
                                      ;
                                      ;               Registers AX, DI, CX, DX not preserved
                                      ;
                                TL_DTA_IN  PROC  NEAR
0098                                    JCXZ    DtaI            ;Just exit if count is zero
0098  E3 10                             STATUS                  ;Disable interrupts
                              +         PUSHF
009A  9C                      +         CLI
009B  FA                              ;
                                        MOV     AX,BX           ;Setup transporter
009C  8B C3                             CALL    TL_SET_CTR      ;  location counter
009E  E8 0000 R                         MOV     DX,$RdRAMI      ;Port to read and increment trans addr
00A1  BA 024B                           CLD                     ;Memory addresses also increment
00A4  FC                              ;
                                DtaINa: IN      AL,DX           ;Get current byte
00A5  EC                                STOSB                   ;Stash into memory
00A6  AA                                LOOP    DTaINa          ;  and go back for next
00A7  E2 FC                             ;
                                        RESTORE                 ;Restore interrupt status
                              +         POPF
00A9  9D                       DtaI:  RET                     ;  and exit when no more
00AA  C3                              TL_DTA_IN  ENDP
00AB                                  ;
                                      ;***********************************************************************
                                      ;Module name:  Setup To Receive (TL-SET-RECV)
                                      ;
```

```
                        ;Version:  2.0
                        ;
                        ;Last Update: 5 January 1984
                        ;
                        ;Function:  The Setup To Receive routine is called by any TLD routine
                        ;           to prepare to receive a command or NL message.  The call
                        ;           parameters are the offset of the TST entry and an ORB
                        ;           location if a response is solicited.  If there is an ORB
                        ;           its location is recorded in the TST entry; if there is no
                        ;           ORB, a maximum size retry count is recorded instead of the
                        ;           ORB location.
                        ;
                        ;           If transporter memory shows a previous setup has not been
                        ;           cancelled, no action is taken and an invalid indication is
                        ;           returned.  The invalid indication is also returned if the
                        ;           command cannot be executed.
                        ;
                        ;           The TL-SET-RTRY entry point is called by any routine to
                        ;           request a retry of command execution.  It is also used to
                        ;           retry a transmission.  The TL-SET-TOPN entry does the
                        ;           actual work of executing the command, and is also use by
                        ;           TL-SET-SEND to carry out the message transmission
                        ;
                        ;Procedure:  Program Logic Description, Section 2.6
                        ;
                        ;Called by:  TL-DQ-CRQ, TL-INIT-TLD, TL-RCV-ICS,
                        ;           TL-FETCH-LC, TL-RTN-RSLT, TL-SEND-NLM,
                        ;           LC-RECV-SRVR, TL-SET-SEND
                        ;
                        ;Routines called:  TL-DO-CMND
                        ;
                        ;Input:      (SI) = Offset TST entry
                        ;            (AL) = $True if ORB present
                        ;       (DS):(BX) = Location of the ORB
                        ;
                        ;Output:     (AL) = $True if successful
                        ;                   $False if not successful or invalid
                        ;            (AL) = Return code from transporter
                        ;
                        ;Error procedures:  None
                        ;
                        ;Written by: R.B. Talmadge, Computer Technology Ltd
                        ;
                        ;Updates:  1.  Retry entry TL-SET-RTRY added 22 Dec 1983
                        ;
                        ;*****************************************************************************
                        ;
00AB                    TL_SET_RECV  PROC  NEAR
                        ;
                                TRACE   S:TL-SET-RECV
00AB  80 3E 000B E 00   +       CMP BYTE PTR EC.Trce,$True
00B0  75 0F             +       JNE     ??0000
00B2  50                +       PUSH    AX
00B3  B0 00             +       MOV     AL,0
00B5  56                +       PUSH    SI
```

```
00B6  BE 00A8 R                +          MOV     SI,OFFSET ??0001
00B9  E8 0000 E                +          CALL    EM_TRACE
00BC  5E                       +          POP     SI
00BD  58                       +          POP     AX
00BE  EB 01 90                 +          JMP NEAR PTR ??0000
00C1                           + PC_CODE ENDS
00A8                           + PC_DATA SEGMENT PUBLIC 'DATA'
00A8  00                       + ??0001  LRB     (,OFFSET ??0003 - OFFSET ??0002)
00A9  0D                       +
00AA  53 3A 54 4C 2D 53        + ??0002  DB      "S:TL-SET-RECV"
00B7                           + PC_DATA ENDS
00C1                           + PC_CODE SEGMENT PUBLIC 'CODE'
00C1                           + ??0000  LABEL   NEAR
                               ;
00C1  8A 64 01                         MOV     AH,[SI].TT_ST   ;Pick up current result
00C4  80 FC FF                         CMP     AH,$TrSU        ;Go to exit if previous
00C7  74 05                            JE      SETxst          ;  setup not yet
00C9  80 FC FE                         CMP     AH,$TrSS        ;  cancelled and
00CC  75 20                            JNE     TL_SET_TOPN     ;  no message received
                               ;
00CE  B0 01                    SETxst: MOV     AL,$False         ;Assume condition invalid
00D0  80 3C 00                         CMP     [SI].PN,$True     ;It's OK, however,
00D3  74 02                            JE      SETxsu            ;  if no ORB in use
00D5  B0 00                            MOV     AL,$True
                               ;
00D7                           SETxsu  LABEL   NEAR
                               ;
                                       TRACE   X:TL-SET-RECV
00D7  80 3E 000B E 00          +       CMP BYTE PTR EC.Trce,$True
00DC  75 0F                    +       JNE     ??0004
00DE  50                       +       PUSH    AX
00DF  B0 00                    +       MOV     AL,0
00E1  56                       +       PUSH    SI
00E2  BE 00B7 R                +       MOV     SI,OFFSET ??0005
00E5  E8 0000 E                +       CALL    EM_TRACE
00E8  5E                       +       POP     SI
00E9  58                       +       POP     AX
00EA  EB 01 90                 +       JMP NEAR PTR ??0004
00ED                           + PC_CODE ENDS
00B7                           + PC_DATA SEGMENT PUBLIC 'DATA'
00B7  00                       + ??0005  LRB     (,OFFSET ??0007 - OFFSET ??0006)
00B8  0D                       +
00B9  58 3A 54 4C 2D 53        + ??0006  DB      "X:TL-SET-RECV"
00C6                           + PC_DATA ENDS
00ED                           + PC_CODE SEGMENT PUBLIC 'CODE'
00ED                           + ??0004  LABEL   NEAR
                               ;
00ED  C3                               RET                           ;Return to caller
                               ;
                               ;-------------------------------------------------------------------------
                               ; This is both the reentry point for TL-SET-RECV when a setup is to be  |
                               ; carried out, and the transfer point which carries out the actual work|
                               ; of the TL-SET-SEND routine.                                           |
                               ;-------------------------------------------------------------------------
                               ;
```

```
00EE                            TL_SET_TOPN  LABEL  NEAR
                                ;
00EE  88 04                             MOV     [SI].PN,AL                      ;Capture ORB indicator
00F0  C6 44 10 10                       MOV     BYTE PTR [SI].TT_Orb,$MaxRty    ;Assume no ORB
                                ;
00F4  3C 00                             CMP     AL,$True                ;Skip ORB setup
00F6  75 06                             JNE     TL_SET_RTRY             ;  if not present
00F8  89 5C 10                          MOV     [SI].TT_Orb,BX          ;Save full address
00FB  8C 5C 12                          MOV     [SI].TT_Orbs,DS         ; if it is present
                                ;
                                ;-----------------------------------------------------------------------
                                ; This is both the reentry point for TL-SET-TOPN and the subroutine    |
                                ; called to retry either a Set Receive or Set Send command.           |
                                ;-----------------------------------------------------------------------
                                ;
00FE                            TL_SET_RTRY  LABEL  NEAR

00FE  53                                PUSH    BX              ;Save registers used
00FF  51                                PUSH    CX              ;  by transporter routine
0100  52                                PUSH    DX
                                ;
0101  8B 5C 06                          MOV     BX,[SI].RR        ;Fetch transporter addresses
0104  8B 4C 04                          MOV     CX,[SI].CCB       ;  result record and CCB,
0107  C6 44 01 FF                       MOV     [SI].TT_ST,$TrSU  ;  record passage in TST
010B  E8 0043 R                         CALL    TL_DO_CMND        ;Execute the command
                                ;
010E  8A E0                             MOV     AH,AL             ;Save result code
0110  B0 00                             MOV     AL,$True          ;Assume command executed
0112  80 FC FF                          CMP     AH,$TrSU          ;Reset return
0115  75 02                             JNE     SNxit             ;  code if not
0117  B0 01                             MOV     AL,$False
                                ;
0119  5A                        SNxit:  POP     DX                ;Restore registers
011A  59                                POP     CX                ;  and exit
011B  5B                                POP     BX
                                ;
                                        TRACE   E/E+S:TL-SET-XXXX
011C  80 3E 000B E 00           +       CMP BYTE PTR EC.Trce,$True
0121  75 0F                     +       JNE     ??0008
0123  50                        +       PUSH    AX
0124  B0 00                     +       MOV     AL,0
0126  56                        +       PUSH    SI
0127  BE 00C6 R                 +       MOV     SI,OFFSET ??0009
012A  E8 0000 E                 +       CALL    EM_TRACE
012D  5E                        +       POP     SI
012E  58                        +       POP     AX
012F  EB 01 90                  +       JMP NEAR PTR ??0008
0132                            + PC_CODE ENDS
00C6                            + PC_DATA SEGMENT PUBLIC 'DATA'
00C6  00                        + ??0009  LRB     <,OFFSET ??000B - OFFSET ??000A>
00C7  11                        +
00C8  45 2F 45 2B 53 3A         + ??000A  DB      "E/E+S:TL-SET-XXXX"
00D9                            + PC_DATA ENDS
0132                            + PC_CODE SEGMENT PUBLIC 'CODE'
0132                            + ??0008  LABEL   NEAR
```

```
0132  C3                        ;
                                        RET
                                ;
0133                            TL_SET_RECV  ENDP
                                ;
                                ;*****************************************************************************
                                ;Module name:  Send Message (TL-SET-SEND)                                     |
                                ;                                                                            |
                                ;Version:  2.0                                                               |
                                ;                                                                            |
                                ;Last Update: 22 December 1983                                               |
                                ;                                                                            |
                                ;Function:  The Send Message routine is called by any TLD routine to         |
                                ;           send a message of any kind to another machine.  The call         |
                                ;           parameters are the offset of the TST entry and an ORB            |
                                ;           location if a wait for completion is required.  If there         |
                                ;           is an ORB its location is recorded in the TST entry; if          |
                                ;           there is no ORB, a maximum size retry count is recorded          |
                                ;           instead of the ORB location.                                     |
                                ;                                                                            |
                                ;           The actual work of the routine is done by the TL-SET-TOPN        |
                                ;           execution sequence of TL-SET-SEND.                               |
                                ;                                                                            |
                                ;Procedure:  Program Logic Description, Section 2.6.1                        |
                                ;                                                                            |
                                ;Called by:  TL-DQ-CRQ, TL-INIT-TLD, TL-RCV-ICS,                             |
                                ;            TL-FETCH-LC, TL-RTN-RSLT, TL-SEND-NLM,                          |
                                ;            LC-RECV-SRVR                                                    |
                                ;                                                                            |
                                ;Routines called:  TL-DO-CMND                                                |
                                ;                                                                            |
                                ;Input:       (SI) = Offset TST entry                                         |
                                ;             (AL) = $True if ORB present                                     |
                                ;       (DS):(BX) = Location of the ORB                                       |
                                ;                                                                            |
                                ;Output:      (AL) = $True if successful                                      |
                                ;                    $False if not successful                                |
                                ;             (AH) = Return code from transporter                            |
                                ;                                                                            |
                                ;Error procedures:  None                                                     |
                                ;                                                                            |
                                ;Written by: R.B. Talmadge, Computer Technology Ltd                          |
                                ;                                                                            |
                                ;Updates:  None.                                                             |
                                ;*****************************************************************************
                                ;
0133                            TL_SET_SEND  PROC  NEAR
                                ;
                                        TRACE   S:TL-SET-SEND
0133  80 3E 000B E 00          +         CMP BYTE PTR EC.Trce,$True
0138  75 0F                   +         JNE     ??000C
013A  50                      +         PUSH    AX
013B  B0 00                   +         MOV     AL,0
013D  56                      +         PUSH    SI
```

```
013E  BE 00D9 R               +          MOV     SI,OFFSET ??000D
0141  E8 0000 E               +          CALL    EM_TRACE
0144  5E                     +          POP     SI
0145  58                     +          POP     AX
0146  EB 01 90               +          JMP NEAR PTR ??000C
0149                          + PC_CODE ENDS
00D9                          + PC_DATA SEGMENT PUBLIC 'DATA'
00D9  00                      + ??000D  LRB     (,OFFSET ??000F - OFFSET ??000E)
00DA  0D                      +
00DB  53 3A 54 4C 2D 53       + ??000E  DB      "S:TL-SET-SEND"
00E8                          + PC_DATA ENDS
0149                          + PC_CODE SEGMENT PUBLIC 'CODE'
0149                          + ??000C  LABEL   NEAR
                                ;
0149  EB A3                             JMP     TL_SET_TOPN             ;That was easy
                                ;
014B                            TL_SET_SEND  ENDP
                                ;
                                ;*****************************************************************************
                                ;Module name:  Timeout Setup (TL-SET-TIME)                                    |
                                ;                                                                             |
                                ;Version:  2.0                                                                |
                                ;                                                                             |
                                ;Last Update: 5 January 1984                                                  |
                                ;                                                                             |
                                ;Function:  The Timeout Setup routine is called by any TLD routine to         |
                                ;           set up a timing loop for receipt of a message.  The call          |
                                ;           parameters are the offset of a TST entry already setup to         |
                                ;           receive, the location of an ORB, and a network address.           |
                                ;           The expiration time is calculated from the value in field         |
                                ;           RB.OP of the ORB.  An Echo command timing loop is started         |
                                ;           with interrupt service routine set to TL-INT-TIME, which          |
                                ;           is part of TL-SET-TIME.  That routine ends the timing             |
                                ;           loop if a message is received, or if time-out occurs.             |
                                ;                                                                             |
                                ;Procedure:  Program Logic Description, Section 2.6.1                         |
                                ;                                                                             |
                                ;Called by:  TL-FETCH-LC                                                      |
                                ;                                                                             |
                                ;Routines called:  TL-DO-CMND, TL-BYT-OUT, DM-ERR-LOG                         |
                                ;                                                                             |
                                ;Input:       (SI) = Offset TST entry                                         |
                                ;             (AL) = NAD of machine expected to send message                  |
                                ;       (DS):(BX) = Location of the ORB                                       |
                                ;                                                                             |
                                ;Output:     None                                                             |
                                ;                                                                             |
                                ;Error procedures:  An error message is logged if a timing loop is            |
                                ;                   already in progress when the routine is called            |
                                ;                                                                             |
                                ;Written by: R.B. Talmadge, Computer Technology Ltd                           |
                                ;                                                                             |
                                ;Updates:  None                                                               |
                                ;                                                                             |
                                ;*****************************************************************************
```

```
                                ;
014B                            TL_SET_TIME  PROC  NEAR
                                ;
                                        TRACE   S:TL-SET-TIME
014B  80 3E 000B E 00         +         CMP BYTE PTR EC.Trce,$True
0150  75 0F                   +         JNE     ??0010
0152  50                      +         PUSH    AX
0153  B0 00                   +         MOV     AL,0
0155  56                      +         PUSH    SI
0156  BE 00E8 R               +         MOV     SI,OFFSET ??0011
0159  E8 0000 E               +         CALL    EM_TRACE
015C  5E                      +         POP     SI
015D  58                      +         POP     AX
015E  EB 01 90                +         JMP NEAR PTR ??0010
0161                          + PC_CODE ENDS
00E8                          + PC_DATA SEGMENT PUBLIC 'DATA'
00E8  00                      + ??0011  LRB     (,OFFSET ??0013 - OFFSET ??0012)
00E9  0D                      +
00EA  53 3A 54 4C 2D 53       + ??0012  DB      "S:TL-SET-TIME"
00F7                          + PC_DATA ENDS
0161                          + PC_CODE SEGMENT PUBLIC 'CODE'
0161                          + ??0010  LABEL   NEAR
                                ;
0161  80 3C 00                          CMP     [SI].PN,$True   ;Just exit if the
0164  74 01                             JE      TMEst           ;  message has already
0166  C3                                RET                     ;  been received
                                ;
0167  80 3E 0094 R 00           TMEst:  CMP     VarOP.PN,$True  ;If a timing loop is
016C  75 0E                             JNE     TMEok           ;  already in progress
016E  C6 04 01                          MOV     [SI].PN,$False  ;  treat the request
0171  C6 07 03                          MOV     [BX].CC,$OPtto  ;  as if it timed out
                                ;
                                        LOG     $MND03          ;Then log an error
0174  50                      +         PUSH    AX
0175  B0 0B                   +         MOV     AL,$MND03
0177  E8 0000 E               +         CALL    DM_ERR_LOG
017A  58                      +         POP     AX
017B  C3                                RET                     ;  and return to the caller
                                ;
                                ;       If the loop can be started, store the current time in field
                                ;       TT.ORB of the VarOP TST entry, save ORB offset, turn on
                                ;       timing flag, and start off the first echo command
                                ;
017C  51                        TMEok:  PUSH    CX              ;Save working registers
017D  52                                PUSH    DX
017E  A2 0096 R                         MOV     VarOP.SK,AL     ;Save NAD
                                ;
                                        DISABLE                 ;Mask off transporter
0181  9C                      +         PUSHF
0182  FA                      +         CLI
0183  50                      +         PUSH    AX
0184  E4 21                   +         IN      AL,$P82op1
0186  0C 05                   +         OR      AL,$P82EML
0188  E6 21                   +         OUT     $P82op1,AL
018A  58                      +         POP     AX
```

```
018B  9D                          +          POPF
                                             TIME                    ;Get the current time
018C  B4 2C                       +          MOV     AH,2CH
018E  CD 21                       +          INT     21H
0190  C6 06 0094 R 00                        MOV     VarOP.PN,$True  ;Set timing loop flag
0195  89 36 009E R                           MOV     VarOP.Dsz,SI    ;Save TST entry offset
0199  BE 0094 R                              MOV     SI,OFFSET VarOP ;Now point to VarOP TST
019C  89 4C 10                               MOV     [SI].TT_Orb,CX  ;Save the
019F  89 54 12                               MOV     [SI].TT_Orbs,DX ;  current time
                                             ENABLE                  ;Restore transporter
01A2  9C                          +          PUSHF
01A3  FA                          +          CLI
01A4  50                          +          PUSH    AX
01A5  E4 21                       +          IN      AL,$P82op1
01A7  24 FA                       +          AND     AL,$P82enb
01A9  E6 21                       +          OUT     $P82op1,AL
01AB  58                          +          POP     AX
01AC  9D                          +          POPF
                                    ;
01AD  53                                     PUSH    BX
01AE  8A 4C 02                               MOV     CL,[SI].SK      ;Fetch NAD
01B1  8B 5C 04                               MOV     BX,[SI].CCB     ;Calculate transporter
01B4  8D 5F 04                               LEA     BX,[BX].Sock    ;  address and send
01B7  E8 000B R                              CALL    TL_BYT_OUT      ;  NAD to memory
                                    ;
01BA  B1 02                                  MOV     CL,$Echo        ;Now make sure there
01BC  88 4C 03                               MOV     [SI].TT_OP,CL   ;  is an Echo command
01BF  8B 5C 04                               MOV     BX,[SI].CCB     ;  in the TST entry
01C2  8D 1F                                  LEA     BX,[BX].CC_Cmnd ;  and in the CCB
01C4  E8 000B R                              CALL    TL_BYT_OUT
                                    ;
01C7  C6 44 01 FF                            MOV     [SI].TT_ST,$TrSU  ;Set TST status
01CB  8B 4C 04                               MOV     CX,[SI].CCB     ;Go do the first Echo
01CE  8B 5C 06                               MOV     BX,[SI].RR
01D1  E8 0043 R                              CALL    TL_DO_CMND
01D4  5B                                     POP     BX              ;Restore registers
01D5  5A                                     POP     DX
01D6  59                                     POP     CX
                                    ;
                                             TRACE   E:TL-SET-TIME
01D7  80 3E 000B E 00             +          CMP BYTE PTR EC.Trce,$True
01DC  75 0F                       +          JNE     ??0014
01DE  50                          +          PUSH    AX
01DF  B0 00                       +          MOV     AL,0
01E1  56                          +          PUSH    SI
01E2  BE 00F7 R                   +          MOV     SI,OFFSET ??0015
01E5  E8 0000 E                   +          CALL    EM_TRACE
01E8  5E                          +          POP     SI
01E9  58                          +          POP     AX
01EA  EB 01 90                    +          JMP NEAR PTR ??0014
01ED                              + PC_CODE  ENDS
00F7                              + PC_DATA  SEGMENT PUBLIC 'DATA'
00F7  00                          + ??0015   LRB     (,OFFSET ??0017 - OFFSET ??0016)
00F8  0D                          +
00F9  45 3A 54 4C 2D 53           + ??0016   DB      "E:TL-SET-TIME"
```

```
0106                            + PC_DATA ENDS
01ED                            + PC_CODE SEGMENT PUBLIC 'CODE'
01ED                            + ??0014  LABEL   NEAR
                                  ;
01ED  C3                                RET                     ;Return to caller
                                  ;
                                  ;     The following routine is the TL-INT-TIME interrupt service routine,
                                  ;     which is an integral part of the TL-SET-TIME routine.  This routine
                                  ;     checks if there is a timing loop, and if so checks for time expired.
                                  ;     If it has expired, the ORB completion flag is set accordingly, and
                                  ;     the receive is cancelled.  If time remains, another Echo is issued.
                                  ;
01EE                              TL_INT_TIME  LABEL  NEAR
                                  ;
                                        TRACE   S:TL-INT-TIME
01EE  80 3E 000B E 00           +     CMP BYTE PTR EC.Trce,$True
01F3  75 0F                     +     JNE     ??0018
01F5  50                        +     PUSH    AX
01F6  B0 00                     +     MOV     AL,0
01F8  56                        +     PUSH    SI
01F9  BE 0106 R                 +     MOV     SI,OFFSET ??0019
01FC  E8 0000 E                 +     CALL    EM_TRACE
01FF  5E                        +     POP     SI
0200  58                        +     POP     AX
0201  EB 01 90                  +     JMP NEAR PTR ??0018
0204                            + PC_CODE ENDS
0106                            + PC_DATA SEGMENT PUBLIC 'DATA'
0106  00                        + ??0019  LRB     (,OFFSET ??001B - OFFSET ??001A)
0107  0D                        +
0108  53 3A 54 4C 2D 49         + ??001A  DB      "S:TL-INT-TIME"
0115                            + PC_DATA ENDS
0204                            + PC_CODE SEGMENT PUBLIC 'CODE'
0204                            + ??0018  LABEL   NEAR
                                  ;
0204  80 3C 00                          CMP     [SI].PN,$True   ;If flag is true,
0207  74 01                             JE      TIMlp           ;  go process timing loop
0209  C3                                RET                     ;Exit if no timing loop
                                  ;
020A  80 7C 03 02                 TIMlp:  CMP     [SI].TT_OP,$Echo        ;Continue if last
020E  74 01                             JE      TIMlpp                  ;  command was Echo
0210  C3                                RET                             ;Exit if not
                                  ;
0211  8B 7C 0A                   TIMlpp: MOV     DI,[SI].Dsz     ;Fetch Receive TST entry offset
0214  8B 5D 10                           MOV     BX,[DI].TT_Orb  ;Fetch location ORB
0217  3C C0                              CMP     AL,$TrEA        ;Echo not acknowledged
0219  74 2D                              JE      TIMlp0          ;  is the same as timeout
                                  ;
                                  ;     If timeout occurs, the timing loop is ended, an End Receive is
                                  ;     issued for the socket, and the ORB completion code is set
                                  ;
021B  C6 04 01                    TIMout: MOV     [SI].PN,$False  ;Turn off timing
021E  C6 05 01                    TIMoux: MOV     [DI].PN,$False  ;Turn off reception
0221  C6 07 03                            MOV     [BX].CC,$OPtto  ;Set timeout into ORB
                                  ;
0224  8A 4D 04                           MOV     CL,[DI].Sock    ;Fetch receive socket number
```

```
0227  8B 7C 04                        MOV     DI,[SI].CCB     ;Calculate transporter
022A  8D 5D 04                        LEA     BX,[DI].Sock    ;  address and send
022D  E8 000B R                       CALL    TL_BYT_OUT      ;  socket to CCB
                                ;
0230  B1 10                           MOV     CL,$EndRcv      ;Now insert the End
0232  88 4C 03                        MOV     [SI].TT_OP,CL   ;  Receive in the TST
0235  8D 1D                           LEA     BX,[DI].CC_Cmnd ;  and into the CCB
0237  E8 000B R                       CALL    TL_BYT_OUT
                                ;
                                ;       Reenter here to do either the End Receive or Echo command
                                ;       and then return to the interrupt driver
                                ;
023A  C6 44 01 FF               TIMxit: MOV     [SI].TT_ST,$TrSU  ;Set TST status
023E  8B 4C 04                        MOV     CX,[SI].CCB     ;Go do the command
0241  8B 5C 06                        MOV     BX,[SI].RR
0244  E8 0043 R                       CALL    TL_DO_CMND
0247  C3                              RET                     ;Return
                                ;
                                ;       If the Echo was acknowledged, test for timeout
                                ;
0248                            TIM1p0: DISABLE                 ;Mask off transporter
0248  9C                      +         PUSHF
0249  FA                      +         CLI
024A  50                      +         PUSH    AX
024B  E4 21                   +         IN      AL,$P82op1
024D  0C 05                   +         OR      AL,$P82EML
024F  E6 21                   +         OUT     $P82op1,AL
0251  58                      +         POP     AX
0252  9D                      +         POPF
                                        TIME                    ;Get current time
0253  B4 2C                   +         MOV     AH,2CH
0255  CD 21                   +         INT     21H
0257  8B 44 10                        MOV     AX,[SI].TT_Orb  ;If hours value of previous
025A  3A E5                            CMP     AH,CH           ;  not equal to current
025C  74 03                            JE      TIM1p1          ;  increment current minutes
025E  80 C1 3C                         ADD     CL,60           ;  field by 1 hour
0261  32 E4                     TIM1p1: XOR     AH,AH           ;Isolate minutes fields
0263  32 ED                            XOR     CH,CH
0265  2B C8                            SUB     CX,AX           ;CL has minutes difference
                                ;
0267  8B 44 12                        MOV     AX,[SI].TT_Orbs ;Isolate seconds value
026A  86 C4                            XCHG    AL,AH           ; of previous time
026C  32 E4                            XOR     AH,AH
026E  86 D6                            XCHG    DL,DH           ;Isolate seconds value
0270  32 F6                            XOR     DH,DH           ;  of current time
0272  2B D0                            SUB     DX,AX           ;Seconds difference in DX
                                ;
0274  B0 3C                            MOV     AL,60           ;Compute total
0276  F6 E1                            MUL     CL              ;  difference in
0278  03 D0                            ADD     DX,AX           ;  seconds
027A  32 E4                            XOR     AH,AH
                                ;
                                        ENABLE                  ;Restore transporter
027C  9C                      +         PUSHF
027D  FA                      +         CLI
```

```
027E  50                    +          PUSH    AX
027F  E4 21                 +          IN      AL,$P82op1
0281  24 FA                 +          AND     AL,$P82enb
0283  E6 21                 +          OUT     $P82op1,AL
0285  58                    +          POP     AX
0286  9D                    +          POPF
0287  8A 47 01                        MOV     AL,[BX].OP      ;Compare difference
028A  3B D0                          CMP     DX,AX           ;  with period requested
028C  7D 8D                          JGE     TIMout          ;Timeout if not less
                          ;
028E  EB AA                          JMP     TIMxit          ;Redo Echo if less
                          ;
0290                      TL_SET_TIME  ENDP
                          ;
                          ;*****************************************************************************
                          ;Module name:  Return Command Result (TL-RTN-RSLT)                           |
                          ;                                                                            |
                          ;Version:  2.0                                                               |
                          ;                                                                            |
                          ;Last Update: 4 January 1983                                                 |
                          ;                                                                            |
                          ;Function:  The Return Command Result routine is called by the               |
                          ;           Command Interpreter to return the result of a command            |
                          ;           to the requesting machine.  TL-RTN-RSLT returns the data         |
                          ;           from the location specified by the ORB, and also sends           |
                          ;           the current value of the disk status in field RQ.DSC             |
                          ;                                                                            |
                          ;Procedure:  Program Logic Description, Section 2.6.2                        |
                          ;                                                                            |
                          ;Called By:  CI-MAIN                                                         |
                          ;                                                                            |
                          ;Routines called:  TL-SET-SEND, TL-WRD-OUT, TL-BYT-OUT                       |
                          ;                  TL-DTA-OUT, LC-RTN-RSLT                                   |
                          ;                                                                            |
                          ;Input:  DS:(SI) = Location of the ORB                                       |
                          ;                                                                            |
                          ;Output:  Completion code returned in ORB                                    |
                          ;                                                                            |
                          ;Error procedures:  None                                                     |
                          ;                                                                            |
                          ;Written by: R.B. Talmadge, Computer Technology Ltd                          |
                          ;                                                                            |
                          ;Updates:  None                                                              |
                          ;*****************************************************************************
                          ;
0290                      TL_RTN_RSLT  PROC  NEAR
                          ;
                                  TRACE   S:TL-RTN-RSLT
0290  80 3E 000B E 00     +          CMP BYTE PTR EC.Trce,$True
0295  75 0F               +          JNE     ??001C
0297  50                  +          PUSH    AX
0298  B0 00               +          MOV     AL,0
029A  56                  +          PUSH    SI
029B  BE 0115 R           +          MOV     SI,OFFSET ??001D
```

```
029E  E8 0000 E                +          CALL    EM_TRACE
02A1  5E                       +          POP     SI
02A2  58                       +          POP     AX
02A3  EB 01 90                 +          JMP NEAR PTR ??001C
02A6                           + PC_CODE ENDS
0115                           + PC_DATA SEGMENT PUBLIC 'DATA'
0115  00                       + ??001D  LRB     (,OFFSET ??001F - OFFSET ??001E)
0116  0D                       +
0117  53 3A 54 4C 2D 52        + ??001E  DB      "S:TL-RTN-RSLT"
0124                           + PC_DATA ENDS
02A6                           + PC_CODE SEGMENT PUBLIC 'CODE'
02A6                           + ??001C  LABEL   NEAR
                                 ;
02A6  C6 04 FF                           MOV     [SI].CC,$OPinp    ;Denote 'operation in progress'
02A9  C6 44 01 10                        MOV     [SI].OP,$MaxRty   ;Operation retry count
02AD  57                                 PUSH    DI                ;Save register
02AE  8B 3E 0004 E                       MOV     DI,RQ.Top         ;  and find top item in CRQ
                                 ;
                                 ;       If the top item is not queued or is not the current
                                 ;       command, exit with an 'invalid ORB' indication
                                 ;
02B2  8A 05                              MOV     AL,[DI].Use     ;Fetch Usage field and test flag bits
02B4  24 C0                              AND     AL,$RQUseR+$RQUseE
02B6  3C C0                              CMP     AL,$RQUseR+$RQUseE
02B8  74 05                              JE      RSLok           ;Go process if conditions ok
02BA  5F                                 POP     DI              ;If not, restore register'
02BB  C6 04 02                           MOV     [SI].CC,$OPtrb  ;  indicate termination,
02BE  C3                                 RET                     ;  and exit
                                 ;
                                 ;       If the CRQ item is OK, test for a local or immediate command
                                 ;       and pass the baton to the appropriate routine if either
                                 ;
02BF  F6 05 20                   RSLok:  TEST    [DI].Use,$RQUseL
02C2  74 03                              JZ      RSLnlc          ;If local, transfer to
02C4  E9 0000 E                          JMP     LC_RTN_RSLT     ;  Local Command Send routine
                                 ;
02C7  F6 05 10                   RSLnlc: TEST    [DI].Use,$RQUseI
02CA  74 05                              JZ      RSLdo
02CC  5F                                 POP     DI              ;If internal, set
02CD  C6 04 00                           MOV     [SI].CC,$OPcme  ;  operation complete, no
02D0  C3                                 RET                     ;  error, and exit
                                 ;
                                 ;       If a the command is from another machine, set up the CCB
                                 ;       and user control area for the message, move the data from
                                 ;       memory to the transporter, and execute the send operation
                                 ;
02D1  53                         RSLdo:  PUSH    BX              ;Save Working Registers
02D2  51                                 PUSH    CX
02D3  52                                 PUSH    DX
02D4  8B 4D 02                           MOV     CX,[DI].Len     ;Reply length from CRQ
02D7  3B 4C 02                           CMP     CX,[SI].Sze     ;Compare with length in
02DA  7E 03                              JLE     RSLdo1          ;  ORB and take the
02DC  8B 4C 02                           MOV     CX,[SI].Sze     ;  smaller of the values
                                 ;
02DF  89 4C 02                   RSLdo1: MOV     [SI].Sze,CX     ;Save data length
```

```
02E2  41                                    INC     CX              ;Add 1 for disk status
02E3  8B 1E 0072 R                          MOV     BX,TrmRPY.RR    ;Compute transporter
02E7  83 C3 04                              ADD     BX,TYPE RRKD    ;  address control area
02EA  E8 001A R                             CALL    TL_WRD_OUT      ;Insert reply length
                                    ;
02ED  8A 0E 0001 E                          MOV     CL,RQ.Dsc       ;Fetch disk status
02F1  8D 5F 02                              LEA     BX,[BX].Dstat   ;Transporter address status
02F4  E8 000B R                             CALL    TL_BYT_OUT      ;Insert disk status
                                    ;
02F7  8B 4C 02                              MOV     CX,[SI].Sze     ;Fetch data length and
02FA  8B 1E 0074 R                          MOV     BX,TrmRPY.Dtl   ;  transporter address data
02FE  56                                    PUSH    SI
02FF  8B 74 04                              MOV     SI,[SI].RB_Data     ;Memory address of data
0302  E8 0085 R                             CALL    TL_DTA_OUT      ;Insert data into transporter
0305  5E                                    POP     SI
                                    ;
0306  8A 4D 01                              MOV     CL,[DI].NAD     ;Fetch network address
0309  8B 1E 0070 R                          MOV     BX,TrmRPY.CCB   ;Compute transporter location
030D  8D 5F 0B                              LEA     BX,[BX].Dhost   ;  to place the address
0310  E8 000B R                             CALL    TL_BYT_OUT      ;Put into CCB
                                    ;
0313  8B DE                                 MOV     BX,SI           ;ORB offset to BX for TL-SET-SEND
0315  B8 0000                              MOV     AX,$True        ;Indicate there is an ORB
0318  BE 006C R                             MOV     SI,OFFSET TrmRPY  ;Fetch TST entry offset
031B  E8 0133 R                             CALL    TL_SET_SEND       ;  and go do transmission
                                    ;
                                    ;       Now restore registers and wait around until the completion flag
                                    ;       in the ORB changes, then exit.  The flag will be changed by the
                                    ;       TL-INT-RSLT routine (see below), which is the interrupt service
                                    ;       routine for socket B0 send, TST entry TrmRPY.
                                    ;
031E  8B F3                                 MOV     SI,BX           ;Restore all registers
0320  5A                                    POP     DX
0321  59                                    POP     CX
0322  5B                                    POP     BX
0323  5F                                    POP     DI
0324  E8 0000 E                             CALL    TL_INT_WAIT     ;Call the wait routine
                                    ;
0327  80 3C 07                              CMP     [SI].CC,$OPtcl          ;Exit unless incorrect
032A  75 0A                                 JNE     RSLdx                   ;  control length failure
032C  F6 06 0001 E 80                       TEST    RQ.Dsc,$RQdse           ;If so, and if the disk
0331  74 03                                 JZ      RSLdx                   ;  status shows error,
0333  E8 0393 R                             CALL    TL_FETCH_LC             ;  send result via Fetch LC
                                    ;
0336                                RSLdx   LABEL   NEAR
                                    ;
                                            TRACE   E:TL-RTN-RSLT
0336  80 3E 000B E 00          +            CMP BYTE PTR EC.Trce,$True
033B  75 0F                    +            JNE     ??0020
033D  50                       +            PUSH    AX
033E  B0 00                    +            MOV     AL,0
0340  56                       +            PUSH    SI
0341  BE 0124 R                +            MOV     SI,OFFSET ??0021
0344  E8 0000 E                +            CALL    EM_TRACE
0347  5E                       +            POP     SI
```

MOV BX, TrmRPY.CC ; Fetch Transporter
LEA BX, [BX].Data ; address CCB data byte field
MOV CX, [SI].SZE
CALL TL_WRD_OUT ; Get data size

```
0348  58                        +          POP      AX
0349  EB 01 90                  +          JMP NEAR PTR ??0020
034C                            + PC_CODE ENDS
0124                            + PC_DATA SEGMENT PUBLIC 'DATA'
0124  00                        + ??0021  LRB      (,OFFSET ??0023 - OFFSET ??0022)
0125  0D                        +
0126  45 3A 54 4C 2D 52         + ??0022  DB       "E:TL-RTN-RSLT"
0133                            + PC_DATA ENDS
034C                            + PC_CODE SEGMENT PUBLIC 'CODE'
034C                            + ??0020  LABEL    NEAR
                                ;
034C  C3                                 RET                         ;  then exit
                                ;
                                ;-------------------------------------------------------------------------
                                ; The following routine is the TL-INT-RSLT interrupt service routine,   |
                                ; which is an integral part of the TL-RTN-RSLT routine (Section 2.6.2) |
                                ;-------------------------------------------------------------------------
                                ;
034D                            TL_INT_RSLT  LABEL  NEAR
                                ;
                                          TRACE    S:TL-INT-RSLT
034D  80 3E 000B E 00           +         CMP BYTE PTR EC.Trce,$True
0352  75 0F                     +         JNE      ??0024
0354  50                        +         PUSH     AX
0355  B0 00                     +         MOV      AL,0
0357  56                        +         PUSH     SI
0358  BE 0133 R                 +         MOV      SI,OFFSET ??0025
035B  E8 0000 E                 +         CALL     EM_TRACE
035E  5E                        +         POP      SI
035F  58                        +         POP      AX
0360  EB 01 90                  +         JMP NEAR PTR ??0024
0363                            + PC_CODE ENDS
0133                            + PC_DATA SEGMENT PUBLIC 'DATA'
0133  00                        + ??0025  LRB      (,OFFSET ??0027 - OFFSET ??0026)
0134  0D                        +
0135  53 3A 54 4C 2D 49         + ??0026  DB       "S:TL-INT-RSLT"
0142                            + PC_DATA ENDS
0363                            + PC_CODE SEGMENT PUBLIC 'CODE'
0363                            + ??0024  LABEL    NEAR
                                ;
0363  80 3C 00                           CMP      [SI].PN,$True    ;There should be an ORB
0366  74 08                              JE       RSLokk
                                          LOG      $MND06           ;Log an error and
0368  50                        +         PUSH     AX
0369  B0 13                     +         MOV      AL,$MND06
036B  E8 0000 E                 +         CALL     DM_ERR_LOG
036E  58                        +         POP      AX
036F  C3                                 RET                         ;  exit if no ORB
                                ;
0370  8B 5C 10                  RSLokk:  MOV      BX,[SI].TT_Orb   ;Fetch ORB location from TST entry
0373  A8 80                              TEST     AL,$TrNG
0375  75 04                              JNZ      RSLng            ;Set completion code and
0377  C6 07 00                           MOV      [BX].CC,$OPcne   ;  exit if successful
037A  C3                                 RET                       ;  completion
                                ;
```

```
037B  3C 83                   RSLng:  CMP     AL,$TrCL          ;If receive socket
037D  75 04                           JNE     RSLng1            ;  has different control
037F  C6 07 07                        MOV     [BX].CC,$OPtcl    ;  length, terminate
0382  C3                              RET                       ;  with that indication
                              ;
0383  FE 4F 01               RSLng1: DEC     [BX].OP           ;Decrement the retry count
0386  75 04                           JNZ     RSLrty            ;  and retry transmission if not zero
0388  C6 07 05                        MOV     [BX].CC,$OPtng    ;Indicate transporter didn't
038B  C3                              RET                       ;  do its thing and exit
                              ;
038C  BE 006C R               RSLrty: MOV     SI,OFFSET TrmRPY  ;Fetch TST entry offset
038F  E8 00FE R                       CALL    TL_SET_RTRY       ;  and go do transmission
0392  C3                              RET                      ;Return to driver
                              ;
0393                          TL_RTN_RSLT  ENDP
                              ;
                              ;*****************************************************************************
                              ;Module name:  Fetch Long Command (TL-FETCH-LC)                               |
                              ;                                                                             |
                              ;Version:  2.0                                                                |
                              ;                                                                             |
                              ;Last Update: 4 January 1983                                                  |
                              ;                                                                             |
                              ;Function:  The Fetch Long Command routine is called by the Command           |
                              ;           Interpreter to fetch the remainder of a long command.             |
                              ;           TL-FETCH-LC carries out the Disk Server protocol procedure        |
                              ;           and returns the data to the location specified by the             |
                              ;           input ORB.                                                        |
                              ;                                                                             |
                              ;Procedure:  Program Logic Description, Section 2.6.1                         |
                              ;                                                                             |
                              ;Called By:  CI-MAIN                                                          |
                              ;                                                                             |
                              ;Routines called:  TL-SET-SEND, TL-SET-RECV, TL-SET-TIME,                     |
                              ;                  TL-BYT-OUT, TL-WRD-OUT, TL-BYT-IN,                         |
                              ;                  TL-WRD-IN, LC-FETCH-LC, TL-FETCH-IC                        |
                              ;                                                                             |
                              ;Input: (DS):(SI) = Location of the ORB                                       |
                              ;                                                                             |
                              ;Output:  Completion code returned in ORB                                     |
                              ;                                                                             |
                              ;Error procedures:  None                                                      |
                              ;                                                                             |
                              ;Written by: R.B. Talmadge, Computer Technology Ltd                           |
                              ;                                                                             |
                              ;Updates:  None                                                               |
                              ;*****************************************************************************
                              ;
0393                          TL_FETCH_LC  PROC  NEAR
                              ;
                                      TRACE   S:TL-FETCH-LC
0393  80 3E 000B E 00         +        CMP BYTE PTR EC.Trce,$True
0398  75 0F                   +        JNE     ??0028
039A  50                      +        PUSH    AX
```

```
039B  B0 00                    +          MOV     AL,0
039D  56                       +          PUSH    SI
039E  BE 0142 R                +          MOV     SI,OFFSET ??0029
03A1  E8 0000 E                +          CALL    EM_TRACE
03A4  5E                       +          POP     SI
03A5  58                       +          POP     AX
03A6  EB 01 90                 +          JMP NEAR PTR ??0028
03A9                           + PC_CODE  ENDS
0142                           + PC_DATA  SEGMENT PUBLIC 'DATA'
0142  00                       + ??0029   LRB     (,OFFSET ??002B - OFFSET ??002A)
0143  0D                       +
0144  53 3A 54 4C 2D 46        + ??002A   DB      "S:TL-FETCH-LC"
0151                           + PC_DATA  ENDS
03A9                           + PC_CODE  SEGMENT PUBLIC 'CODE'
03A9                           + ??0028   LABEL   NEAR
                                 ;
03A9  C6 04 FF                           MOV     [SI].CC,$OPinp    ;Denote 'operation in progress'
03AC  C6 44 01 10                        MOV     [SI].OP,$MaxRty   ;Operation retry count
03B0  57                                 PUSH    DI                ;Save register
03B1  8B 3E 0004 E                       MOV     DI,RQ.Top         ;  and find top item in CRQ
                                 ;
                                 ;       If the top item is not queued or is not the current
                                 ;       command, exit with an 'invalid ORB' indication
                                 ;
03B5  8A 05                              MOV     AL,[DI].Use     ;Fetch Usage field and test flag bits
03B7  24 C0                              AND     AL,$RQUseR+$RQUseE
03B9  3C C0                              CMP     AL,$RQUseR+$RQUseE
03BB  74 05                              JE      FCHok           ;Go process if conditions ok
03BD  5F                                 POP     DI              ;If not, restore register'
03BE  C6 04 02                           MOV     [SI].CC,$OPtrb  ;  indicate termination,
03C1  C3                                 RET                     ;  and exit
                                 ;
                                 ;       If the CRQ item is OK, test for a local or immediate command
                                 ;       and pass the baton to the appropriate routine if either
                                 ;
03C2  F6 05 20                   FCHok:  TEST    [DI].Use,$RQUseL
03C5  74 04                              JZ      FCHnlc
03C7  5F                                 POP     DI              ;If local, transfer to
03C8  E9 0000 E                          JMP     LC_FETCH_LC     ;  Local Command Fetch routine
                                 ;
03CB  F6 05 10                   FCHnlc: TEST    [DI].Use,$RQUseI
03CE  74 04                              JZ      FCHdo
03D0  5F                                 POP     DI              ;If internal, transfer to
03D1  E9 0000 E                          JMP     TL_FETCH_IC     ;  Internal Command Fetch routine
                                 ;
                                 ;       If a the command is from another machine, prepare the GO
                                 ;       message and then set up the A0 socket to receive the reply
                                 ;
03D4  53                         FCHdo:  PUSH    BX              ;Save Working Registers
03D5  51                                 PUSH    CX
03D6  52                                 PUSH    DX
03D7  B9 474F                            MOV     CX,'GO'         ;Setup 'GO' message with
03DA  8A 26 0001 E                       MOV     AH,RQ.Dsc       ;  error bit from the
03DE  80 E4 80                           AND     AH,$RQdse       ;  disk status byte
03E1  0A EC                              OR      CH,AH
```

```
03E3  8B 1E 0088 R                      MOV     BX,TrmGO.Dtl    ;Fetch transporter address
03E7  E8 001A R                         CALL    TL_WRD_OUT      ;  of GO message and store data
                                ;
03EA  8A 4D 01                          MOV     CL,[DI].NAD     ;Fetch network address
03ED  8B 1E 0084 R                      MOV     BX,TrmGO.CCB    ;Compute transporter location
03F1  8D 5F 0B                          LEA     BX,[BX].Dhost   ;  to place the address
03F4  E8 000B R                         CALL    TL_BYT_OUT      ;Put into CCB
                                ;
                                ;       The receive of the GO reply is set up unless the disk status
                                ;       to be returned shows an error.  In that case, the GO is sent
                                ;       at this point without preparing for a reply, as the sender
                                ;       should not respond with more of the command unless status is ok
                                ;
                                        STATUS                  ;Disable interrupts
03F7  9C                        +       PUSHF
03F8  FA                        +       CLI
03F9  8B DE                             MOV     BX,SI           ;Setup needs ORB offset in SI
03FB  B8 0000                           MOV     AX,$True        ;Indicate there is an ORB
03FE  BE 0030 R                         MOV     SI,OFFSET RcvGO ;Fetch TST entry offset
0401  F6 06 0001 E 80                   TEST    RQ.Dsc,$RQdse   ;  for receive GO if
0406  75 06                             JNZ     FCHdo1          ;  disk status not error
0408  E8 00AB R                         CALL    TL_SET_RECV     ;  and set up the reply
040B  EB 07 90                          JMP     FCHdo2          ;Wait for result
                                ;
040E  BE 0080 R                 FCHdo1: MOV     SI,OFFSET TrmGO ;Fetch TST entry send GO
0411  E8 0133 R                         CALL    TL_SET_SEND     ;  and send it off
                                ;
                                ;       Now restore registers and wait around until the completion flag
                                ;       in the ORB changes, then exit.  The flag will be changed by the
                                ;       TL-INT-LCR or the TL-INT-LCT routine (see below), which are the
                                ;       interrupt service routines for socket A0 receive (TST entry RcvGO)
                                ;       and for socket A0 transmit (TST entry TrmGO).
                                ;
0414                            FCHdo2: RESTORE                 ;Restore interrupt status
0414  9D                        +       POPF
0415  8B F3                             MOV     SI,BX           ;Restore all registers
0417  5A                                POP     DX
0418  59                                POP     CX
0419  5B                                POP     BX
041A  5F                                POP     DI
041B  E8 0000 E                         CALL    TL_INT_WAIT     ;Call the wait routine
                                ;
                                        TRACE   E:TL-FETCH-LC
041E  80 3E 000B E 00           +       CMP BYTE PTR EC.Trce,$True
0423  75 0F                     +       JNE     ??002C
0425  50                        +       PUSH    AX
0426  B0 00                     +       MOV     AL,0
0428  56                        +       PUSH    SI
0429  BE 0151 R                 +       MOV     SI,OFFSET ??002D
042C  E8 0000 E                 +       CALL    EM_TRACE
042F  5E                        +       POP     SI
0430  58                        +       POP     AX
0431  EB 01 90                  +       JMP NEAR PTR ??002C
0434                            + PC_CODE ENDS
0151                            + PC_DATA SEGMENT PUBLIC 'DATA'
```

```
0151  00                        + ??002D  LRB     (,OFFSET ??002F - OFFSET ??002E)
0152  0D                        +
0153  45 3A 54 4C 2D 46         + ??002E  DB      "E:TL-FETCH-LC"
0160                            + PC_DATA ENDS
0434                            + PC_CODE SEGMENT PUBLIC 'CODE'
0434                            + ??002C  LABEL   NEAR
                                  ;
0434  C3                              RET                     ;  then exit
                                  ;
                                  ;-------------------------------------------------------------------------
                                  ;  The following routine is the TL-INT-LCR interrupt service routine,   |
                                  ;  which is an integral part of the TL-FETCH-LC routine (Section 2.6.1)|
                                  ;-------------------------------------------------------------------------
                                  ;
0435                              TL_INT_LCR  LABEL  NEAR
                                  ;
                                        TRACE   S:TL-INT-LCR
0435  80 3E 000B E 00           +        CMP BYTE PTR EC.Trce,$True
043A  75 0F                     +        JNE     ??0030
043C  50                        +        PUSH    AX
043D  B0 00                     +        MOV     AL,0
043F  56                        +        PUSH    SI
0440  BE 0160 R                 +        MOV     SI,OFFSET ??0031
0443  E8 0000 E                 +        CALL    EM_TRACE
0446  5E                        +        POP     SI
0447  58                        +        POP     AX
0448  EB 01 90                  +        JMP NEAR PTR ??0030
044B                            + PC_CODE ENDS
0160                            + PC_DATA SEGMENT PUBLIC 'DATA'
0160  00                        + ??0031  LRB     (,OFFSET ??0033 - OFFSET ??0032)
0161  0C                        +
0162  53 3A 54 4C 2D 49         + ??0032  DB      "S:TL-INT-LCR"
016E                            + PC_DATA ENDS
044B                            + PC_CODE SEGMENT PUBLIC 'CODE'
044B                            + ??0030  LABEL   NEAR
                                  ;
044B  80 3C 00                        CMP     [SI].PN,$True   ;There should be an ORB
044E  74 01                           JE      SUGok           ;Just exit if one
0450  C3                              RET                     ;  is not there
                                  ;
0451  8B 5C 10                  SUGok:  MOV     BX,[SI].TT_Orb  ;Fetch ORB location from TST entry
0454  3C FE                           CMP     AL,$TrSS        ;If the entry is for successful
0456  74 14                           JE      SUGtrm          ;  setup, send the GO message
0458  3C 00                           CMP     AL,$TrOK        ;If a message has been received
045A  74 19                           JE      SUGmsg          ;  go to fetch the data
                                  ;
045C  FE 4F 01                        DEC     [BX].OP         ;Decrement the retry count
045F  75 04                           JNZ     SUGrty          ;  and retry setup if not zero
0461  C6 07 06                        MOV     [BX].CC,$OPttf  ;Indicate transporter didn't
0464  C3                              RET                     ;  do its thing and exit
                                  ;
0465  BE 0030 R                 SUGrty: MOV     SI,OFFSET RcvGO ;Fetch TST entry offset
0468  E8 00FE R                       CALL    TL_SET_RTRY     ;  and go to set up the reply
046B  C3                              RET                     ;Return to driver
                                  ;
```

```
                                ;       If the entry is for successful receive setup, send off the
                                ;       GO message and exit.  TL-INT-LCT will set up the timing loop.
                                ;
046C  B0 00                     SUGtrm: MOV     AL,$True        ;Indicate ORB present
046E  BE 0080 R                         MOV     SI,OFFSET TrmGO ;Set TST entry for transmit
0471  E8 0133 R                         CALL    TL_SET_SEND     ;Send off the GO message
0474  C3                                RET                     ;  and return to TL-INT-DRVR
                                ;
                                ;       If the entry if for message received, fetch the data from the
                                ;       transporter to where the ORB specifies, set the completion
                                ;       flag, and then return to the driver.
                                ;
0475  80 3C 00                  SUGmsg: CMP     [SI].PN,$True   ;Just exit if
0478  74 01                             JE      SUGmsh          ;  timeout has
047A  C3                                RET                     ;  occurred
                                ;
047B  C6 04 01                  SUGmsh: MOV     [SI].PN,$False  ;Clear ORB flag
047E  C6 06 0094 R 01                   MOV     VarOP.PN,$False ;Stop timing loop
0483  53                                PUSH    BX              ;Save ORB location
0484  8B 5C 06                         MOV     BX,[SI].RR      ;Get transporter location
0487  8D 5F 01                         LEA     BX,[BX].Shost   ;  of NAD of sender
048A  E8 002C R                         CALL    TL_BYT_IN       ;Fetch the NAD
048D  8B 1E 0004 E                      MOV     BX,RQ.Top       ;Offset of top entry
0491  3A 47 01                          CMP     AL,[BX].NAD     ;Accept the data if from
0494  74 05                             JE      SUGdat          ;  the right machine
0496  5B                                POP     BX              ;If not, recover ORB,
0497  C6 07 03                         MOV     [BX].CC,$OPtto  ;  treat as a time-out,
049A  C3                                RET                     ;  and return to driver
                                ;
049B  8B 5C 06                  SUGdat: MOV     BX,[SI].RR      ;Get transporter location
049E  8D 5F 02                          LEA     BX,[BX].Rcvlen  ;  of the received data length
04A1  E8 0039 R                         CALL    TL_WRD_IN       ;Fetch the length
                                ;
04A4  5B                                POP     BX              ;Recover ORB offset
04A5  B1 00                             MOV     CL,$OPcne       ;Assume length ok
04A7  3B 47 02                          CMP     AX,[BX].Sze     ;All ok if
04AA  74 0A                             JE      SUGdaz          ;  sizes match
04AC  7C 03                             JL      SUGday          ;Use received size
04AE  8B 47 02                          MOV     AX,[BX].Sze     ;  or ORB size,
04B1  89 47 02                  SUGday: MOV     [BX].Sze,AX     ;  whichever is smaller
04B4  B1 01                             MOV     CL,$OPinl       ;Set for incorrect length
                                ;
04B6  88 0F                     SUGdaz: MOV     [BX].CC,CL      ;Completion code to ORB
04B8  8B C8                             MOV     CX,AX           ;Length to CX for move routine
04BA  8B 7F 04                          MOV     DI,[BX].RB_Data ;Address to transfer to
04BD  8B 5C 08                          MOV     BX,[SI].Dtl     ;Transporter location of data
04C0  E8 0098 R                         CALL    TL_DTA_IN       ;Go fetch the data
                                ;
04C3  C3                                RET                     ;Return to driver
                                ;
                                ;-----------------------------------------------------------------------
                                ;  The following routine is the TL-INT-LCT interrupt service routine,   |
                                ;  which is an integral part of the TL-FETCH-LC routine.  This routine  |
                                ;  sets up the timing loop if the GO transmission was successful.  If   |
                                ;  transmission failed, the ORB completion flag is set accordingly.     |
```

```
                                ;--------------------------------------------------------------------------------
                                ;
04C4                            TL_INT_LCT  LABEL  NEAR
                                ;
                                        TRACE   S:TL-INT-LCT
04C4  80 3E 000B E 00         +         CMP BYTE PTR EC.Trce,$True
04C9  75 0F                   +         JNE     ??0034
04CB  50                      +         PUSH    AX
04CC  B0 00                   +         MOV     AL,0
04CE  56                      +         PUSH    SI
04CF  BE 016E R               +         MOV     SI,OFFSET ??0035
04D2  E8 0000 E               +         CALL    EM_TRACE
04D5  5E                      +         POP     SI
04D6  58                      +         POP     AX
04D7  EB 01 90                +         JMP NEAR PTR ??0034
04DA                          + PC_CODE ENDS
016E                          + PC_DATA SEGMENT PUBLIC 'DATA'
016E  00                      + ??0035  LRB     (,OFFSET ??0037 - OFFSET ??0036)
016F  0C                      +
0170  53 3A 54 4C 2D 49       + ??0036  DB      "S:TL-INT-LCT"
017C                          + PC_DATA ENDS
04DA                          + PC_CODE SEGMENT PUBLIC 'CODE'
04DA                          + ??0034  LABEL   NEAR
                                ;
04DA  80 3C 00                          CMP     [SI].PN,$True   ;There should be an ORB
04DD  74 01                             JE      TRGok           ;Just exit if one
04DF  C3                                RET                     ;  is not there
                                ;
04E0  8B 5C 10                  TRGok:  MOV     BX,[SI].TT_Orb  ;Fetch ORB location from TST entry
04E3  A8 80                             TEST    AL,$TrNG        ;If the entry is for successful
04E5  74 10                             JZ      TRGtme          ;  transmission, set up timing loop
                                ;
04E7  FE 4F 01                          DEC     [BX].OP         ;Decrement the retry count
04EA  75 04                             JNZ     TRGrty          ;  and retry setup if not zero
04EC  C6 07 05                          MOV     [BX].CC,$OPtng  ;Indicate transporter didn't
04EF  C3                                RET                     ;  do its thing and exit
                                ;
04F0  BE 0080 R                 TRGrty: MOV     SI,OFFSET TrmGO ;Fetch TST entry offset
04F3  E8 00FE R                         CALL    TL_SET_RTRY     ;  and go do the retry
04F6  C3                                RET                     ;Return to driver
                                ;
                                ;       If the GO was transmitted properly, call the Timeout Setup
                                ;       routine to set a timing loop provided the GO was for a
                                ;       valid command.  If a disk error code was returned, set
                                ;       the ORB completion flag and exit.
                                ;
04F7  F6 06 0001 E 80           TRGtme: TEST    RQ.Dsc,$RQdse   ;Continue if GO was
04FC  74 04                             JZ      TRGtmf          ;  for no disk error
04FE  C6 07 00                          MOV     [BX].CC,$OPcne  ;If no receive expected, set
0501  C3                                RET                     ;  ORB completion and exit
                                ;
0502  BE 0030 R                 TRGtmf: MOV     SI,OFFSET RcvGO ;Fetch TST entry for receive
0505  C6 47 01 0A                      MOV     [BX].OP,$MaxWte ;Set time-out count
0509  8B 3E 0004 E                      MOV     DI,RQ.Top
050D  8A 45 01                          MOV     AL,[DI].NAD     ;Fetch network address
```

```
0510  E8 014B R                 CALL    TL_SET_TIME     ;Set up timing loop
0513  C3                        RET                     ;Return to driver

                          ;
0514                      TL_FETCH_LC  ENDP
                          ;
                          ;*****************************************************************************
                          ;Module name:  Send Name Lookup Message (TL-SEND-NLM)                        |
                          ;                                                                            |
                          ;Version:  2.0                                                               |
                          ;                                                                            |
                          ;Last Update: 14 November 1983                                               |
                          ;                                                                            |
                          ;Function:  The Send Name Lookup Message routine is called by any            |
                          ;           routine which wants to transmit a Name Lookup Message to         |
                          ;           another machine.  TL-SEND-NLM inserts the message into           |
                          ;           transporter memory and calls TL-SET-SEND to do the               |
                          ;           transmission.  The interrupt service routine TL-INT-NLM,         |
                          ;           which is part of TL-SET-SEND, completes the action.              |
                          ;                                                                            |
                          ;Procedure:  Program Logic Description, Section 2.7.6                        |
                          ;                                                                            |
                          ;Called By:  TL-RCV-WHO, TL-RCV-WHERE                                        |
                          ;                                                                            |
                          ;Routines called:  TL-DTA-OUT, TL-SET-SEND, TL-SET-RECV                      |
                          ;                                                                            |
                          ;Input:  DS:(SI) = Location message                                          |
                          ;           (CL) = NAD of machine to receive message                         |
                          ;                                                                            |
                          ;Output:  None.                                                              |
                          ;                                                                            |
                          ;Error procedures:  None                                                     |
                          ;                                                                            |
                          ;Written by: R.B. Talmadge, Computer Technology Ltd                          |
                          ;                                                                            |
                          ;Updates:  None                                                              |
                          ;                                                                            |
                          ;*****************************************************************************
                          ;
0514                      TL_SEND_NLM  PROC  NEAR
                          ;
                                  TRACE   S:TL-SEND-NLM
0514  80 3E 000B E 00     +          CMP BYTE PTR EC.Trce,$True
0519  75 0F               +          JNE     ??0038
051B  50                  +          PUSH    AX
051C  B0 00               +          MOV     AL,0
051E  56                  +          PUSH    SI
051F  BE 017C R           +          MOV     SI,OFFSET ??0039
0522  E8 0000 E           +          CALL    EM_TRACE
0525  5E                  +          POP     SI
0526  58                  +          POP     AX
0527  EB 01 90            +          JMP NEAR PTR ??0038
052A                      + PC_CODE ENDS
017C                      + PC_DATA SEGMENT PUBLIC 'DATA'
017C  00                  + ??0039  LRB     <,OFFSET ??003B - OFFSET ??003A>
017D  0D                  +
```

```
017E  53 3A 54 4C 2D 53       + ??003A  DB      "S:TL-SEND-NLM"
018B                          + PC_DATA ENDS
052A                          + PC_CODE SEGMENT PUBLIC 'CODE'
052A                          + ??0038  LABEL   NEAR
                                ;
052A  80 3E 0044 R 00                   CMP     TrmNLM.PN,$True     ;Cannot send message if Local
052F  75 01                             JNE     NSok                ;  Command doing send, so
0531  C3                                RET                         ;  return and allow reception
                                ;
0532  8B 3E 0048 R             Nsok:    MOV     DI,TrmNLM.CCB   ;Fetch transporter
0536  8D 5D 0B                         LEA     BX,[DI].Dhost   ;  address of NAD
0539  E8 000B R                         CALL    TL_BYT_OUT      ;NAD to transporter
                                ;
053C  B9 0012                           MOV     CX,TYPE NLM     ;Length of message
053F  8D 5D 08                          LEA     BX,[DI].Datl    ;Place length into
0542  E8 001A R                         CALL    TL_WRD_OUT      ;  CCB in transporter
                                ;
0545  8B 1E 004C R                      MOV     BX,TrmNLM.Dtl   ;Transporter address for data
0549  E8 0085 R                         CALL    TL_DTA_OUT      ;Send data to transporter
054C  FE 0E 0005 E                      DEC     MT.MT_Mod       ;Assure reception not restarted
                                ;
0550  BE 0044 R                         MOV     SI,OFFSET TrmNLM        ;Get TST entry offset
0553  B0 01                             MOV     AL,$False               ;Set no ORB
0555  E8 0133 R                         CALL    TL_SET_SEND             ;Go send the message
                                ;
                                        TRACE   E:TL-SEND-NLM
0558  80 3E 000B E 00        +          CMP BYTE PTR EC.Trce,$True
055D  75 0F                   +          JNE     ??003C
055F  50                      +          PUSH    AX
0560  B0 00                   +          MOV     AL,0
0562  56                      +          PUSH    SI
0563  BE 018B R                +          MOV     SI,OFFSET ??003D
0566  E8 0000 E                +          CALL    EM_TRACE
0569  5E                      +          POP     SI
056A  58                      +          POP     AX
056B  EB 01 90                +          JMP NEAR PTR ??003C
056E                          + PC_CODE ENDS
018B                          + PC_DATA SEGMENT PUBLIC 'DATA'
018B  00                      + ??003D  LRB     (,OFFSET ??003F - OFFSET ??003E)
018C  0D                      +
018D  45 3A 54 4C 2D 53       + ??003E  DB      "E:TL-SEND-NLM"
019A                          + PC_DATA ENDS
056E                          + PC_CODE SEGMENT PUBLIC 'CODE'
056E                          + ??003C  LABEL   NEAR
                                ;
056E  C3                                RET                         ;   and return
                                ;
                                ;-------------------------------------------------------------------------------
                                ;  The following routine is the TL-INT-NLM interrupt service routine,          |
                                ;  which is an integral part of the TL-SEND-NLM routine.  This routine         |
                                ;  retries the transmission if there is a failure, until either there          |
                                ;  is success or the retry count is exeeded.  It also restarts Name            |
                                ;  Lookup message reception when the transmission sequence is done.            |
                                ;-------------------------------------------------------------------------------
                                ;
```

```
056F                            TL_INT_NLM  LABEL  NEAR
                                ;
                                        TRACE   S:TL-INT-NLM
056F  80 3E 000B E 00         +         CMP BYTE PTR EC.Trce,$True
0574  75 0F                   +         JNE     ??0040
0576  50                      +         PUSH    AX
0577  B0 00                   +         MOV     AL,0
0579  56                      +         PUSH    SI
057A  BE 019A R               +         MOV     SI,OFFSET ??0041
057D  E8 0000 E               +         CALL    EM_TRACE
0580  5E                      +         POP     SI
0581  58                      +         POP     AX
0582  EB 01 90                +         JMP NEAR PTR ??0040
0585                          + PC_CODE ENDS
019A                          + PC_DATA SEGMENT PUBLIC 'DATA'
019A  00                      + ??0041  LRB     (,OFFSET ??0043 - OFFSET ??0042)
019B  0C                      +
019C  53 3A 54 4C 2D 49       + ??0042  DB      "S:TL-INT-NLM"
01A8                          + PC_DATA ENDS
0585                          + PC_CODE SEGMENT PUBLIC 'CODE'
0585                          + ??0040  LABEL   NEAR
                                ;
0585  80 3C 00                         CMP     [SI].PN,$True   ;If there is an ORB
0588  75 03                            JNE     NSIgo           ;  pass the baton to the
058A  E9 0000 E                        JMP     LC_INT_NLM      ;  Local Command int rtn
                                ;
058D  A8 80                   NSIgo:   TEST    AL,$TrNG        ;If the entry is for sucessful
058F  74 09                            JZ      NSIrs           ;  transmission, restart reception
                                ;
0591  FE 4C 10                         DEC     BYTE PTR [SI].TT_Orb  ;Decrement the retry count
0594  74 04                            JZ      NSIrs                 ;Restart reception if zero
0596  E8 00FE R                        CALL    TL_SET_RTRY           ;If not, do a retry
0599  C3                               RET                           ;Return to driver
                                ;
                                ;       Reenter here if successful or retry count exceeded
                                ;
059A  B0 01                   NSIrs:   MOV     AL,$False       ;Set for no ORB,
059C  BE 0008 R                        MOV     SI,OFFSET RcvNLM  ;  assure TST pointer,
059F  E8 00AB R                        CALL    TL_SET_RECV       ;  request restart,
05A2  C3                               RET                       ;  and exit
                                ;
05A3                            TL_SEND_NLM  ENDP
                                ;
                                ;*****************************************************************
                                ;               End of the Assembly housekeeping goes here      |
                                ;*****************************************************************
                                ;
05A3                            PC_CODE ENDS
                                ;
                                        PUBLIC  TT                      ;TST header
                                        PUBLIC  TTbgn                   ;Start of regular TST entries
                                        PUBLIC  TTend                   ;End of regular entries
                                        PUBLIC  RcvICS                  ;TST entry for ICS receive
                                        PUBLIC  RcvNLM                  ;TST entry for NLM receive
                                        PUBLIC  TrmNLM                  ;TST entry for NLM transmit
```

```
                        PUBLIC  TrmNAD              ;TST entry for sending NAD
                        PUBLIC  VarOP               ;TST entry for variable opns
                        PUBLIC  TL_BYT_IN           ;Fetch byte from transporter
                        PUBLIC  TL_WRD_IN           ;Fetch word from transporter
                        PUBLIC  TL_DTA_IN           ;Fetch string from transporter
                        PUBLIC  TL_BYT_OUT          ;Put byte into transporter
                        PUBLIC  TL_WRD_OUT          ;Put word into transporter
                        PUBLIC  TL_DTA_OUT          ;Put data into transporter
                        PUBLIC  TL_DO_CMND          ;Execute transporter command
                        PUBLIC  TL_SET_RECV         ;Setup message receive
                        PUBLIC  TL_SET_SEND         ;Do message transmission
                        PUBLIC  TL_SET_RTRY         ;Do a command retry
                        PUBLIC  TL_RTN_RSLT         ;Return command result
                        PUBLIC  TL_FETCH_LC         ;Fetch long command
                        PUBLIC  TL_SEND_NLM         ;Send Name Lookup message
                        PUBLIC  TL_SET_TIME         ;Timeout setup
                ;
                        END
```

Macros:

Name	Length
COMPARE.	000F
DISABLE.	0003
ENABLE	0003
ENTER.	0012
EXIT	0006
EYE.	0001
FILLIT	0011
FIV.	0002
GEN_START.	0002
LOG.	0005
MOVEB.	000F
MOVEONE.	000E
RESTORE.	0001
SCANB.	0012
SELEXIT.	0002
SELREST.	000D
SELSAVE.	0018
SIV.	0003
START.	001A
STATUS	0001
TENTER	0003
TEXIT.	0002
TIME	0001
TRACE.	0009

Structures and records:

Name	Width / Shift	# fields / Width	Mask	Initial
CCBLK.	000C	0009		
CC_CMND.	0000			
RRA.	0002			
SOCK	0004			
CC_DATA.	0006			
DATL	0008			
CONLN.	000A			
DHOST.	000B			
CRQE	0008	0004		
USE.	0000			
NAD.	0001			
LEN.	0002			
CMND	0004			
CRQH	0008	0005		
QST.	0000			
DSC.	0001			
END.	0002			
TOP.	0004			
BOT.	0006			
ECT.	0040	001A		
IDEN	0000			
EC_DAT	0008			

```
  EC_DRV . . . . . .  0009
  EC_FLG . . . . . .  000A
  TRCE . . . . . .  000B
  COLD . . . . . .  000C
  EC_TRM . . . . . .  000D
  MAX. . . . . .  000E
  NEXT . . . . . .  0010
  DIR. . . . . .  0012
  NAM. . . . . .  001A
  SERV . . . . . .  0022
  EC_ATRU. . . . . .  002C
  EC_ATRS. . . . . .  002D
  EC_ATRL. . . . . .  002E
  EC_TBI . . . . . .  002F
  TLTMCK . . . . . .  0030
  TLTMRS . . . . . .  0031
  KEEPOUT. . . . . .  0032
  INTDFR . . . . . .  0033
  DOSDKO . . . . . .  0034
  DOSDKS . . . . . .  0036
  DDRH . . . . . .  0038
  DDRHS. . . . . .  003A
  TCKO . . . . . .  003C
  TCKS . . . . . .  003E
LRB. . . . . .  0002  0002
  CDE. . . . . .  0000
  LR_LEN . . . . . .  0001
NLM. . . . . .  0012  0005
  PID. . . . . .  0000
  MGT. . . . . .  0002
  SRC. . . . . .  0004
  DEV. . . . . .  0006
  NL_NAME. . . . . .  0008
NLMT . . . . . .  0016  0009
  MT_USE . . . . . .  0000
  MT_NAD . . . . . .  0001
  OPC. . . . . .  0004
  MT_MOD . . . . . .  0005
  MT_NAME. . . . . .  0006
  MT_SRC . . . . . .  0010
  MT_DEV . . . . . .  0011
  RSVD . . . . . .  0012
ORB. . . . . .  0008  0005
  CC . . . . . .  0000
  OP . . . . . .  0001
  SZE. . . . . .  0002
  RB_DATA. . . . . .  0004
  RB_DATAS . . . . . .  0006
RRKD . . . . . .  0004  0003
  RCODE. . . . . .  0000
  SHOST. . . . . .  0001
  RCVLEN . . . . . .  0002
RRRCV. . . . . .  0004  0002
  LOFCM. . . . . .  0000
  ELOFRP . . . . . .  0002
```

```
RRSND. . . . . .          0003    0002
  ALOFRP . . . . . .      0000
  DSTAT. . . . . .        0002
TSTE . . . . . .          0014    000C
  PN . . . . . .          0000
  TT_ST. . . . . .        0001
  SK . . . . . .          0002
  TT_OP. . . . . .        0003
  CCB. . . . . .          0004
  RR . . . . . .          0006
  DTL. . . . . .          0008
  DSZ. . . . . .          000A
  ISR. . . . . .          000C
  ISRS . . . . . .        000E
  TT_ORB . . . . . .      0010
  TT_ORBS. . . . . .      0012
TSTH . . . . . .          0004    0004
  INT. . . . . .          0000
  TT_CI. . . . . .        0001
  NE . . . . . .          0002
  TT_NAD . . . . . .      0003
TSTLE. . . . . .          0014    000B
  LE_CC. . . . . .        0000
  LE_OP. . . . . .        0001
  LE_LEN . . . . . .      0002
  TRM. . . . . .          0004
  TRMS . . . . . .        0006
  RCV. . . . . .          0008
  RCVS . . . . . .        000A
  LE_ISR . . . . . .      000C
  LE_ISRS. . . . . .      000E
  LE_BP. . . . . .        0010
```

Segments and groups:

```
                N a m e         Size    align   combine class

PCS. . . . . .              GROUP
  PC_CODE. . . . . .        05A3    PARA    PUBLIC  'CODE'
  PC_DATA. . . . . .        01A8    PARA    PUBLIC  'DATA'
```

Symbols:

```
                N a m e         Type    Value   Attr

CI$. . . . . .              Number  0000
DEBUG$ . . . . . .          Number  0001
DM$. . . . . .              Number  0000
DM_ERR_LOG . . . . . .      L NEAR  0000            External
DORES. . . . . .            L NEAR  006C    PC_CODE
DORES1 . . . . . .          L NEAR  0073    PC_CODE
DR$. . . . . .              Number  0000
DTAI . . . . . .            L NEAR  00AA    PC_CODE
DTAINA . . . . . .          L NEAR  00A5    PC_CODE
DTAO . . . . . .            L NEAR  0097    PC_CODE
```

DTAOUA	L NEAR	0092	PC_CODE
EC	V BYTE	0000	External
EI$	Number	0000	
EM_TRACE	L NEAR	0000	External
EYE$	Number	0001	
FCHDO	L NEAR	03D4	PC_CODE
FCHDO1	L NEAR	040E	PC_CODE
FCHDO2	L NEAR	0414	PC_CODE
FCHNLC	L NEAR	03CB	PC_CODE
FCHOK	L NEAR	03C2	PC_CODE
IS$	Number	0000	
LC$	Number	0000	
LC_FETCH_LC	L NEAR	0000	External
LC_INT_NLM	L NEAR	0000	External
LC_RCV_CMND	L NEAR	0000	External
LC_RTN_RSLT	L NEAR	0000	External
MT	V BYTE	0000	External
NA$	Number	0000	
NOCOM$	Number	0000	
NSIGO	L NEAR	058D	PC_CODE
NSIRS	L NEAR	059A	PC_CODE
NSOK	L NEAR	0532	PC_CODE
PI$	Number	0000	
RCVGO	L 0014	0030	PC_DATA
RCVICS	L 0014	001C	PC_DATA Global
RCVNLM	L 0014	0008	PC_DATA Global
RQ	V BYTE	0000	External
RSLDO	L NEAR	02D1	PC_CODE
RSLDO1	L NEAR	02DF	PC_CODE
RSLDX	L NEAR	0336	PC_CODE
RSLNG	L NEAR	037B	PC_CODE
RSLNG1	L NEAR	0383	PC_CODE
RSLNLC	L NEAR	02C7	PC_CODE
RSLOK	L NEAR	02BF	PC_CODE
RSLOKK	L NEAR	0370	PC_CODE
RSLRTY	L NEAR	038C	PC_CODE
SETXST	L NEAR	00CE	PC_CODE
SETXSU	L NEAR	00D7	PC_CODE
SM$	Number	0000	
SNXIT	L NEAR	0119	PC_CODE
STRBC	L NEAR	0076	PC_CODE
STRBCR	L NEAR	0079	PC_CODE
SUGDAT	L NEAR	049B	PC_CODE
SUGDAY	L NEAR	04B1	PC_CODE
SUGDAZ	L NEAR	04B6	PC_CODE
SUGMSG	L NEAR	0475	PC_CODE
SUGMSH	L NEAR	047B	PC_CODE
SUGOK	L NEAR	0451	PC_CODE
SUGRTY	L NEAR	0465	PC_CODE
SUGTRM	L NEAR	046C	PC_CODE
TBINTA$	Number	0001	
TENV$	Number	0000	
TIMLP	L NEAR	020A	PC_CODE
TIMLP0	L NEAR	0248	PC_CODE
TIMLP1	L NEAR	0261	PC_CODE

```
TIMLPP . . . . . .  L NEAR  0211  PC_CODE
TIMOUT . . . . . .  L NEAR  021B  PC_CODE
TIMOUX . . . . . .  L NEAR  021E  PC_CODE
TIMXIT . . . . . .  L NEAR  023A  PC_CODE
TL$. . . . . .  Number  0001
TL_BYT_IN. . . . . .  N PROC  002C  PC_CODE Global  Length =000D
TL_BYT_OUT . . . . . .  N PROC  000B  PC_CODE Global  Length =000F
TL_DO_CMND . . . . . .  N PROC  0043  PC_CODE Global  Length =0042
TL_DTA_IN. . . . . .  N PROC  0098  PC_CODE Global  Length =0013
TL_DTA_OUT . . . . . .  N PROC  0085  PC_CODE Global  Length =0013
TL_FETCH_IC. . . . . .  L NEAR  0000  External
TL_FETCH_LC. . . . . .  N PROC  0393  PC_CODE Global  Length =0181
TL_INT_LCR . . . . . .  L NEAR  0435  PC_CODE
TL_INT_LCT . . . . . .  L NEAR  04C4  PC_CODE
TL_INT_NAD . . . . . .  L NEAR  0000  External
TL_INT_NLM . . . . . .  L NEAR  056F  PC_CODE
TL_INT_RSLT. . . . . .  L NEAR  034D  PC_CODE
TL_INT_TIME. . . . . .  L NEAR  01EE  PC_CODE
TL_INT_WAIT. . . . . .  L NEAR  0000  External
TL_OUTA. . . . . .  L NEAR  0024  PC_CODE
TL_RCV_ICS . . . . . .  L NEAR  0000  External
TL_RCV_NLM . . . . . .  L NEAR  0000  External
TL_RTN_RSLT. . . . . .  N PROC  0290  PC_CODE Global  Length =0103
TL_SEND_NLM. . . . . .  N PROC  0514  PC_CODE Global  Length =008F
TL_SET_CTR . . . . . .  N PROC  0000  PC_CODE Length =000B
TL_SET_RECV. . . . . .  N PROC  00AB  PC_CODE Global  Length =0088
TL_SET_RTRY. . . . . .  L NEAR  00FE  PC_CODE Global
TL_SET_SEND. . . . . .  N PROC  0133  PC_CODE Global  Length =0018
TL_SET_TIME. . . . . .  N PROC  014B  PC_CODE Global  Length =0145
TL_SET_TOPN. . . . . .  L NEAR  00EE  PC_CODE
TL_WRD_IN. . . . . .  N PROC  0039  PC_CODE Global  Length =000A
TL_WRD_OUT . . . . . .  N PROC  001A  PC_CODE Global  Length =0012
TMEOK. . . . . .  L NEAR  017C  PC_CODE
TMEST. . . . . .  L NEAR  0167  PC_CODE
TRACE$ . . . . . .  Number  0001
TRGOK. . . . . .  L NEAR  04E0  PC_CODE
TRGRTY . . . . . .  L NEAR  04F0  PC_CODE
TRGTME . . . . . .  L NEAR  04F7  PC_CODE
TRGTMF . . . . . .  L NEAR  0502  PC_CODE
TRMGO. . . . . .  L 0014  0080  PC_DATA
TRMNAD . . . . . .  L 0014  0058  PC_DATA Global
TRMNLM . . . . . .  L 0014  0044  PC_DATA Global
TRMRPY . . . . . .  L 0014  006C  PC_DATA
TT . . . . . .  L DWORD 0004  PC_DATA Global
TTBGN. . . . . .  E BYTE  0008  PC_DATA Global
TTEND. . . . . .  E BYTE  00A8  PC_DATA Global
VAROP. . . . . .  L 0014  0094  PC_DATA Global
$$DIR. . . . . .  Number  0008
$$NAM. . . . . .  Number  0008
$$NEXT . . . . . .  Number  0004
$$SERV . . . . . .  Number  000A
$ALLNAD. . . . . .  Number  00FF
$ATRNRM. . . . . .  Number  0000
$CALLCA. . . . . .  Number  0023
$CALLCB. . . . . .  Number  001B
```

$CALLDI.	Number	0066
$CALLDS.	Number	0021
$CALLEM.	Number	000A
$CALLKR.	Number	0034
$CALLTK.	Number	001C
$CALLTM.	Number	0067
$CALLTP.	Number	0001
$CLINT	Number	024D
$CONS.	Number	0080
$CONSPT.	Number	0040
$CTLGO	Number	0000
$CTLIS	Number	0004
$CTLRP	Number	0003
$DIINT	Number	024C
$DTAGO	Number	0002
$DTAIS	Number	0004
$DTALC	Alias	$MAXTBD
$DTAND	Number	0000
$DTANL	Number	0012
$DTARP	Alias	$MAXTBD
$ECHO.	Number	0002
$ENDRCV.	Number	0010
$ENINT	Number	024E
$FALSE	Number	0001
$INITT	Number	0020
$INITTA.	Number	0100
INT.	Number	00CD
$LCABT	Number	00FF
$LCAOK	Number	0000
$LCMAXL.	Alias	$MAXTBD
$LCNOP	Alias	$FALSE
$LCOPN	Alias	$TRUE
$MAXDLY.	Number	7FFF
$MAXMSG.	Number	0066
$MAXRTY.	Number	0010
$MAXTBD.	Number	0211
$MAXWTE.	Number	000A
$MND00	Number	0001
$MND01	Number	0009
$MND02	Number	000A
$MND03	Number	000B
$MND04	Number	000C
$MND05	Number	0011
$MND06	Number	0013
$MNT00	Number	0002
$MNT01	Number	0003
$MNT010.	Number	0010
$MNT011.	Number	0012
$MNT012.	Number	0014
$MNT02	Number	0004
$MNT03	Number	0005
$MNT04	Number	0006
$MNT05	Number	0007
$MNT06	Number	0008
$MNT07	Number	000D

```
$MNT00 . . . . . .  Number  000E
$MNT09 . . . . . .  Number  000F
$MTADD . . . . . .  Number  0003
$MTDEL . . . . . .  Number  0000
$NADOK . . . . . .  Number  0040
$NAMELN. . . . . .  Number  000A
$NLANY . . . . . .  Number  0001
$NLBYE . . . . . .  Number  00FF
$NLDNY . . . . . .  Number  00FF
$NLDSK . . . . . .  Number  0001
$NLHI. . . . . .  Number  0000
$NLIDN . . . . . .  Number  0010
$NLWHO . . . . . .  Number  0002
$NLWHR . . . . . .  Number  0003
$ONE . . . . . .  Number  0001
$OPCNE . . . . . .  Number  0000
$OPINL . . . . . .  Number  0001
$OPINP . . . . . .  Number  00FF
$OPTCL . . . . . .  Number  0007
$OPTDF . . . . . .  Number  0004
$OPTNG . . . . . .  Number  0005
$OPTRB . . . . . .  Number  0002
$OPTTF . . . . . .  Number  0006
$OPTTO . . . . . .  Number  0003
$P82EML. . . . . .  Number  0005
$P82ENB. . . . . .  Number  00FA
$P82EOI. . . . . .  Number  0020
$P82OP0. . . . . .  Number  0020
$P82OP1. . . . . .  Number  0021
$P82RIS. . . . . .  Number  000B
$P82TML. . . . . .  Number  0001
$PROID . . . . . .  Number  01FE
$RDRAM . . . . . .  Number  0249
$RDRAMI. . . . . .  Number  024B
$RDSTAT. . . . . .  Number  0248
$RQCSZE. . . . . .  Number  0004
$RQDSE . . . . . .  Number  0080
$RQENT . . . . . .  Number  0001
$RQFUL . . . . . .  Number  00FF
$RQMT. . . . . .  Number  0000
$RQUSEE. . . . . .  Number  0080
$RQUSEI. . . . . .  Number  0010
$RQUSEL. . . . . .  Number  0020
$RQUSER. . . . . .  Number  0040
$SEND. . . . . .  Number  0040
$SETUP . . . . . .  Number  00F0
$SKNUM . . . . . .  Number  0060
$SNDNAD. . . . . .  Number  00CF
$SOKRC . . . . . .  Number  00B0
$SOKRL . . . . . .  Number  00A0
$SOKRM . . . . . .  Number  0080
$SOKTD . . . . . .  Number  00B0
$SOKTM . . . . . .  Number  0080
$STINT . . . . . .  Number  024F
$TACCB . . . . . .  Number  05C6
```

$TADTL	Number	05D6	
$TARR.	Number	05D2	
$TBINIT.	Number	0010	
$TBINTI.	Number	0080	
$TBNOTK.	Number	00DF	
$TBRENB.	Number	0040	
$TBRTCK.	Number	0020	
$TICKMX.	Number	000A	
$TRAPFL.	Number	0001	
$TRCCON.	Number	0002	
$TRCDSK.	Number	0001	
$TRCHRD.	Number	0004	
$TRCL.	Number	0083	
$TRDL.	Number	0081	
$TREA.	Number	00C0	
$TREDY	Number	0080	
$TRENB	Number	0020	
$TRIS.	Number	0084	
$TRIT.	Number	0086	
$TRNG.	Number	0080	
$TRNR.	Number	0085	
$TRNS.	Number	0082	
$TROK.	Number	0000	
$TRPND	Number	0010	
$TRSS.	Number	00FE	
$TRST.	Number	007F	
$TRSU.	Number	00FF	
$TRUE.	Number	0000	
$VIR_DRV1.	Number	0001	
$WHOAMI.	Number	0001	
$WRCTRH.	Number	0248	
$WRCTRL.	Number	024A	
$WRRAM	Number	024B	
$WRSTBE.	Number	0249	
$ZERO.	Number	0000	
??0000	L NEAR	00C1	PC_CODE
??0001	L WORD	00A8	PC_DATA
??0002	L BYTE	00AA	PC_DATA
??0003	E BYTE	00B7	PC_DATA
??0004	L NEAR	00ED	PC_CODE
??0005	L WORD	00B7	PC_DATA
??0006	L BYTE	00B9	PC_DATA
??0007	E BYTE	00C6	PC_DATA
??0008	L NEAR	0132	PC_CODE
??0009	L WORD	00C6	PC_DATA
??000A	L BYTE	00C8	PC_DATA
??000B	E BYTE	00D9	PC_DATA
??000C	L NEAR	0149	PC_CODE
??000D	L WORD	00D9	PC_DATA
??000E	L BYTE	00DB	PC_DATA
??000F	E BYTE	00E8	PC_DATA
??0010	L NEAR	0161	PC_CODE
??0011	L WORD	00E8	PC_DATA
??0012	L BYTE	00EA	PC_DATA
??0013	E BYTE	00F7	PC_DATA

??0014	L NEAR	01ED	PC_CODE
??0015	L WORD	00F7	PC_DATA
??0016	L BYTE	00F9	PC_DATA
??0017	E BYTE	0106	PC_DATA
??0018	L NEAR	0204	PC_CODE
??0019	L WORD	0106	PC_DATA
??001A	L BYTE	0108	PC_DATA
??001B	E BYTE	0115	PC_DATA
??001C	L NEAR	02A6	PC_CODE
??001D	L WORD	0115	PC_DATA
??001E	L BYTE	0117	PC_DATA
??001F	E BYTE	0124	PC_DATA
??0020	L NEAR	034C	PC_CODE
??0021	L WORD	0124	PC_DATA
??0022	L BYTE	0126	PC_DATA
??0023	E BYTE	0133	PC_DATA
??0024	L NEAR	0363	PC_CODE
??0025	L WORD	0133	PC_DATA
??0026	L BYTE	0135	PC_DATA
??0027	E BYTE	0142	PC_DATA
??0028	L NEAR	03A9	PC_CODE
??0029	L WORD	0142	PC_DATA
??002A	L BYTE	0144	PC_DATA
??002B	E BYTE	0151	PC_DATA
??002C	L NEAR	0434	PC_CODE
??002D	L WORD	0151	PC_DATA
??002E	L BYTE	0153	PC_DATA
??002F	E BYTE	0160	PC_DATA
??0030	L NEAR	044B	PC_CODE
??0031	L WORD	0160	PC_DATA
??0032	L BYTE	0162	PC_DATA
??0033	E BYTE	016E	PC_DATA
??0034	L NEAR	04DA	PC_CODE
??0035	L WORD	016E	PC_DATA
??0036	L BYTE	0170	PC_DATA
??0037	E BYTE	017C	PC_DATA
??0038	L NEAR	052A	PC_CODE
??0039	L WORD	017C	PC_DATA
??003A	L BYTE	017E	PC_DATA
??003B	E BYTE	018B	PC_DATA
??003C	L NEAR	056E	PC_CODE
??003D	L WORD	018B	PC_DATA
??003E	L BYTE	018D	PC_DATA
??003F	E BYTE	019A	PC_DATA
??0040	L NEAR	0585	PC_CODE
??0041	L WORD	019A	PC_DATA
??0042	L BYTE	019C	PC_DATA
??0043	E BYTE	01A8	PC_DATA

Warning Errors	Severe Errors
0	0