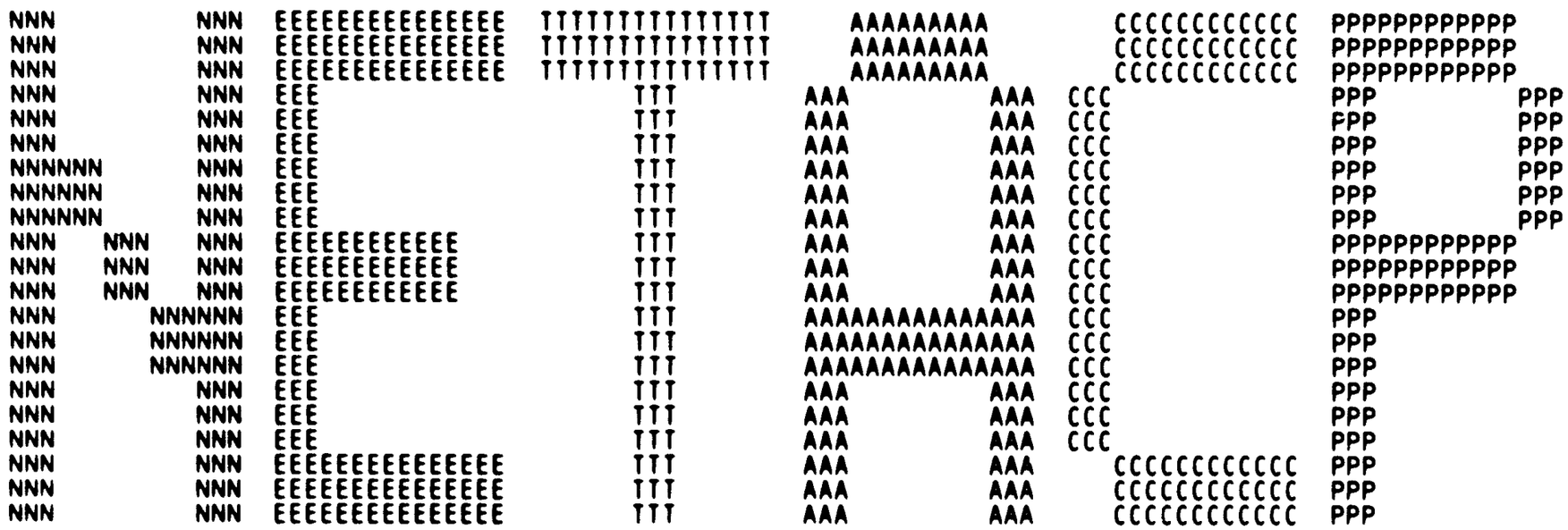
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT AAAAAAAAA CCCCCCCCCCC PPPPPPPPPPPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNNNNN NNN EEE TTT AAA AAA CCC PPP PPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPPP
NNN NNN NNN EEEEEEEEEEE TTT AAA AAA CCC PPPPPPPPPPPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNNNNN EEE TTT AAAAAAAAAAAAAAA CCC PPP
NNN NNN EEE TTT AAA AAA CCC PPP
NNN NNN EEE TTT AAA AAA CCC PPP
NNN NNN EEE TTT AAA AAA CCC PPP
NNN NNN EEEEEEEEEEEEEEE TTT AAA AAA CCCCCCCCCCC PPP
NNN NNN EEEEEEEEEEEEEEE TTT AAA AAA CCCCCCCCCCC PPP
NNN NNN EEEEEEEEEEEEEEE TTT AAA AAA CCCCCCCCCCC PPP

```
NN       NN  DDDDDDDD    DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
NN       NN  DDDDDDDD    DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
NN       NN  DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
NN       NN  DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
NNNN     NN  DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
NNNN     NN  DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
NN  NN   NN  DD      DD  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
NN  NN   NN  DD      DD  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
NN    NNNN  DD      DD  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
NN    NNNN  DD      DD  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
NN      NN  DD      DD  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
NN      NN  DD      DD  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
NN      NN  DDDDDDDD    DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....
NN      NN  DDDDDDDD    DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
0000     1          .TITLE  NDDRIVER - DECnet DLE driver
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;*****************************************************************************
0000     5  ;*                                                                           *
0000     6  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8  ;*  ALL RIGHTS RESERVED.                                                     *
0000     9  ;*                                                                           *
0000    10  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    11  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    12  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    13  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    14  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    15  ;*  TRANSFERRED.                                                             *
0000    16  ;*                                                                           *
0000    17  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    18  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    19  ;*  CORPORATION.                                                             *
0000    20  ;*                                                                           *
0000    21  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    22  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23  ;*                                                                           *
0000    24  ;*                                                                           *
0000    25  ;*****************************************************************************
0000    26
0000    27  ;++
0000    28  ; FACILITY:     DECnet-VAX
0000    29  ;
0000    30  ; ABSTRACT:
0000    31  ;
0000    32  ;       This driver implements the DLE facility in DECnet-VAX which
0000    33  ;       allows programs direct access to a DECnet circuit.  This is
0000    34  ;       primarily used to implement MOP support.
0000    35  ;
0000    36  ; ENVIRONMENT:
0000    37  ;
0000    38  ;       MODE = KERNEL
0000    39  ;
0000    40  ; AUTHOR:
0000    41  ;
0000    42  ;       Tim Halvorsen, December 1982
0000    43  ;
0000    44  ; MODIFIED BY:
0000    45  ;
0000    46  ;       V03-005 LMP0275         L. Mark Pilant,         12-Jul-1984  19:48
0000    47  ;               Initialize the ACL info in the ORB to be a null descriptor
0000    48  ;               list rather than an empty queue.  This avoids the overhead
0000    49  ;               of locking and unlocking the ACL mutex, only to find out
0000    50  ;               that the ACL was empty.
0000    51  ;
0000    52  ;       V03-004 LMP0221         L. Mark Pilant,         30-Mar-1984  11:36
0000    53  ;               Change UCB$L_OWNUIC to ORB$L_OWNER and UCB$W_VPROT to
0000    54  ;               ORB$W_PROT.
0000    55  ;
0000    56  ;       V003    RNG0003         Rod Gamache     28-Jun-1983
0000    57  ;               Remove internal definition for IRP$Q_STATION.
```

```
0000     58 ;
0000     59 ;       V002    TMH0002         Tim Halvorsen   08-May-1983
0000     60 ;               Fix case where user receive was not being posted if the
0000     61 ;               LPD was shutdown while a DLE receive was outstanding.
0000     62 ;
0000     63 ;       V001    TMH0001         Tim Halvorsen   20-Apr-1983
0000     64 ;               Fix cancel routine to not abort pending user I/O, but to
0000     65 ;               instead wait for the datalink driver to return the datalink
0000     66 ;               IRPs before aborting user requests.
0000     67 ;               Add bugcheck which checks for the condition of receiving a
0000     68 ;               message from the datalink without any pending user receive
0000     69 ;               outstanding.
0000     70 ;               Insert DWBs onto global DWB list in order by NETACP channel
0000     71 ;               number, so that it can be used as the collating sequence.
0000     72 ;--
```

```
                        0000     74          .SBTTL  DECLARATIONS
                        0000     75 ;
                        0000     76 ; VMS definitions
                        0000     77 ;
                        0000     78          $ABDDEF
                        0000     79          $AQBDEF
                        0000     80          $CANDEF
                        0000     81          $CCBDEF
                        0000     82          $CRBDEF
                        0000     83          $CXBDEF
                        0000     84          $DDBDEF
                        0000     85          $DEVDEF
                        0000     86          $DDTDEF
                        0000     87          $DYNDEF
                        0000     88          $IODEF
                        0000     89          $IPLDEF
                        0000     90          $IRPDEF
                        0000     91          $JIBDEF
                        0000     92          $ORBDEF
                        0000     93          $PCBDEF
                        0000     94          $PRDEF
                        0000     95          $PRVDEF
                        0000     96          $RSNDEF
                        0000     97          $SSDEF
                        0000     98          $UCBDEF
                        0000     99          $VADEF
                        0000    100          $VCBDEF
                        0000    101          $VECDEF
                        0000    102
                        0000    103 ;
                        0000    104 ; Network definitions
                        0000    105 ;
                        0000    106
                        0000    107          $DWBDEF                 ; DLE window control block
                        0000    108          $NETSYMDEF              ; Get NET$C_IPL symbol
                        0000    109          $RCBDEF                 ; Get RCB$W_MCOUNT symbol
                        0000    110
                        0000    111
                        0000    112 ;
                        0000    113 ; Definitions that follow the standard UCB fields
                        0000    114 ;
                        0000    115
                        0000    116          $DEFINI UCB                         ; Start of UCB definitions
                        0000    117
00000090                0000    118 . = UCB$C_LENGTH                             ; Position at end of standard UCB
                        0090    119
                        0090    120 $DEF     UCB$Q_DWB_LIST  .BLKQ   1           ; "Global" DWB listhead
                        0098    121 $DEF     UCB$W_NEXT_ID   .BLKW   1           ; Next unique identifier to allocate
                        009A    122 $DEF     UCB$C_ND_LENGTH                     ; Size of our UCB
                        009A    123
                        009A    124          $DEFEND UCB
                        0000    125
                        0000    126 ;
                        0000    127 ; Overlays of IRP
                        0000    128 ;
                        0000    129
                        0000    130          $DEFINI IRP
```

```
                     0000   131
0000005C             0000   132 . = IRP$L_ARB + 4
                     005C   133 $DEF     IRP$C_STDLEN                    ; End of standard IRP
                     005C   134
                     005C   135         $DEFEND IRP
                     0000   136
                     0000   137 ;
                     0000   138 ; Define maximum allowable length of a received message for datalinks
                     0000   139 ; which do direct I/O on receives.
                     0000   140 ;
                     0000   141
00000800             0000   142         MAX_DIR_RCV = 2048       ; Define max receive for direct I/O
                     0000   143
                     0000   144 ;
                     0000   145 ; Argument list offsets for FDT routines
                     0000   146 ;
                     0000   147
00000000             0000   148 P1       = 0
00000004             0000   149 P2       = 4
00000008             0000   150 P3       = 8
0000000C             0000   151 P4       = 12
00000010             0000   152 P5       = 16
00000014             0000   153 P6       = 20
                     0000   154
                     0000   155 ;
                     0000   156 ; Define format of second longword of IOSB returned in ACCESS and SETMODE
                     0000   157 ; functions to the DLE user.
                     0000   158 ;
                     0000   159
                     0000   160         $DEFINI STAT
                     0000   161
                     0000   162         _VIELD  STAT,0,<-
                     0000   163                 <BC,,M>,-               ; Circuit is a "broadcast circuit"
                     0000   164                 >
                     0000   165
                     0000   166         $DEFEND STAT
                     0000   167
                     0000   168 ;
                     0000   169 ; Define format of NI diagnostics buffer
                     0000   170 ;
                     0000   171
                     0000   172         $DEFINI DIAG
                     0000   173
                     0000   174 $DEF     DIAG_L_DATA     .BLKL   1               ; Address of start of data
                     0004   175 $DEF     DIAG_L_USERBUF  .BLKL   1               ; Address of user buffer
                     0008   176 $DEF     DIAG_W_SIZE     .BLKW   1               ; Size of structure
                     000A   177 $DEF     DIAG_B_TYPE     .BLKB   1               ; Type of structure
0000000C             000B   178                         .BLKB   1
                     000C   179 $EQU     DIAG_C_NIHDRSIZ         14
                     000C   180 $DEF     DIAG_G_DATA     .BLKB   DIAG_C_NIHDRSIZ ; 14 bytes of NI datalink header
                     001A   181 $DEF     DIAG_C_LENGTH                           ; Length of structure
                     001A   182
                     001A   183         $DEFEND DIAG
```

```
                        .SBTTL  DRIVER PROLOGUE TABLE

                ;
                ; DRIVER PROLOGUE TABLE
                ;

                        .PSECT  $$$105_PROLOGUE
                DPTAB   -                                       ; Define driver prologue table
                        END     = DLE$END,-                     ; End of driver
                        ADAPTER = NULL,-                        ; Adapter type
                        UCBSIZE = UCB$C_ND_LENGTH,-             ; UCB size
                        NAME    = NDDRIVER                      ; Driver name

                        DPT_STORE INIT                          ; CONTROL BLOCK INIT VALUES
                                                                ;
                        DPT_STORE CRB,CRB$L_INTD+VEC$L_ADP,L,0  ; No ADP pointer
                        DPT_STORE UCB,UCB$B_FIPL,B,NET$C_IPL    ; Fork IPL
                        DPT_STORE UCB,UCB$B_DIPL,B,NET$C_IPL    ; Device IPL
                        DPT_STORE ORB,ORB$B_FLAGS,B,-
                                                <ORB$M_PROT_16> ; SOGW protection word
                        DPT_STORE ORB,ORB$W_PROT,W,0            ; Default protection
                        DPT_STORE ORB,ORB$L_OWNER,L,<^X010001>  ; Owner UIC
                        DPT_STORE UCB,UCB$L_DEVCHAR,L,-         ; Device characteristics
                                                <DEV$M_NET!-    ;   Network device
                                                 DEV$M_AVL!-    ;   Available
                                                 DEV$M_SHR!-    ;   Shared by multiple users
                                                 DEV$M_MBX!-    ;   Mailbox type (no hardware)
                                                 DEV$M_IDV!-    ;   Input device
                                                 DEV$M_ODV-     ;   Output device
                                                >               ;
                        DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,0       ;
                        DPT_STORE UCB,UCB$W_STS,W,UCB$M_ONLINE  ; Device online
                        DPT_STORE UCB,UCB$W_NEXT_ID,W,0         ; Initialize unique ID number
                                                                ;
                        DPT_STORE REINIT                        ; CONTROL BLOCK RE-INIT VALUES

                        DPT_STORE DDB,DDB$L_DDT,D,ND$DDT
                        DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,D,UNIT_INIT ; Unit Initialization
                        DPT_STORE END
```

```
                        .SBTTL  DRIVER DISPATCH TABLE

                ;
                ; DRIVER DISPATCH TABLE
                ;
                        .PSECT  $$$115_DRIVER,LONG

                        DDTAB   DEVNAM  = ND,-          ; DRIVER DISPATCH TABLE
                                FUNCTB  = FUNCTABLE,-   ; Function decision table address
                                START   = DLE$STARTIO,- ; Start I/O operation
                                CANCEL  = DLE$CANCEL,-  ; Cancel I/O entry point
                                UNSOLIC = DLE$LPD_DOWN  ; "LPD down" entry point
                                                        ;   (called by NETACP only)


                        .SBTTL  FUNCTION DECISION TABLE

                FUNCTABLE:                              ; FUNCTION DECISION TABLE
                        FUNCTAB ,-                      ; Legal Functions
                                <READVBLK,READLBLK,-    ; Read
                                 WRITEVBLK,WRITELBLK,-  ; Write
                                 SETMODE,-              ; Set mailbox message filters
                                 ACCESS,-               ; Logical-link Connect/Reject
                                 ACPCONTROL,-           ; ACP Control function
                                 DEACCESS,-             ; Logical-link Disconnect
                        >
                        FUNCTAB ,-                      ; BUFFERED I/O FUNCTIONS
                                <READVBLK,READLBLK,-    ; Read
                                 WRITEVBLK,WRITELBLK,-  ; Write
                                 SETMODE,-              ; Set mailbox message filters
                                 ACCESS,-               ; Logical-link Connect/Reject
                                 ACPCONTROL,-           ; ACP Control function
                                 DEACCESS,-             ; Logical-link Disconnect
                        >
                        FUNCTAB DLE$FDT_RCV,     <READLBLK>   ; Read
                        FUNCTAB DLE$FDT_XMT,     <WRITELBLK>  ; Write
                        FUNCTAB DLE$FDT_ACCESS,  <ACCESS>     ; Connect Logical-link
                        FUNCTAB DLE$FDT_DEACCESS,<DEACCESS>   ; Disconnect Logical-link
                        FUNCTAB DLE$FDT_SETMODE, <SETMODE>    ; Set mailbox message filters
                        FUNCTAB DLE$FDT_CONTROL, <ACPCONTROL> ; ACP Control
```

```
                     0090   266          .SBTTL  DLE$STARTIO     - Start I/O operation
                     0090   267 ;+
                     0090   268 ;  DLE$STARTIO - START I/O OPERATION
                     0090   269 ;
                     0090   270 ;  This routine is entered when the associated unit is idle and a packet
                     0090   271 ;  is available for processing.  The IRP$L_WIND field is used to locate the
                     0090   272 ;  associated window block.
                     0090   273 ;
                     0090   274 ; Inputs:
                     0090   275 ;
                     0090   276 ;       R5 = UCB address
                     0090   277 ;       R3 = IRP address
                     0090   278 ;
                     0090   279 ;       IPL = FIPL
                     0090   280 ;
                     0090   281 ; Outputs:
                     0090   282 ;
                     0090   283 ;       None
                     0090   284 ;
                     0090   285 ;       R0-R2,R4 are destroyed.
                     0090   286 ;-
                     0090   287 DLE$STARTIO::                                ; Process next IRP
         07E0 8F     BB  0090   288          PUSHR   #^M<R5,R6,R7,R8,R9,R10> ; Save registers
              26     10  0094   289          BSBB    PROC_IO                 ; Process the I/O function
         07E0 8F     BA  0096   290          POPR    #^M<R5,R6,R7,R8,R9,R10> ; Restore registers
    53     58 A5     D0  009A   291          MOVL    UCB$L_IRP(R5),R3        ; Get IRP
              0A     13  009E   292          BEQL    50$                     ; If EQL then its been queued
                         00A0   293                                          ; or suspended, start next I/O
    50     38 A3     7D  00A0   294          MOVQ    IRP$L_IOST1(R3),R0      ; Get IOSB
00000000'GF     17     00A4   295          JMP     G^IOC$REQCOM            ; Complete I/O & start next I/O
                         00AA   296          ;
                         00AA   297          ;   Start next I/O without completing current IRP
                         00AA   298          ;
    53     4C B5     0F  00AA   299 50$:     REMQUE  @UCB$L_IOQFL(R5),R3     ; Get next IRP
              06     1D  00AE   300          BVS     60$                     ; If VS then none
00000000'GF     17     00B0   301          JMP     G^IOC$INITIATE          ; Call Exec to deliver IRP to driver
                         00B6   302 60$:     CLRBIT  #UCB$V_BSY,UCB$W_STS(R5); Allow further IRPs to be delivered
                   05    00BB   303          RSB                             ; Return to Exec
                         00BC   304
                         00BC   305 PROC_IO:
                         00BC   306          ;
                         00BC   307          ;   Get the DWB address (if any) and dispatch on function code with:
                         00BC   308          ;
                         00BC   309          ;       R6 = DWB address (may be zero)
                         00BC   310          ;       R5 = UCB address
                         00BC   311          ;       R3 = IRP address
                         00BC   312          ;
    56     18 A3     D0  00BC   313          MOVL    IRP$L_WIND(R3),R6       ; Get DWB, if any
              02     19  00C0   314          BLSS    10$                     ; If LSS, DWB is in system space
              56     D4  00C2   315          CLRL    R6                      ; Else, Invalidate window pointer
FFFFFFC0 8F     CB  00C4   316 10$:     BICL3   #^C<IO$M_FCODE>,-           ; Get function code without
    57     20 A3         00CA   317                  IRP$W_FUNC(R3),R7       ; modifier bits
                         00CD   318          $DISPATCH  R7,TYPE=B,-          ; Process I/O
                         00CD   319          <-
                         00CD   320            <IO$_ACCESS,     DLE$ACCESS>,-      ; Connect Requests
                         00CD   321            <IO$_DEACCESS,   DLE$DEACCESS>,-    ; Disconnect Requests
                         00CD   322            <IO$_SETMODE,    DLE$SETMODE>,-     ; SetMode Requests
```

```
                00CD   323          <IO$_ACPCONTROL, DLE$CONTROL>,-      ; ACP Control function
                00CD   324          >                                   ; Else, fall thru
00F4 8F    3C   00FD   325          MOVZWL  #SS$_ILLIOFUNC,-            ; Assume fct not supported
  38 A3         0101   326                  IRP$L_IOST1(R3)             ; Set error code
               05 0103   327          RSB                                 ; Return if unknown
```

```
                     0104   329          .SBTTL  DLE$FDT_ACCESS  - IO$_ACCESS  FDT processing
                     0104   330          .SBTTL  DLE$ACCESS      - IO$_ACCESS "startio" processing
                     0104   331
                     0104   332 ;++
                     0104   333 ;  DLE$FDT_ACCESS - IO$_ACCESS FDT processing
                     0104   334 ;  DLE$ACCESS     - Common connect "startio" processing
                     0104   335 ;
                     0104   336 ;  DLE$FDT_ACCESS passes the IRP through the EXEC, where the user parameters
                     0104   337 ;  are packaged into a "complex buffer", to the ACP.  The ACP processes the
                     0104   338 ;  connect request and puts the circuit into MOP mode, if necessary.
                     0104   339 ;
                     0104   340 ;  DLE$ACCESS gets control when the circuit is ready, so that the access
                     0104   341 ;  function can be completed.
                     0104   342 ;
                     0104   343 ;  It should be noted that the size of the DWB is not charged against the user
                     0104   344 ;  byte count or byte limit quotas.  It is assumed that these quotas are at
                     0104   345 ;  least partly used to limit a run away process and that the file quota of a
                     0104   346 ;  process, which is charged, is a sufficient mechanism.
                     0104   347 ;
                     0104   348 ; Inputs:
                     0104   349 ;
                     0104   350 ;       R6 = CCB address
                     0104   351 ;       R5 = UCB address
                     0104   352 ;       R4 = PCB address
                     0104   353 ;       R3 = IRP address
                     0104   354 ;
                     0104   355 ;       P1 = Circuit name desired for DLE
                     0104   356 ;
                     0104   357 ; Outputs:
                     0104   358 ;
                     0104   359 ;       None
                     0104   360 ;-
                     0104   361 NOOPER:
 50   2894 8F    3C  0104   362          MOVZWL  #SS$_NOOPER,R0          ; "operator privilege required"
            05    11  0109   363          BRB     ABORTIO
                     0108   364 DEVNOTMNT:
 50   007C 8F    3C  0108   365          MOVZWL  #SS$_DEVNOTMOUNT,R0     ; "device not mounted"
                     0110   366 ABORTIO:
  00000000'GF    17  0110   367          JMP     G^EXE$ABORTIO           ; Return to abort I/O
                     0116   368
                     0116   369 DLE$FDT_ACCESS::                         ; IO$_ACCESS  "FDT" processing
                     0116   370          ;
                     0116   371          ;   OPER privilege is required to issue a DLE IO$_ACCESS
                     0116   372          ;
                     0116   373          IFNPRIV OPER,NOOPER             ; Error if user doesn't have OPER
                     011C   374          ;
                     011C   375          ;   Check if the ACP is in "dismounting" state, or if it's even
                     011C   376          ;   mounted at all.
                     011C   377          ;
EA 38 A5    13   E1  011C   378          BBC     #DEV$V_MNT,UCB$L_DEVCHAR(R5),DEVNOTMNT ; If not mounted, error
E5 38 A5    15   E0  0121   379          BBS     #DEV$V_DMT,UCB$L_DEVCHAR(R5),DEVNOTMNT ; If dismounting, error
                     0126   380          ;
                     0126   381          ;   Allocate DWB for a window control block
                     0126   382          ;
   51    50 8F    9A  0126   383          MOVZBL  #DWB$C_LENGTH,R1        ; Set block length
           078B    30  012A   384          BSBW    DLE$ALONPGD_Z           ; Allocate/zero non-paged pool
         E0 50     E9  012D   385          BLBC    R0,ABORTIO              ; Exit if error detected
```

```
                                                H 8
4C A3    52    D0  0130   386          MOVL    R2,IRP$L_DIAGBUF(R3)    ; Save DWB address
0A A2    13    90  0134   387          MOVB    #DYN$C_BUFIO,DWB$B_TYPE(R2) ; Setup block type
                   0138   388          ;
                   0138   389          ;   Initialize the DWB
                   0138   390          ;
50   28 A5    D0  0138   391          MOVL    UCB$L_DDB(R5),R0        ; Get address of DDB
50   04 A0    D0  013C   392          MOVL    DDB$L_UCB(R0),R0        ; Get address of ND's UCB0
38 A2    50    D0  0140   393          MOVL    R0,DWB$L_UCB0(R2)       ; Save address of ND's UCB0
   0098 C0    B6  0144   394          INCW    UCB$W_NEXT_ID(R0)       ; Allocate another unique ID number
   0098 C0    B0  0148   395          MOVW    UCB$W_NEXT_ID(R0),-     ; Move unique ID to DWB
     4E A2              014C   396                  DWB$W_ID(R2)
50   14 A2    9E  014E   397          MOVAB   DWB$Q_RCV_MSG(R2),R0    ; Get address of received msg queue
   60   50    D0  0152   398          MOVL    R0,(R0)                 ; Init. listhead
   60   80    DE  0155   399          MOVAL   (R0)+,(R0)
50   1C A2    9E  0158   400          MOVAB   DWB$Q_USER_RCV(R2),R0   ; Get address of user receive queue
   60   50    D0  015C   401          MOVL    R0,(R0)                 ; Init. listhead
   60   80    DE  015F   402          MOVAL   (R0)+,(R0)
50   24 A2    9E  0162   403          MOVAB   DWB$Q_USER_XMT(R2),R0   ; Get address of user transmit queue
   60   50    D0  0166   404          MOVL    R0,(R0)                 ; Init. listhead
   60   80    DE  0169   405          MOVAL   (R0)+,(R0)
50   2C A2    9E  016C   406          MOVAB   DWB$Q_XMT_PND(R2),R0    ; Get address of transmits pending queue
   60   50    D0  0170   407          MOVL    R0,(R0)                 ; Init. listhead
   60   80    DE  0173   408          MOVAL   (R0)+,(R0)
     0C A2    B6  0176   409          INCW    DWB$W_REFCNT(R2)        ; Increment reference count
     0C A3    D0  0179   410          MOVL    IRP$L_PID(R3),-         ; Save PID of accessor
     34 A2              017C   411                  DWB$L_PID(R2)
     28 A3    B0  017E   412          MOVW    IRP$W_CHAN(R3),-        ; Save channel of accessor
     3C A2              0181   413                  DWB$W_CHAN(R2)
                   0183   414          ;
                   0183   415          ;   Insert DWB onto "global" DWB list in our UCB0, so that we
                   0183   416          ;   can keep track of all DWBs for all DLE users.  Make sure
                   0183   417          ;   we keep the DWBs in order by DLL_CHAN.
                   0183   418          ;
  50   38 A2    D0  0183   419          MOVL    DWB$L_UCB0(R2),R0       ; Get DLE UCB0 address
51   0090 C0    9E  0187   420          MOVAB   UCB$Q_DWB_LIST(R0),R1   ; Get address of listhead
     50   51    D0  018C   421          MOVL    R1,R0                   ; Setup for loop
     50   60    D0  018F   422 10$:     MOVL    (R0),R0                 ; Get next entry
     51   50    D1  0192   423          CMPL    R0,R1                   ; End of list?
          07    13  0195   424          BEQL    15$                     ; If so, insert it
        4E A0    B1  0197   425          CMPW    DWB$W_ID(R0),-          ; Are we at the right place?
        4E A2              019A   426                  DWB$W_ID(R2)
          F1    1F  019C   427          BLSSU   10$                     ; If not yet, keep going
  04 B0    62    0E  019E   428 15$:     INSQUE  (R2),@4(R0)             ; Insert into global DWB list
                   01A2   429          ;
                   01A2   430          ;   Increment ACP mount count, so the ACP doesn't go away until
                   01A2   431          ;   we complete this DWB.
                   01A2   432          ;
  50   34 A5    D0  01A2   433          MOVL    UCB$L_VCB(R5),R0        ; Get VCB address
       54 A0    B6  01A6   434          INCW    RCB$W_MCOUNT(R0)        ; Increment ACP mount count
                   01A9   435          ;
                   01A9   436          ;   Send the request to NETACP for further processing.  NETACP will
                   01A9   437          ;   setup the circuit for DLE access, if necessary, and then return
                   01A9   438          ;   the IRP back here again.
                   01A9   439          ;
                   01A9   440          SETBIT  #IRP$V_PHYSIO,-         ; Mark this as a DLE IRP
                   01A9   441                  IRP$W_STS(R3)
00000000'GF    17  01AE   442          JMP     G^ACP$ACCESSNET         ; Continue in EXEC
```

```
                     01B4    443
                     01B4    444 ;
                     01B4    445 ; We arrive here after the ACP has finished with the IRP, and has requeued
                     01B4    446 ; it to this driver for further processing.
                     01B4    447 ;
                     01B4    448 ; Inputs:
                     01B4    449 ;
                     01B4    450 ;       R5 = UCB address
                     01B4    451 ;       R3 = IRP address
                     01B4    452 ;
                     01B4    453 ;       IRP$L_DIAGBUF = DWB address
                     01B4    454 ;       IRP$L_EXTEND = Address of CXB holding initial unsolicited MOP message
                     01B4    455 ;
                     01B4    456 ;               CXB$W_LENGTH    = Message length in bytes (not incl. NI header)
                     01B4    457 ;               CXB$C_HEADER    = 14-byte NI datalink header
                     01B4    458 ;               CXB$C_HEADER+14 = Message
                     01B4    459 ;
                     01B4    460 ; Outputs:
                     01B4    461 ;
                     01B4    462 ;       None
                     01B4    463 ;
                     01B4    464
                     01B4    465 DLE$ACCESS::                                ; IO$_ACCESS "startio" processing
          008C   30  01B4    466         BSBW    GET_WNDSC                   ; Get CCB$L_WIND image descriptor
            67   D4  01B7    467         CLRL    (R7)                        ; Init CCB$L_WIND image
                     01B9    468         SETBIT  #IRP$V_FUNC,IRP$W_STS(R3)   ; Mark for write back
    32 A3    01   B0  01BE    469         MOVW    #1,IRP$W_BCNT(R3)           ; Write back one descriptor
    56   4C A3   D0  01C2    470         MOVL    IRP$L_DIAGBUF(R3),R6        ; Get DWB address
         4C A3   D4  01C6    471         CLRL    IRP$L_DIAGBUF(R3)           ; Clear pointer in IRP to DWB
      6A 38 A3   E9  01C9    472         BLBC    IRP$L_IOST1(R3),80$         ; If ACP returned with error, exit
                     01CD    473         ;
                     01CD    474         ;   Successful access
                     01CD    475         ;
                     01CD    476         SETBIT  #DWB$V_RUN,-                ; Mark DWB in "run" state
                     01CD    477                 DWB$W_FLAGS(R6)
            67   56  D0  01D2    478         MOVL    R6,(R7)                     ; Setup CCB$L_WIND value
                     01D5    479         ;
                     01D5    480         ;   Setup the flags which tell whether the datalink does
                     01D5    481         ;   buffered or direct transmits/receives.
                     01D5    482         ;
       50    48 A6   D0  01D5    483         MOVL    DWB$L_DLL_UCB(R6),R0        ; Get UCB address
     50    0088 C0   D0  01D9    484         MOVL    UCB$L_DDT(R0),R0            ; Get DDT address
       50    08 A0   D0  01DE    485         MOVL    DDT$L_FDT(R0),R0            ; Get FDT address
    05 08 A0    21   E1  01E2    486         BBC     #IO$_READLBLK,8(R0),20$     ; If driver does buffered receives
                     01E7    487         SETBIT  #DWB$V_DLL_RBF,DWB$W_FLAGS(R6) ; then set flag for "buffered"
    05 08 A0    20   E1  01EC    488 20$:    BBC     #IO$_WRITELBLK,8(R0),30$    ; If driver does buffered transmits
                     01F1    489         SETBIT  #DWB$V_DLL_XBF,DWB$W_FLAGS(R6) ; then set flag for "buffered"
                     01F6    490 30$:    ;
                     01F6    491         ;   If NETACP returned the initial MOP message for this DLE user,
                     01F6    492         ;   then set it up as if we just received the message.
                     01F6    493         ;
            55   53   D0  01F6    494         MOVL    R3,R5                       ; Save address of ACCESS IRP
       52   54 A5   D0  01F9    495         MOVL    IRP$L_EXTEND(R5),R2         ; Get CXB address
              2A    13  01FD    496         BEQL    40$                         ; Skip if none
            0544    30  01FF    497         BSBW    INIT_RCV_IRP                ; Initialize datalink receive IRP
          27 50     E9  0202    498         BLBC    R0,90$                      ; Branch if error detected
       24 A3   52   D0  0205    499         MOVL    R2,IRP$L_IOSB(R3)           ; Attach CXB to datalink IRP
```

```
                     07   E1  0209   500          BBC     #IRP$V_DIAGBUF,-        ; Br if not diagnostic buffer
                   0E 2A A3      020B   501                  IRP$W_STS(R3),35$
                50   4C A3   D0  020E   502          MOVL    IRP$L_DIAGBUF(R3),R0    ; Get address of diag buffer
                        2C   BB  0212   503          PUSHR   #^M<R2,R3,R5>           ; Save registers
                        0E   28  0214   504          MOVC    #DIAG_C_NIHDRSIZ,-      ; Copy NI header to diag buffer
                     48 A2       0216   505                  CXB$C_HEADER(R2),-
                     00 B0       0218   506                  @DIAG_L_DATA(R0)
                        2C   BA  021A   507          POPR    #^M<R2,R3,R5>           ; Restore registers
                        01   B0  021C   508 35$:     MOVW    S^#SS$_NORMAL,-         ; Set successful receive operation
                     38 A3       021E   509                  IRP$L_IOST1(R3)
                     0C A2   B0  0220   510          MOVW    CXB$W_LENGTH(R2),-      ; Set size of message in IOSB
                     3A A3       0223   511                  IRP$L_IOST1+2(R3)
                18 B6    63   0E  0225   512          INSQUE  (R3),@DWB$Q_RCV_MSG+4(R6) ; Insert message on receive queue
                             0229   513          ;
                             0229   514          ;   Queue initial receive to datalink driver
                             0229   515          ;
                   50    01   3C  0229   516 40$:     MOVZWL  S^#SS$_NORMAL,R0        ; Setup I/O status
                      01DE   30  022C   517 90$:     BSBW    GET_STATUS              ; Setup second IOSB longword
                   53    55   D0  022F   518          MOVL    R5,R3                   ; Restore ACCESS IRP address
                38 A3    50   7D  0232   519          MOVQ    R0,IRP$L_IOST1(R3)      ; Store status in IRP
                             05  0236   520          RSB                             ; Return to post ACCESS I/O
                             0237   521
                             0237   522 ;
                             0237   523 ;  Unsuccessful access
                             0237   524 ;
                             0237   525 80$:     SETBIT  #DWB$V_DELETE,-         ; Mark DWB to be deleted
                             0237   526                  DWB$W_FLAGS(R6)
                      007F   30  023C   527          BSBW    DEALLOC_DWB             ; Deallocate DWB, if possible
                      00C6   30  023F   528          BSBW    RESTORE_QUOTA           ; Restore quota
                             05  0242   529          RSB                             ; On return, complete I/O
```

```
                     0243   531          .SBTTL  Get ABD descriptors
                     0243   532 ;+
                     0243   533 ; These routines return descriptors of the various QIO parameters from
                     0243   534 ; the ABD.
                     0243   535 ;
                     0243   536 ; Inputs:
                     0243   537 ;
                     0243   538 ;       R3 = IRP
                     0243   539 ;
                     0243   540 ; Outputs:
                     0243   541 ;
                     0243   542 ;       R7 = Address of actual data field in ABD
                     0243   543 ;       R8 = Size of data field
                     0243   544 ;-
                     0243   545          .ENABL  LSB
                     0243   546
                     0243   547 GET_WNDSC:                              ; Get window descriptor
          58    D4   0243   548          CLRL    R8                     ; Get descriptor offset
          12    11   0245   549          BRB     10$                    ; Continue
                     0247   550 GET_P1DSC:                              ; Get P1 descriptor
     58   08    D0   0247   551          MOVL    #8,R8                  ; Get descriptor offset
          0D    11   024A   552          BRB     10$                    ; Continue
                     024C   553 GET_P2DSC:                              ; Get P2 descriptor
     58   10    D0   024C   554          MOVL    #8*2,R8                ; Get descriptor offset
          08    11   024F   555          BRB     10$                    ; Continue in common
                     0251   556 GET_P3DSC:                              ; Get P3 descriptor
     58   18    D0   0251   557          MOVL    #8*3,R8                ; Get descriptor offset
          03    11   0254   558          BRB     10$                    ; Continue in common
                     0256   559 GET_P4DSC:                              ; Get P4 descriptor
          58   20    D0   0256   560          MOVL    #8*4,R8           ; Get desciptor offset
       58    2C B3    C0   0259   561 10$:     ADDL    @IRP$L_SVAPTE(R3),R8   ; Get descriptor address
          57   88    3C   025D   562          MOVZWL  (R8)+,R7          ; Get offset to data
     57   FF A847    9E   0260   563          MOVAB   -1(R8)[R7],R7     ; Get ptr to data after skipping
                     0265   564                                         ; over access mode byte
          58   68    3C   0265   565          MOVZWL  (R8),R8           ; Get length of data
                     05   0268   566          RSB
                     0269   567
                     0269   568          .DSABL  LSB
```

```
                     0269   570          .SBTTL  DLE$FDT_DEACCESS- IO$_DEACCESS FDT processing
                     0269   571          .SBTTL  DLE$DEACCESS    - IO$_DEACCESS "startio" processing
                     0269   572 ;+
                     0269   573 ; DLE$FDT_DEACCESS - User QIO request to disassociate DWB with the I/O channel
                     0269   574 ;
                     0269   575 ; Inputs:
                     0269   576 ;
                     0269   577 ;       R6 = CCB address
                     0269   578 ;       R5 = UCB address
                     0269   579 ;       R4 = PCB address
                     0269   580 ;       R3 = IRP address
                     0269   581 ;
                     0269   582 ; Outputs:
                     0269   583 ;
                     0269   584 ;       None
                     0269   585 ;-
                     0269   586 DLE$FDT_DEACCESS::                          ; IO$_DEACCESS FDT routine
  18 A3    01    CA  0269   587          BICL    #1,IRP$L_WIND(R3)           ; Clear interlock bit
           08    19  026D   588          BLSS    10$                         ; If LSS then link is accessed
50    00AC 8F    3C  026F   589          MOVZWL  #SS$_FILNOTACC,R0           ; Say "link not accessed"
      FE99       31  0274   590          BRW     ABORTIO                     ; Abort the I/O
                     0277   591 10$:     ;
                     0277   592          ;   Abort all outstanding I/O on the channel
                     0277   593          ;
      07F8 8F    BB  0277   594          PUSHR   #^M<R3,R4,R5,R6,R7,R8,R9,R10>
                     027B   595          DSBINT  UCB$B_FIPL(R5)              ; Synchronize
  56   18 A3    D0  0282   596          MOVL    IRP$L_WIND(R3),R6           ; Get DWB address
        01BA    30  0286   597          BSBW    CANCEL_ALL                  ; Cancel all outstanding I/O
                     0289   598          ENBINT                              ; Restore IPL
      07F8 8F    BA  028C   599          POPR    #^M<R3,R4,R5,R6,R7,R8,R9,R10>
                     0290   600          ;
                     0290   601          ;   Send the request to NETACP for further processing
                     0290   602          ;
                     0290   603          SETBIT  #IRP$V_PHYSIO,-             ; Mark this as a DLE IRP
                     0290   604                  IRP$W_STS(R3)
00000000'GF    17    0295   605          JMP     G^ACP$DEACCESS              ; Goto common IO$_DEACCESS FDT routine
                     029B   606
                     029B   607 ;
                     029B   608 ; We arrive here after the ACP has finished with the IRP, and has requeued
                     029B   609 ; it to this driver for further processing.
                     029B   610 ;
                     029B   611 ; Inputs:
                     029B   612 ;
                     029B   613 ;       R6 = DWB address
                     029B   614 ;       R5 = UCB address
                     029B   615 ;       R3 = IRP address
                     029B   616 ;
                     029B   617 ; Outputs:
                     029B   618 ;
                     029B   619 ;       None
                     029B   620 ;
                     029B   621
                     029B   622 DLE$DEACCESS::                              ; IO$_DEACCESS "startio" processing
        FFA5    30   029B   623          BSBW    GET_WNDSC                   ; Get CCB$L_WIND image descriptor
          67    D4   029E   624          CLRL    (R7)                        ; Clear CCB$L_WIND image in the buffer
                     02A0   625          SETBIT  #IRP$V_FUNC,IRP$W_STS(R3)   ; Mark for write back
 32 A3    01    B0  02A5   626          MOVW    #1,IRP$W_BCNT(R3)           ; Write back 1 (the window) ABD
```

```
                                                M 8
                     01    3C  02A9   627        MOVZWL  S^#SS$_NORMAL,-          ; Request has succeeded
                 38  A3         02AB   628                IRP$L_IOST1(R3)
                             02AD   629        CLRBIT  #DWB$V_RUN,-             ; Mark DWB no longer active
                             02AD   630                DWB$W_FLAGS(R6)
                             02B2   631        SETBIT  #DWB$V_DELETE,-          ; Mark DWB to be deleted
                             02B2   632                DWB$W_FLAGS(R6)
                   0004      30  02B7   633        BSBW    DEALLOC_DWB              ; Deallocate DWB, if possible
                   004B      30  02BA   634        BSBW    RESTORE_QUOTA            ; Restore quota
                             05  02BD   635        RSB
```

```
                     02BE    637          .SBTTL  DEALLOC_DWB     - Deallocate DWB
                     02BE    638 ;+
                     02BE    639 ; DEALLOC_DWB - Deallocate DWB, if possible
                     02BE    640 ;
                     02BE    641 ; This routine is called to deallocate the DWB if an access request fails,
                     02BE    642 ; or as a result of a deaccess function.  If the DWB still has I/O pending,
                     02BE    643 ; the DWB is marked for delete, so that when the I/O completes, the DWB is
                     02BE    644 ; deleted.
                     02BE    645 ;
                     02BE    646 ; Inputs:
                     02BE    647 ;
                     02BE    648 ;       R6 = DWB address (may be zero)
                     02BE    649 ;
                     02BE    650 ; Outputs:
                     02BE    651 ;
                     02BE    652 ;       None
                     02BE    653 ;
                     02BE    654 ;       R0-R1 are destroyed.
                     02BE    655 ;-
                     02BE    656 DEALLOC_DWB:
          56   D5    02BE    657          TSTL    R6                      ; Check DWB address
          2F   13    02C0    658          BEQL    90$                     ; Skip if none
          04   E1    02C2    659          BBC     #DWB$V_DELETE,-         ; If not yet marked for delete,
    2A 0E A6         02C4    660                  DWB$W_FLAGS(R6),90$     ; then wait until it is allowed
                     02C7    661          ;
                     02C7    662          ;   Deallocate all received messages
                     02C7    663          ;
51  14 B6   0F      02C7    664 10$:     REMQUE  @DWB$Q_RCV_MSG(R6),R1   ; Get received message IRP
          04   1D    02CB    665          BVS     20$                     ; If VS then none
          27   10    02CD    666          BSBB    DEALLOC_MSG             ; Deallocate received message
          F6   11    02CF    667          BRB     10$                     ; Loop
                     02D1    668 20$:     ;
                     02D1    669          ;   If any IRPs are still queued to the datalink, then leave
                     02D1    670          ;   the DWB around until the I/O completes.
                     02D1    671          ;
     47 A6   95      02D1    672          TSTB    DWB$B_IRPCNT(R6)        ; Any IRPs still queued to datalink?
          1B   12    02D4    673          BNEQ    90$                     ; If so, wait for them to complete
                     02D6    674          ;
                     02D6    675          ;   Deallocate the DWB
                     02D6    676          ;
50   38 A6   D0      02D6    677          MOVL    DWB$L_UCB0(R6),R0       ; Get UCB address
50   34 A0   D0      02DA    678          MOVL    UCB$L_VCB(R0),R0        ; Get VCB address
          12   13    02DE    679          BEQL    100$                    ; If none, fatal error
     54 A0   B7      02E0    680          DECW    RCB$W_MCOUNT(R0)        ; Decrement ACP mount count
   56   66   0F      02E3    681          REMQUE  (R6),R6                 ; Remove from UCB$Q_DWB_LIST
   50   56   D0      02E6    682          MOVL    R6,R0                   ; Get buffer address
00000000'GF   16      02E9    683          JSB     G^COM$DRVDEALMEM        ; Deallocate block
          56   D4    02EF    684          CLRL    R6                      ; Invalidate pointer
          05   05    02F1    685 90$:     RSB
                     02F2    686
                     02F2    687 100$:    BUG_CHECK NETNOSTATE,FATAL      ; ACP dismounted while DWB still active
                     02F6    688
                     02F6    689 ;
                     02F6    690 ; Deallocate a received message IRP
                     02F6    691 ;
                     02F6    692
                     02F6    693 DEALLOC_MSG:
```

```
                50    24 A1    D0  02F6   694           MOVL    IRP$L_IOSB(R1),R0       ; Get CXB address, if any
                         02    13  02FA   695           BEQL    15$                     ; Skip if none
                         03    10  02FC   696           BSBB    50$                     ; Deallocate it
                   50    51    D0  02FE   697 15$:      MOVL    R1,R0                   ; Setup IRP address
00000000'GF             16    0301   698 50$:      JSB     G^COM$DRVDEALMEM        ; Deallocate it
                         05    0307   699           RSB
```

```
                                0308   701          .SBTTL  RESTORE_QUOTA   - Restore "access" quota
                                0308   702 ;+
                                0308   703 ; RESTORE_QUOTA - Restore quota taken for accessed channel
                                0308   704 ;
                                0308   705 ; This routine is called to return the user's quota when an access request
                                0308   706 ; fails, or on a deaccess function.
                                0308   707 ;
                                0308   708 ; Inputs:
                                0308   709 ;
                                0308   710 ;       R3 = IRP address
                                0308   711 ;
                                0308   712 ; Outputs:
                                0308   713 ;
                                0308   714 ;       R0-R1 destroyed.
                                0308   715 ;-
                                0308   716 RESTORE_QUOTA:
      50    0C A3    3C        0308   717          MOVZWL  IRP$L_PID(R3),R0         ; Get process index
51    00000000'GF    D0        030C   718          MOVL    G^SCH$GL_PCBVEC,R1       ; Get address of PCB vector
        51    6140   D0        0313   719          MOVL    (R1)[R0],R1              ; Get PCB address
   0C A3    60 A1    D1        0317   720          CMPL    PCB$L_PID(R1),IRP$L_PID(R3) ; Still the same process?
             08       12        031C   721          BNEQ    90$                      ; If not, skip it
    50   0080 C1    D0        031E   722          MOVL    PCB$L_JIB(R1),R0         ; Get JIB address
           30 A0    B6        0323   723          INCW    JIB$W_FILCNT(R0)         ; Return quota taken for access
                    05        0326   724 90$:     RSB
```

```
                                0327   726          .SBTTL  DLE$FDT_SETMODE - Process IO$_SETMODE request
                                0327   727 ;+
                                0327   728 ; DLE$FDT_SETMODE       Process IO$_SETMODE QIO
                                0327   729 ;
                                0327   730 ; Sets the substate for the DLE adjacency.
                                0327   731 ;
                                0327   732 ; Inputs:
                                0327   733 ;
                                0327   734 ;       R6 = CCB address
                                0327   735 ;       R5 = UCB address
                                0327   736 ;       R4 = PCB address
                                0327   737 ;       R3 = IRP address
                                0327   738 ;
                                0327   739 ;       P2(AP) = Address of UNA P2 buffer (used only for DLE access to UNA)
                                0327   740 ;       P3(AP) = Address of 6 byte remote NI address (used only for UNA)
                                0327   741 ;       P4(AP) = Address of 4 byte DLE substate for connection
                                0327   742 ;
                                0327   743 ; Outputs:
                                0327   744 ;
                                0327   745 ;       I/O is posted
                                0327   746 ;-
                                0327   747 DLE$FDT_SETMODE::                        ; Process IO$_SETMODE function
52  18 A3  01  CB              0327   748          BICL3   #1,IRP$L_WIND(R3),R2    ; Clear interlock bit
               3B  18           032C   749          BGEQ    60$                     ; Error if link not accessed
                                032E   750          ;
                                032E   751          ;   If P4 specified, update the DLE substate
                                032E   752          ;
    50  0C AC  D0              032E   753          MOVL    P4(AP),R0               ; Is P4 specified?
               0A  13           0332   754          BEQL    15$                     ; If not, skip it
                                0334   755          IFNORD  #4,(R0),70$             ; If not readable, report error
    46 A2      60  90           033A   756          MOVB    (R0),DWB$B_SUBSTA(R2)   ; Set the new sub-state
                                033E   757 15$:     ;
                                033E   758          ;   If P3 specified, update the remote NI address
                                033E   759          ;
    50  08 AC  D0              033E   760          MOVL    P3(AP),R0               ; Is P3 specified?
               0E  13           0342   761          BEQL    25$                     ; If not, skip it
                                0344   762          IFNORD  #6,(R0),70$             ; If not readable, report error
    40 A2      80  D0           034A   763          MOVL    (R0)+,DWB$G_REMNOD(R2)  ; Store remote NI address
    44 A2      60  B0           034E   764          MOVW    (R0),DWB$G_REMNOD+4(R2)
                                0352   765 25$:     ;
                                0352   766          ;   Send the request to NETACP so that it can finish processing
                                0352   767          ;   those arguments which require process level action (like the
                                0352   768          ;   UNA P2 buffer).
                                0352   769          ;
                                0352   770          SETBIT  #IRP$V_PHYSIO,-         ; Mark this as a DLE IRP
                                0352   771                  IRP$W_STS(R3)
            29  10              0357   772          BSBB    SETMODE_ACPBUF          ; Build ACP complex buffer
                                0359   773          SETIPL  #IPL$_SYNCH             ; Synchronize with I/O database
    50  34 A5  D0              035C   774          MOVL    UCB$L_VCB(R5),R0        ; Get VCB address
        0C A0  B6              0360   775          INCW    VCB$W_TRANS(R0)         ; Increment transaction count
00000000'GF    17              0363   776          JMP     G^EXE$QIOACPPKT         ; Queue packet to AQB and RET
                                0369   777
                                0369   778          ;
                                0369   779          ;   Error paths
                                0369   780          ;
50  00AC 8F    3C              0369   781 60$:     MOVZWL  #SS$_FILNOTACC,R0       ; Say "link not accessed"
               03  11           036E   782          BRB     80$                     ; Abort the I/O
```

```
                     0370   783
          50   0C    3C  0370   784 70$:    MOVZWL  #SS$_ACCVIO,R0          ; Access violation reading NI address
             FD9A    31  0373   785 80$:    BRW     ABORTIO                 ; Abort the I/O
                     0376   786
                     0376   787 ;
                     0376   788 ; We arrive here after the ACP has finished with the IRP, and has requeued
                     0376   789 ; it to this driver for further processing.
                     0376   790 ;
                     0376   791 ; Inputs:
                     0376   792 ;
                     0376   793 ;       R6 = DWB address
                     0376   794 ;       R5 = UCB address
                     0376   795 ;       R3 = IRP address
                     0376   796 ;
                     0376   797 ; Outputs:
                     0376   798 ;
                     0376   799 ;       None
                     0376   800 ;
                     0376   801
                     0376   802 DLE$SETMODE::                               ; IO$_SETMODE "startio" processing
        07 38 A3     E9  0376   803         BLBC    IRP$L_IOST1(R3),90$      ; If error, leave IOST2 alone
              0090   30  037A   804         BSBW    GET_STATUS              ; Get status flags
     3C A3    51     D0  037D   805         MOVL    R1,IRP$L_IOST2(R3)      ; Store them in second longword
                     05  0381   806 90$:    RSB                             ; Post I/O with ACP status in IOST1
```

```
                                0382   808          .SBTTL  SETMODE_ACPBUF  - Build SETMODE ACP complex buffer
                                0382   809 ;+
                                0382   810 ; SETMODE_ACPBUF - Build complex buffer to pass QIO parameters to ACP
                                0382   811 ;
                                0382   812 ; This routine is called by ACCESS, DEACCESS, ACPCONTROL and SETMODE
                                0382   813 ; to pass it's QIO arguments (P1-P6) to the ACP in a complex buffer
                                0382   814 ; (a series of ABD descriptors).
                                0382   815 ;
                                0382   816 ; Inputs:
                                0382   817 ;
                                0382   818 ;       R6 = CCB address
                                0382   819 ;       R5 = UCB address
                                0382   820 ;       R4 = PCB address
                                0382   821 ;       R3 = IRP address
                                0382   822 ;
                                0382   823 ;       P2 = Address of a descriptor of a input buffer (read, not written)
                                0382   824 ;
                                0382   825 ; Outputs:
                                0382   826 ;
                                0382   827 ;       IRP$L_SVAPTE = Address of complex buffer
                                0382   828 ;
                                0382   829 ;       All errors go to ABORTIO.
                                0382   830 ;
                                0382   831 ;       R0-R2,R9-R10 are destroyed.
                                0382   832 ;-
                                0382   833 SETMODE_ACPBUF:
      51   0000004B 8F    D0   0382   834          MOVL    #12+<7*9>,R1            ; Compute basic ABD overhead for 6 args
                                0389   835                                          ; plus a window descriptor
       5A     04 AC    D0   0389   836          MOVL    P2(AP),R10              ; Get address of descriptor
                 0C    13   038D   837          BEQL    10$                     ; If specified,
                                038F   838          IFNORD  #8,(R10),70$            ; and if descriptor can be read,
          50     6A    3C   0395   839          MOVZWL  (R10),R0                ; Get size of P2 buffer
          51     50    C0   0398   840          ADDL    R0,R1                   ; Add in the size of the buffer
                 53    DD   039B   841 10$:     PUSHL   R3                      ; Save IRP address
           00000000'GF    16   039D   842          JSB     G^EXE$ALLOCBUF          ; Allocate complex buffer
                 53 8ED0   03A3   843          POPL    R3                      ; Restore IRP address
              61 50    E9   03A6   844          BLBC    R0,90$                  ; Exit if error detected
        2C A3    52    D0   03A9   845          MOVL    R2,IRP$L_SVAPTE(R3)     ; Save address of buffer
        32 A3    07    B0   03AD   846          MOVW    #7,IRP$W_BCNT(R3)       ; Save number of descriptors
                       03B1   847          SETBIT  #IRP$V_COMPLX,IRP$W_STS(R3) ; Mark buffer as complex
        62    0C A2    9E   03B6   848          MOVAB   12(R2),(R2)             ; First longword points to data
                 3C    BB   03BA   849          PUSHR   #^M<R2,R3,R4,R5>        ; Save registers
0C A2   38    00    6E    00   03BC   850          MOVC5   #0,(SP),#0,#7*8,12(R2)  ; Zero all the descriptors,
                 51    53    D0   03C3   851          MOVL    R3,R1                   ; leaving R1 set to area afterwards
                 3C    BA   03C6   852          POPR    #^M<R2,R3,R4,R5>        ; Restore registers
        50    0C A2    9E   03C8   853          MOVAB   12(R2),R0               ; Point to first descriptor
                       03CC   854          ;
                       03CC   855          ;   Copy window pointer to complex buffer
                       03CC   856          ;
        02 A0    04    B0   03CC   857          MOVW    #4,ABD$W_COUNT(R0)      ; Set size of window pointer
     59    51    50    C3   03D0   858          SUBL3   R0,R1,R9                ; Compute offset to data area
           60    59    B0   03D4   859          MOVW    R9,ABD$W_TEXT(R0)       ; Set offset to window pointer
  04 A0    04 A6    9E   03D7   860          MOVAB   CCB$L_WIND(R6),ABD$L_USERVA(R0) ; Set address in process space
           81    94   03DC   861          CLRB    (R1)+                   ; Set access mode = kernel
        81    04 A6    D0   03DE   862          MOVL    CCB$L_WIND(R6),(R1)+    ; Copy actual window pointer
                       03E2   863          ;
                       03E2   864          ;   Copy P2 argument to complex buffer
```

```
                                03E2    865          ;
                50    10    C0  03E2    866          ADDL    #2+8,R0                 ; Skip to P2 descriptor
                      5A    D5  03E5    867          TSTL    R10                     ; Was P2 specified?
                      1D    13  03E7    868          BEQL    50$                     ; If not, skip the following
              02 A0   6A    B0  03E9    869          MOVW    (R10),ABD$W_COUNT(R0)   ; Set size of P2 buffer
           59 51      50    C3  03ED    870          SUBL3   R0,R1,R9                ; Compute offset to data area
              60      59    B0  03F1    871          MOVW    R9,ABD$W_TEXT(R0)       ; Set offset to window pointer
        04 A0    04   AA    D0  03F4    872          MOVL    4(R10),ABD$L_USERVA(R0) ; Set address in process space
           81    0B   A3    90  03F9    873          MOVB    IRP$B_RMOD(R3),(R1)+    ; Copy access mode
                      3F    BB  03FD    874          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers
61      04 BA         6A    28  03FF    875          MOVC    (R10),@4(R10),(R1)      ; Copy actual P2 buffer
                      3F    BA  0404    876          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers
                            05  0406    877 50$:     RSB
                                0407    878
                50    0C    3C  0407    879 70$:     MOVZWL  #SS$_ACCVIO,R0          ; Access violation
                   FD03       31  040A    880 90$:     BRW     ABORTIO                 ; Abort the I/O
```

```
                        040D   882          .SBTTL  GET_STATUS      - Refresh DLE status flags for IOSB
                        040D   883 ;+
                        040D   884 ; GET_STATUS - Construct updated status flags for 2nd longword of IOSB
                        040D   885 ;
                        040D   886 ; This routine is called each time we want to return the DLE status flags
                        040D   887 ; to the caller.  Currently, these flags are returned on all ACCESS and
                        040D   888 ; SETMODE functions.
                        040D   889 ;
                        040D   890 ; The flags are defined by the STAT structure.
                        040D   891 ;
                        040D   892 ; Inputs:
                        040D   893 ;
                        040D   894 ;       R6 = DWB address
                        040D   895 ;
                        040D   896 ; Outputs:
                        040D   897 ;
                        040D   898 ;       R1 = DLE status flags
                        040D   899 ;
                        040D   900 ;       All other registers are preserved.
                        040D   901 ;-
                        040D   902 GET_STATUS:
                51      D4      040D   903          CLRL    R1                      ; Init. flags longword
                                040F   904          ;
                                040F   905          ;   Set a flag indicating if the circuit being used is
                                040F   906          ;   a "broadcast circuit" (Ethernet) or not.
                                040F   907          ;
        04 0E A6        03      E1      040F   908          BBC     #DWB$V_BC,DWB$W_FLAGS(R6),20$ ; If flag set,
                                0414   909          SETBIT  #STAT_V_BC,R1           ; Indicate that the circuit is an NI
                                05      0418   910 20$:     RSB
```

```
                     0419   912          .SBTTL  DLE$FDT_CONTROL - IO$_ACPCONTROL FDT processing
                     0419   913          .SBTTL  DLE$CONTROL     - IO$_ACPCONTROL "startio" processing
                     0419   914 ;+
                     0419   915 ;  DLE$FDT_CONTROL      - IO$_ACPCONTROL FDT processing
                     0419   916 ;  DLE$CONTROL          - IO$_ACPCONTROL "startio" processing
                     0419   917 ;
                     0419   918 ;  The FDT routine simply routes the IRP through the Exec to the ACP.  The Exec
                     0419   919 ;  builds a "complex buffer" describing the control function.  The ACP will
                     0419   920 ;  requeue any IRP to the driver if it does not recognize the control function.
                     0419   921 ;  The driver has been designed to handle some of its own control functions
                     0419   922 ;  since many are protocol or control block format specific.
                     0419   923 ;
                     0419   924 ;  Inputs:
                     0419   925 ;
                     0419   926 ;       R6 = CCB address
                     0419   927 ;       R5 = UCB address
                     0419   928 ;       R4 = PCB address
                     0419   929 ;       R3 = IRP address
                     0419   930 ;
                     0419   931 ;  Outputs:
                     0419   932 ;
                     0419   933 ;       None
                     0419   934 ;-
                     0419   935 DLE$FDT_CONTROL::                        ; "FDT" phase for IO$_ACPCONTROL
 18 A3   01   CA     0419   936          BICL    #1,IRP$L_WIND(R3)       ; Always clear interlock flag
                     041D   937          SETBIT  #IRP$V_PHYSIO,-         ; Mark this as a DLE IRP
                     041D   938                  IRP$W_STS(R3)
00000000'GF   17     0422   939          JMP     G^ACP$MODIFY            ; Continue in EXEC
                     0428   940
                     0428   941 ;
                     0428   942 ;  We arrive here after the ACP has finished with the IRP, and has requeued
                     0428   943 ;  it to this driver for further processing.
                     0428   944 ;
                     0428   945 ;  Inputs:
                     0428   946 ;
                     0428   947 ;       R6 = DWB address (may be zero)
                     0428   948 ;       R5 = UCB address
                     0428   949 ;       R3 = IRP address
                     0428   950 ;
                     0428   951 ;  Outputs:
                     0428   952 ;
                     0428   953 ;       None
                     0428   954 ;
                     0428   955 DLE$CONTROL::                            ; "Startio" phase for IO$_ACPCONTROL
 50   2C A3   D0     0428   956          MOVL    IRP$L_SVAPTE(R3),R0     ; Get ptr to complex buffer
      06   13     042C   957          BEQL    10$                     ; Branch if none (if $CANCEL issues it)
   50 60   D0     042E   958          MOVL    (R0),R0                 ; Get pointer to window descriptor
      60   D4     0431   959          CLRL    (R0)                    ; Prevent I/O from affecting the window
                     0433   960                                          ; by clearing byte count in descriptor
           05     0433   961          RSB                             ; Exit with ACP status unchanged
                     0434   962
                     0434   963 ;
                     0434   964 ;  $CANCEL issues an IO$_ACPCONTROL function without any buffer to signal
                     0434   965 ;  a cancellation of an accessed channel.
                     0434   966 ;
                     0434   967
      01   3C     0434   968 10$:    MOVZWL  S^#SS$_NORMAL,-         ; Set I/O status
```

```
                38 A3        0436   969                     IRP$L_IOST1(R3)         ;
            56  01  CA       0438   970         BICL    #1,R6                       ; Clear interlock bit
                01  19       043B   971         BLSS    40$                         ; If LSS then valid DWB
                             043D   972         ;
                             043D   973         ;   This is a cancel of a channel with an ACCESS pending.  Since
                             043D   974         ;   the ACP has the IRP in-hand, there is nothing we can do to
                             043D   975         ;   get it back from here.
                             043D   976         ;
                    05       043D   977         RSB
                             043E   978
                             043E   979         ;
                             043E   980         ;   Cancel all outstanding I/O for active channel.
                             043E   981         ;
                0002    30   043E   982 40$:    BSBW    CANCEL_ALL                  ; Cancel all outstanding I/O
                    05       0441   983         RSB                                 ; Exit
```

```
                        0442   985          .SBTTL  DLE$CANCEL      - Cancel I/O routine
                        0442   986 ;+
                        0442   987 ; DLE$CANCEL    - Cancel I/O entry point
                        0442   988 ;
                        0442   989 ; Most of the work for the Cancel-I/O sequence will occur when the special
                        0442   990 ; IO$_ACPCONTROL QIO is issued by the $CANCEL system service.  A cancel
                        0442   991 ; issued without the channel being accessed requires no work.
                        0442   992 ;
                        0442   993 ; Inputs:
                        0442   994 ;
                        0442   995 ;       R8 = Reason for cancel (CAN$C_CANCEL or CAN$C_DASSGN)
                        0442   996 ;       R5 = UCB address
                        0442   997 ;       R4 = PCB address
                        0442   998 ;       R3 = IRP address if UCB is busy
                        0442   999 ;       R2 = Channel number
                        0442  1000 ;
                        0442  1001 ;       IPL = FIPL
                        0442  1002 ;
                        0442  1003 ; Outputs:
                        0442  1004 ;
                        0442  1005 ;       R0-R3 may be destroyed.
                        0442  1006 ;-
                        0442  1007 DLE$CANCEL::                            ; Cancel I/O entry point
05                      0442  1008          RSB                            ; Done
```

```
                     0443  1010          .SBTTL  CANCEL_ALL      - Cancel all outstanding I/O
                     0443  1011 ;+
                     0443  1012 ; CANCEL_ALL    - Run down all outstanding I/O
                     0443  1013 ;
                     0443  1014 ; This routine is called to run down all I/O pending for the channel.
                     0443  1015 ;
                     0443  1016 ; Inputs:
                     0443  1017 ;
                     0443  1018 ;       R6 = DWB address
                     0443  1019 ;
                     0443  1020 ; Outputs:
                     0443  1021 ;
                     0443  1022 ;       None
                     0443  1023 ;
                     0443  1024 ;       R0-R2 are destroyed.
                     0443  1025 ;-
                     0443  1026 CANCEL_ALL::                            ; Cancel all outstanding I/O
  0178 8F    BB      0443  1027          PUSHR   #^M<R3,R4,R5,R6,R8>     ; Save regs
                     0447  1028          ;
                     0447  1029          ;   Abort all pending user transmits which haven't yet been
                     0447  1030          ;   queued to the datalink.
                     0447  1031          ;
  58   24 A6   9E    0447  1032          MOVAB   DWB$Q_USER_XMT(R6),R8   ; Get user transmit IRP listhead
         46   10    044B  1033          BSBB    ABORT_IRPS              ; Drain it
                     044D  1034          ;
                     044D  1035          ;   Issue a cancel request to the datalink to abort any I/O
                     044D  1036          ;   requests we have outstanding to the datalink.  When the
                     044D  1037          ;   receive IRPs come back aborted, we will return that status
                     044D  1038          ;   in the corresponding user IRP.
                     044D  1039          ;
     58    00   9A    044D  1040          MOVZBL  #CAN$C_CANCEL,R8        ; Indicate "cancel", not "deassign"
   52   4C A6   AE    0450  1041          MNEGW   DWB$W_DLL_CHAN(R6),R2   ; Set channel to datalink
           53   D4    0454  1042          CLRL    R3                      ; No "current IRP"
   50   38 A6   D0    0456  1043          MOVL    DWB$L_UCB0(R6),R0       ; Get address of ND's UCB0
   50   34 A0   D0    045A  1044          MOVL    UCB$L_VCB(R0),R0        ; Get address of our VCB
   51   10 A0   D0    045E  1045          MOVL    VCB$L_AQB(R0),R1        ; Get address of our AQB
   50   0C A1   3C    0462  1046          MOVZWL  AQB$L_ACPPID(R1),R0     ; Get ACP process index
54 00000000'GF   D0    0466  1047          MOVL    G^SCH$GL_PCBVEC,R4      ; Get address of PCB vector
     54   6440   D0    046D  1048          MOVL    (R4)[R0],R4             ; Get PCB address
0C A1   60 A4   D1    0471  1049          CMPL    PCB$L_PID(R4),AQB$L_ACPPID(R1) ; Still the same process?
           16   12    0476  1050          BNEQ    20$                     ; If not, skip it
   55   48 A6   D0    0478  1051          MOVL    DWB$L_DLL_UCB(R6),R5    ; Get datalink UCB address
                     047C  1052          DSBINT  UCB$B_FIP[(R5)          ; Change IPL to datalink's level
 51   0088 C5   D0    0483  1053          MOVL    UCB$L_DDT(R5),R1        ; Get datalink DDT
     0C B1   16    0488  1054          JSB     @DDT$[_CANCEL(R1)       ; Call datalink cancel entry point
                     048B  1055          ENBINT                          ; Restore IPL
  0178 8F    BA      048E  1056 20$:     POPR    #^M<R3,R4,R5,R6,R8>     ; Restore regs
           05    0492  1057          RSB                             ; Done
                     0493  1058
                     0493  1059 ABORT_IRPS:
   53   00 B8   0F    0493  1060          REMQUE  @(R8),R3                ; Get IRP
           12   1D    0497  1061          BVS     30$                     ; If VS then none
  0830 8F   3C    0499  1062          MOVZWL  #SS$_CANCEL,-           ; Setup I/O status and 0 transfer size
     38 A3         049D  1063                  IRP$[_IOST1(R3)
   55   1C A3   D0    049F  1064          MOVL    IRP$L_UCB(R3),R5        ; Get UCB address
00000000'GF   16    04A3  1065          JSB     G^COM$POST              ; Another packet for the heap
           E8   11    04A9  1066          BRB     ABORT_IRPS              ; Loop
```

```
 05  04AB  1067 30$:     RSB                                ; Done
```

```
                        04AC  1069          .SBTTL  DLE$LPD_DOWN    - The circuit has gone away
                        04AC  1070 ;+
                        04AC  1071 ; DLE$LPD_DOWN - Handle "LPD down" notification from NETACP
                        04AC  1072 ;
                        04AC  1073 ; This routine is called by NETACP when the LPD becomes unusable for reasons
                        04AC  1074 ; beyond our control.  We must locate all DLE users associated with this
                        04AC  1075 ; circuit, and cause the DLE sessions to be terminated.
                        04AC  1076 ;
                        04AC  1077 ; Inputs:
                        04AC  1078 ;
                        04AC  1079 ;       IPL = 0
                        04AC  1080 ;
                        04AC  1081 ;       R5 = any DLE UCB address
                        04AC  1082 ;       R8 = LPD ID
                        04AC  1083 ;       R0 = Status code
                        04AC  1084 ;
                        04AC  1085 ; Outputs:
                        04AC  1086 ;
                        04AC  1087 ;       None
                        04AC  1088 ;
                        04AC  1089 ;       R0-R1 is destroyed.
                        04AC  1090 ;-
                        04AC  1091 DLE$LPD_DOWN::
                        04AC  1092          DSBINT  UCB$B_FIPL(R5)          ; Synchronize
     00C0 8F    BB      04B3  1093          PUSHR   #^M<R6,R7>              ; Save registers
  50   28 A5    D0      04B7  1094          MOVL    UCB$L_DDB(R5),R0        ; Get DLE DDB address
  50   04 A0    D0      04BB  1095          MOVL    DDB$L_UCB(R0),R0        ; Get DLE UCB0 address
57   0090 C0    9E      04BF  1096          MOVAB   UCB$Q_DWB_LIST(R0),R7   ; Get address of listhead
     56   57    D0      04C4  1097          MOVL    R7,R6                   ; Setup for loop
     56   66    D0      04C7  1098 10$:     MOVL    (R6),R6                 ; Skip to next DWB
     57   56    D1      04CA  1099          CMPL    R6,R7                   ; End of list?
          10    13      04CD  1100          BEQL    90$                     ; If so, exit
  58   3E A6    B1      04CF  1101          CMPW    DWB$W_PATH(R6),R8       ; Does the circuit ID match?
          F2    12      04D3  1102          BNEQ    10$                     ; If not, keep looking
                        04D5  1103          ;
                        04D5  1104          ;   Clear RUN flag, so that no further I/O is allowed on channel
                        04D5  1105          ;
ED 0E A6    00    E5    04D5  1106          BBCC    #DWB$V_RUN,DWB$W_FLAGS(R6),10$ ; Clear RUN flag
                        04DA  1107                                          ; If not in RUN state, don't do cancel
                        04DA  1108          ;
                        04DA  1109          ;   Cancel all outstanding I/O operations to datalink
                        04DA  1110          ;
        FF66    30      04DA  1111          BSBW    CANCEL_ALL              ; Cancel all I/O
          E8    11      04DD  1112          BRB     10$                     ; Keep looping
                        04DF  1113
     00C0 8F    BA      04DF  1114 90$:     POPR    #^M<R6,R7>              ; Restore registers
                        04E3  1115          ENBINT                          ; Restore IPL
                  05    04E6  1116          RSB
```

```
                                1118          .SBTTL  DLE$FDT_RCV     - FDT for IO$_READxBLK requests
                                1119 ;+
                                1120 ; DLE$FDT_RCV - FDT routine for read requests
                                1121 ;
                                1122 ; This routine prepares an IRP to be used to receive a message.
                                1123 ;
                                1124 ; The NOW modifier is allowed so the user can request immediate
                                1125 ; notification if there are no messages pend' g.
                                1126 ;
                                1127 ; Inputs:
                                1128 ;
                                1129 ;       R5 = UCB address
                                1130 ;       R4 = PCB address
                                1131 ;       R3 = IRP address
                                1132 ;
                                1133 ; Outputs:
                                1134 ;
                                1135 ;       None
                                1136 ;-
                                1137 DLE$FDT_RCV:
                                1138          ;
                                1139          ;   If the P5 buffer is specified, then it must point to a 14 byte
                                1140          ;   user writable buffer into which will be stored the NI datalink
                                1141          ;   header when the receive completes.
                                1142          ;
57   10 AC    D0  04E7  1143          MOVL    P5(AP),R7               ; Get address of buffer
       29    13  04EB  1144          BEQL    20$                     ; Skip if not specified
                 04ED  1145          IFNOWRT #DIAG_C_NIHDRSIZ,(R7),70$ ; Error if buffer not writable
       07    E0  04F3  1146          BBS     #IRP$V_DIAGBUF,-        ; If diagnostic buffer already given,
 1E 2A A3         04F5  1147                  IRP$W_STS(R3),20$       ; then skip it
   51   1A    9A  04F8  1148          MOVZBL  #DIAG_C_LENGTH,R1       ; Get size of diagnostics buffer
     03CB    30  04FB  1149          BSBW    DLE$ALCNONPAGED         ; Allocate buffer
      21 50    E9  04FE  1150          BLBC    R0,80$                  ; Branch if error detected
 4C A3    52    D0  0501  1151          MOVL    R2,IRP$L_DIAGBUF(R3)    ; Save address of buffer
 08 A2    51    B0  0505  1152          MOVW    R1,DIAG_W_SIZE(R2)      ; Set length of structure
    0C A2       9E  0509  1153          MOVAB   DIAG_G_DATA(R2),-       ; Set address of data portion
       62            050C  1154                  DIAG_L_DATA(R2)
 04 A2    57    D0  050D  1155          MOVL    R7,DIAG_L_USERBUF(R2)   ; Set address of user buffer
                  0511  1156          SETBIT  #IRP$V_DIAGBUF,-        ; Mark diagnostics buffer present
                  0511  1157                  IRP$W_STS(R3)
                  0516  1158 20$:      ;
                  0516  1159          ;   Join common data buffer probing
                  0516  1160          ;
52   00000000'GF  9E  0516  1161          MOVAB   G^EXE$READCHKR,R2       ; Setup buffer probe action routine
          0F    11  051D  1162          BRB     DLE$FDT_RW              ; Join common code
                  051F  1163
                  051F  1164          ;
                  051F  1165          ;   Error paths
                  051F  1166          ;
   50   0C    9A  051F  1167 70$:     MOVZBL  S^#SS$_ACCVIO,R0        ; Assume access violation
      FBEB    31  0522  1168 80$:     BRW     ABORTIO
```

```
                     0525  1170          .SBTTL  DLE$FDT_XMT     - FDT for IO$_WRITExBLK requests
                     0525  1171 ;+
                     0525  1172 ; DLE$FDT_XMT - FDT routine for write requests
                     0525  1173 ;
                     0525  1174 ; This routine prepares an IRP to be used to write a message.
                     0525  1175 ;
                     0525  1176 ; Inputs:
                     0525  1177 ;
                     0525  1178 ;       R5 = UCB address
                     0525  1179 ;       R4 = PCB address
                     0525  1180 ;       R3 = IRP address
                     0525  1181 ;
                     0525  1182 ; Outputs:
                     0525  1183 ;
                     0525  1184 ;       None
                     0525  1185 ;-
                     0525  1186 DLE$FDT_XMT:
        52 00000000'GF   9E  0525  1187          MOVAB   G^EXE$WRITECHKR,R2      ; Setup buffer probe action routine
                   00    11  052C  1188          BRB     DLE$FDT_RW              ; Join common code
```

```
                        052E  1190          .SBTTL  DLE$FDT_RW      - DLE FDT Read/Write processing
                        052E  1191 ;+
                        052E  1192 ; DLE$FDT_RW    - Complete FDT Read/Write buffered I/O
                        052E  1193 ;
                        052E  1194 ; Probe the buffer.  Allocate a system buffer to receive
                        052E  1195 ; or transmit the data and queue the IRP to the XWB.
                        052E  1196 ;
                        052E  1197 ; Inputs:
                        052E  1198 ;
                        052E  1199 ;       R5 = UCB address
                        052E  1200 ;       R4 = PCB address
                        052E  1201 ;       R3 = IRP address
                        052E  1202 ;       R2 = Address of buffer access check routine
                        052E  1203 ;
                        052E  1204 ;       P1(AP) = Address of user buffer
                        052E  1205 ;       P2(AP) = Length of user buffer
                        052E  1206 ;       P5(AP) = Address of 14-byte buffer to receive NI datalink header
                        052E  1207 ;
                        052E  1208 ; Outputs:
                        052E  1209 ;
                        052E  1210 ;       None
                        052E  1211 ;-
                        052E  1212 DLE$FDT_RW::                                 ; DLE Read/Write FDT
 30 A3   B4             052E  1213          CLRW    IRP$W_BOFF(R3)              ; No byte quota taken yet
 2C A3   D4             0531  1214          CLRL    IRP$L_SVAPTE(R3)            ; Clear the buffer pointer
 38 A3   7C             0534  1215          CLRQ    IRP$L_IOST1(R3)             ; Clear IOSB image
                        0537  1216          ;
                        0537  1217          ;   Must only use the P1 and P2 parameters once.  Else, a malicious
                        0537  1218          ;   user could DMA some bogus data to overlay the argument list
                        0537  1219          ;   contents after they have been verified to be okay via a probe.
                        0537  1220          ;   Thus, copy P1 and P2 to registers and only use the registers from
                        0537  1221          ;   now on.
                        0537  1222          ;
    5A   6C   D0        0537  1223          MOVL    P1(AP),R10              ; Point to users' data
 5B   04 AC   3C        053A  1224          MOVZWL  P2(AP),R11              ; Number of bytes
                        053E  1225          ;
                        053E  1226          ;   Test for user buffer accessiblity
                        053E  1227          ;
    50   5A   7D        053E  1228          MOVQ    R10,R0                  ; Setup user's buffer descriptor
         53   DD        0541  1229          PUSHL   R3                      ; Save IRP address
         62   16        0543  1230          JSB     (R2)                    ; Is buffer accessible?
                        0545  1231                                          ; - clobbers R0-R3, sets up IRP$W_BCNT
         53 8ED0        0545  1232          POPL    R3                      ; Restore IRP address
      31 50   E9        0548  1233          BLBC    R0,5$                   ; If LBC then access violation
                        054B  1234          ;
                        054B  1235          ;   Get the window block, and make sure the channel has been accessed.
                        054B  1236          ;
50   00AC 8F   3C        054B  1237          MOVZWL  #SS$_FILNOTACC,R0       ; Assume not accessed yet
  56   18 A3   D0        0550  1238          MOVL    IRP$L_WIND(R3),R6       ; Get DWB address
         26   13        0554  1239          BEQL    5$                      ; If EQL window is not valid
      23 56   E8        0556  1240          BLBS    R6,5$                   ; If LBC window is not valid
                        0559  1241          SETIPL  #NET$C_IPL              ; Synchronize access to DWB
                        055C  1242          ;
                        055C  1243          ;   Consume user's byte quota needed for buffer.  Wait if necessary.
                        055C  1244          ;
    51   5B   D0        055C  1245          MOVL    R11,R1                  ; Setup size of user's buffer
       00AC   30        055F  1246          BSBW    DLE$FDT_BYTQUO          ; Consume byte quota needed.  May wait
```

```
                          0562  1247                                    ; at IPL$_ASTDEL if needed, but always
                          0562  1248                                    ; returns at NET$C_IPL.
                    17 50  E9 0562  1249          BLBC    R0,5$           ; If LBC then return to abort I/O
                          0565  1250          SETIPL  #IPL$_ASTDEL        ; Lower IPL for call to EXE$ALLOCBUF
                          0568  1251                                      ; and to get at user memory to copy
                          0568  1252                                      ; data if needed
51  5B  004C 8F  A1 0568  1253          ADDW3   #CXB$C_OVERHEAD,R11,R1  ; Get total buffer size
                       53  DD 056E  1254          PUSHL   R3              ; Save IRP address
                 0000000'GF  16 0570  1255          JSB     G^EXE$ALLOCBUF  ; Allocate non-paged buffer -- will
                          0576  1256                                      ; wait if needed and allowed by the
                          0576  1257                                      ; processes resource wait mode setting
                       53 8ED0 0576  1258          POPL    R3              ; Restore IRP address
                    03 50  E8 0579  1259          BLBS    R0,10$          ; If LBS then buffer was allocated
                     0088  31 057C  1260 5$:      BRW     100$            ; Return with error in R0
                 2C A3  52  D0 057F  1261 10$:     MOVL    R2,IRP$L_SVAPTE(R3) ; Store buffer address
                          0583  1262                                      ;
                          0583  1263                                      ;    Init the CXB
                          0583  1264                                      ;
                       3E  BB 0583  1265          PUSHR   #^M<R1,R2,R3,R4,R5> ; Save critical regs destroyed by MOVC
                00  6E  00  2C 0585  1266          MOVC5   #0,(SP),#0,-    ; Zero the CXB header
                62  0048 8F    0589  1267                  #CXB$C_HEADER,(R2) ;
                       3E  BA 058D  1268          POPR    #^M<R1,R2,R3,R4,R5> ; Restore regs
                  08 A2  51  B0 058F  1269          MOVW    R1,CXB$W_SIZE(R2) ; Store size for deallocation
                  0A A2  1B  90 0593  1270          MOVB    #DYN$C_CXB,CXB$B_TYPE(R2) ; Setup structure type
                  62   48 A2  9E 0597  1271          MOVAB   CXB$C_HEADER(R2),(R2) ; Store data area ptr in CXB
                  04 A2  5A  D0 059B  1272          MOVL    R10,4(R2)       ; Store user buffer addr
                  14 A2  53  D0 059F  1273          MOVL    R3,CXB$L_IRP(R2) ; Store IRP back-pointer
                         01  E0 05A3  1274          BBS     #IRP$V_FUNC,-   ; If BS then a RCV function
                     0A 2A A3    05A5  1275                    IRP$W_STS(R3),20$ ;
                       53  DD 05A8  1276          PUSHL   R3              ; Save IRP address
              00 B2    6A  5B  28 05AA  1277          MOVC3   R11,(R10),@(R2) ; Move user data
                       53 8ED0 05AF  1278          POPL    R3              ; Restore IRP address
                          05B2  1279 20$:     ;
                          05B2  1280          ;    Check DWB state.  If still active, then continue.  Else return
                          05B2  1281          ;    to abort the I/O.
                          05B2  1282          ;
                          05B2  1283          SETIPL  #NET$C_IPL          ; Synch access to the DWB
           50   00AC 8F  3C 05B5  1284          MOVZWL  #SS$_FILNOTACC,R0   ; Assume wrong state
                     00  E1 05BA  1285          BBC     #DWB$V_RUN,-        ; If connection not active,
                48 0E A6    05BC  1286                  DWB$W_FLAGS(R6),100$ ; then return an error
                     01  E0 05BF  1287          BBS     #IRP$V_FUNC,-       ; If BS then RCV request
                  09 2A A3    05C1  1288                   IRP$W_STS(R3),40$
                          05C4  1289          ;
                          05C4  1290          ;    Finish IO$_WRITE processing
                          05C4  1291          ;
              28 B6   63  0E 05C4  1292          INSQUE  (R3),@DWB$Q_USER_XMT+4(R6) ; Insert IRP onto queue
                    0087  30 05C8  1293          BSBW    DLE$XMT_MSG         ; Transmit the message, if possible
                      2A  11 05CB  1294          BRB     60$                 ; Done
                          05CD  1295          ;
                          05CD  1296          ;    Finish IO$_READ processing
                          05CD  1297          ;
           55   14 B6   0F 05CD  1298 40$:     REMQUE  @DWB$Q_RCV_MSG(R6),R5 ; Get waiting message (IRP)
                     0B  1D 05D1  1299          BVS     50$                 ; If VS then none
                    0271  30 05D3  1300          BSBW    RCV_DONE            ; Complete user IRP (in R3)
                51   55  D0 05D6  1301          MOVL    R5,R1               ; Set datalink receive IRP address
                    FD1A  30 05D9  1302          BSBW    DEALLOC_MSG         ; Deallocate received message IRP/CXB
                      19  11 05DC  1303          BRB     60$                 ; Take common exit
```

```
                        05DE  1304 50$:     ;
                        05DE  1305          ;   No received messages are pending for the user.  If the NOW
                        05DE  1306          ;   modifier was specified, then return an indication that no
                        05DE  1307          ;   messages are pending.
                        05DE  1308          ;
          06    E1      05DE  1309          BBC     #IO$V_NOW,-                 ; If NOW modifier specified,
07 20 A2                05E0  1310                  IRP$W_FUNC(R3),55$
50  0870 8F   3C        05E3  1311          MOVZWL  #SS$_ENDOFFILE,R0           ; Set error status
        1D    11        05E8  1312          BRB     100$                        ; and ABORT the I/O
                        05EA  1313          ;
                        05EA  1314          ;   Receive must wait for incoming data.  Issue receive to datalink
                        05EA  1315          ;   on behalf of the user.
                        05EA  1316          ;
20 B6   63    0E        05EA  1317 55$:     INSQUE  (R3),@DWB$Q_USER_RCV+4(R6) ; Insert on wait queue
      0155    30        05EE  1318          BSBW    INIT_RCV_IRP                ; Initialize datalink receive IRP
    0F 50     E9        05F1  1319          BLBC    R0,70$                      ; Branch if error detected
     019B     30        05F4  1320          BSBW    ISSUE_DLL_RCV               ; Issue the receive to the datalink
   50   01    3C        05F7  1321 60$:     MOVZWL  S^#SS$_NORMAL,R0            ; Success
                        05FA  1322          SETIPL  S^#IPL$_ASTDEL              ; Restore IPL
00000000'GF   17        05FD  1323          JMP     G^EXE$QIORETURN             ; Return to user with success
                        0603  1324
                        0603  1325
                        0603  1326 ;
                        0603  1327 ; An error has been detected.  Abort the I/O request.
                        0603  1328 ;
                        0603  1329
53   20 B6    0F        0603  1330 70$:     REMQUE  @DWB$Q_USER_RCV+4(R6),R3 ; Get the user IRP back
                        0607  1331                                          ; Return INIT_RCV_IRP status to user
                        0607  1332
55   1C A3    D0        0607  1333 100$:    MOVL    IRP$L_UCB(R3),R5            ; Restore UCB address
      FB02    31        060B  1334          BRW     ABORTIO                     ; Abort I/O with status in R0
```

```
                                060E  1336          .SBTTL  DLE$FDT_BYTQUO  - Get Non-paged Pool Quota
                                060E  1337 ;+
                                060E  1338 ; DLE$FDT_BYTQUO  - Get Non-paged Pool Quota
                                060E  1339 ;
                                060E  1340 ; Take indicate bytes from JIB BYTCNT quota.  Wait for them if necessary
                                060E  1341 ; and allowed.  The quota taken is subtracted form JIB$L_BYTCNT and moved
                                060E  1342 ; to IRP$W_BOFF.
                                060E  1343 ;
                                060E  1344 ; IPL is assumed to be NET$C_IPL and may be lowered to IPL$_ASTDEL or
                                060E  1345 ; IPL$_SYNCH.  May go into MWAIT at IPL$_ASTDEL, always return at NET$C_IPL.
                                060E  1346 ;
                                060E  1347 ; Inputs:
                                060E  1348 ;
                                060E  1349 ;       R5 = UCB address
                                060E  1350 ;       R4 = PCB address
                                060E  1351 ;       R3 = IRP address
                                060E  1352 ;       R1 = Quota to take
                                060E  1353 ;
                                060E  1354 ; Outputs:
                                060E  1355 ;
                                060E  1356 ;       R0 = Status code
                                060E  1357 ;
                                060E  1358 ;       All other registers are preserved.
                                060E  1359 ;-
                                060E  1360 DLE$FDT_BYTQUO::                             ; Get non-paged pool quota
                    57    DD    060E  1361          PUSHL   R7                          ; Save registers
            57  0080  C4    D0  0610  1362          MOVL    PCB$L_JIB(R4),R7            ; Get JIB
              20 A7     51    D1  0615  1363 10$:     CMPL    R1,JIB$L_BYTCNT(R7)         ; Enough quota left ?
                        21    1B  0619  1364          BLEQU   20$                         ; If LEQU then yes
              24 A7     51    D1  061B  1365          CMPL    R1,JIB$L_BYTLM(R7)          ; Is it worth waiting
                        2A    1A  061F  1366          BGTRU   30$                         ; If GTRU then waiting won't help
                        0A    E0  0621  1367          BBS     #PCB$V_SSRWAIT,-            ; If BS then wait mode disabled
                  25 24 A4        0623  1368           PCB$L_STS(R4),30$          ;
                   50   01    D0  0626  1369          MOVL    S^#RSN$_ASTWAIT,R0          ; Set resource to wait for
                                0629  1370                                              ;
                                0629  1371          SETIPL  #IPL$_ASTDEL                ; Set PSL for wait state
                        7E    DC  062C  1372          MOVPSL  -(SP)                       ; Save PSL for call to scheduler
                                062E  1373          ASSUME  IPL$_SYNCH  LE  NET$C_IPL
                                062E  1374          SETIPL  #IPL$_SYNCH                 ; Syncronize for call to SCH$RWAIT
                00000000'GF     16  0631  1375          JSB     G^SCH$RWAIT                 ; Wait for resource
                                0637  1376          SETIPL  #NET$C_IPL                  ; Restore IPL
                        D9    11  063A  1377          BRB     10$                         ; Try again
                                063C  1378
              20 A7     51    C2  063C  1379 20$:     SUBL    R1,JIB$L_BYTCNT(R7)         ; Consume bytes
              30 A3     51    B0  0640  1380          MOVW    R1,IRP$W_BOFF(R3)           ; Save quota taken from JIB
                   50   01    D0  0644  1381          MOVL    S^#SS$_NORMAL,R0            ; Indicate success
                        57  8ED0  0647  1382 90$:     POPL    R7                          ; Restore registers
                            05    064A  1383          RSB                                 ; Done
                                064B  1384
            50    2A14  8F    3C  064B  1385 30$:     MOVZWL  #SS$_EXBYTLM,R0             ; Setup error status
                        F5    11  0650  1386          BRB     90$
```

```
                     0652  1388          .SBTTL  DLE$XMT_MSG     - Send message over direct-accessed circuit
                     0652  1389 ;+
                     0652  1390 ; DLE$XMT_MSG   - Send message over direct-accessed circuit
                     0652  1391 ;
                     0652  1392 ; This routine is called whenever a new transmit IRP is placed on the
                     0652  1393 ; user transmit queue.
                     0652  1394 ;
                     0652  1395 ; Inputs:
                     0652  1396 ;
                     0652  1397 ;       R6 = DWB address
                     0652  1398 ;
                     0652  1399 ; Outputs:
                     0652  1400 ;
                     0652  1401 ;       None - message is queued to datalink layer.
                     0652  1402 ;
                     0652  1403 ;       R0-R2 are destroyed.
                     0652  1404 ;-
                     0652  1405 DLE$XMT_MSG::
    00F8 8F    BB    0652  1406          PUSHR   #^M<R3,R4,R5,R6,R7>     ; Save registers
                     0656  1407          ;
                     0656  1408          ;   Get next user transmit to process, if any are pending.
                     0656  1409          ;
 57   24 B6    0F    0656  1410          REMQUE  @DWB$Q_USER_XMT(R6),R7  ; Get next user transmit request
        0E    1D    065A  1411          BVS     19$                     ; Exit if none
                     065C  1412          ;
                     065C  1413          ;   Allocate and initialize an IRP to send to the datalink layer
                     065C  1414          ;
 51   C4 8F    9A    065C  1415          MOVZBL  #IRP$C_LENGTH,R1        ; Set length of block to allocate
       0255    30    0660  1416          BSBW    DLE$ALUNPGD_Z           ; Allocate/zero the IRP
     07 50     E8    0663  1417          BLBS    R0,10$                  ; Branch if got it
24 A6   67     0E    0666  1418          INSQUE  (R7),DWB$Q_USER_XMT(R6) ; Put IRP back on the queue
      007E     31    066A  1419 19$:     BRW     90$                     ; Exit without doing anything
   53   52     D0    066D  1420 10$:     MOVL    R2,R3                   ; Copy block address
0A A3   0A     90    0670  1421          MOVB    #DYN$C_IRP,IRP$B_TYPE(R3) ; Set block type
0B A3   08     90    0674  1422          MOVB    #NET$C_IPL,IRP$B_RMOD(R3) ; Set driver IPL
0C A3   F0'AF  9E    0678  1423          MOVAB   B^DLE$XMT_DONE,IRP$L_PID(R3) ; Set I/O post-processing routine
14 A3   56     D0    067D  1424          MOVL    R6,IRP$L_ASTPRM(R3)     ; Save DWB address in IRP
      48 A6    D0    0681  1425          MOVL    DWB$L_DLL_UCB(R6),-     ; Set address of datalink UCB
      1C A3          0684  1426                  IRP$L_UCB(R3)
         20    B0    0686  1427          MOVW    S^#IO$_WRITELBLK,-      ; Set I/O function code
      20 A3          0688  1428                  IRP$W_FUNC(R3)
23 A3   1F     90    068A  1429          MOVB    #31,IRP$B_PRI(R3)       ; Use lowest priority
      4C A6    AE    068E  1430          MNEGW   DWB$W_DLL_CHAN(R6),-    ; Set channel to datalink
      28 A3          0691  1431                  IRP$W_CHAN(R3)
      40 A6    D0    0693  1432          MOVL    DWB$G_REMNOD(R6),-      ; Set remote NI address if datalink
      40 A3          0696  1433                  IRP$Q_STATION(R3)
      44 A6    B0    0698  1434          MOVW    DWB$G_REMNOD+4(R6),-    ; (no-op if datalink not a UNA)
      44 A3          069B  1435                  IRP$Q_STATION+4(R3)
                     069D  1436          ;
                     069D  1437          ;   Attach the user's message to the datalink IRP
                     069D  1438          ;
54   2C A7    D0    069D  1439          MOVL    IRP$L_SVAPTE(R7),R4     ; Get CXB address
24 A3   54     D0    06A1  1440          MOVL    R4,IRP$L_IOSB(R3)       ; Set IOSB to buffer address
      32 A7    B0    06A5  1441          MOVW    IRP$W_BCNT(R7),-        ; Copy length of message
      32 A3          06A8  1442                  IRP$W_BCNT(R3)
         02    E1    06AA  1443          BBC     #DWB$V_DLL_XBF,-        ; Branch if datalink does direct xmits
   0D 0E A6          06AC  1444                  DWB$W_FLAGS(R6),20$
```

```
                          06AF  1445          ;
                          06AF  1446          ;   The datalink does buffered transmits
                          06AF  1447          ;
                01   B0  06AF  1448          MOVW    #IRP$M_BUFIO,-          ; Assume datalink does buffered
              2A A3       06B1  1449                  IRP$W_STS(R3)           ; transmits
          2C A3   54   D0  06B3  1450          MOVL    R4,IRP$L_SVAPTE(R3)     ; Set buffer address
              30 A3   B4   06B7  1451          CLRW    IRP$W_BOFF(R3)          ; Clear BOFF
                 1B   11   06BA  1452          BRB     30$
                          06BC  1453          ;
                          06BC  1454          ;   The datalink does direct transmits
                          06BC  1455          ;
              2A A3   B4   06BC  1456 20$:     CLRW    IRP$W_STS(R3)           ; Clear BUFIO flag
                 09   EF   06BF  1457          EXTZV   #VA$V_VPN,-             ; Get virtual page number
        51    64    15   06C1  1458                  #VA$S_VPN,(R4),R1
    52   00000000'GF   D0  06C4  1459          MOVL    G^MMG$GL_SPTBASE,R2     ; Get base of system page table
          2C A3   6241   DE  06CB  1460          MOVAL   (R2)[R1],IRP$L_SVAPTE(R3) ; Set PTE address
        64   FE00 8F   AB  06D0  1461          BICW3   #^C<VA$M_BYTE>,(R4),-   ; Set byte offset within page
              30 A3       06D5  1462                  IRP$W_BOFF(R3)
                          06D7  1463          ;
                          06D7  1464          ;   Enter the user's request onto the "transmit pending" queue
                          06D7  1465          ;
              2C A7   D4  06D7  1466 30$:     CLRL    IRP$L_SVAPTE(R7)        ; Mark buffer no longer accessible
           30 B6    67   0E  06DA  1467          INSQUE  (R7),@DWB$Q_XMT_PND+4(R6) ; Insert on "transmit pending" queue
                          06DE  1468          ;
                          06DE  1469          ;   Queue the request to the datalink driver's altstart entry point
                          06DE  1470          ;
              47 A6   96  06DE  1471          INCB    DWB$B_IRPCNT(R6)        ; Increment outstanding I/O count
        55    1C A3   D0  06E1  1472          MOVL    IRP$L_UCB(R3),R5        ; Setup datalink UCB address
     00000000'GF    16   06E5  1473          JSB     G^EXE$ALTQUEPKT         ; Queue the packet to the datalink
          00F8 8F   BA  06EB  1474 90$:     POPR    #^M<R3,R4,R5,R6,R7>     ; Restore registers
                    05  06EF  1475          RSB
```

```
                        06F0  1477          .SBTTL  DLE$XMT_DONE    - Transmit I/O post-processing
                        06F0  1478 ;+
                        06F0  1479 ; DLE$XMT_DONE - Transmit I/O post-processing
                        06F0  1480 ;
                        06F0  1481 ; This routine is called by IOPOST when a transmit request is completed.
                        06F0  1482 ; This is because the address of this routine was put into the IRP PID
                        06F0  1483 ; field before issuing the transmit.  The action here is to complete the
                        06F0  1484 ; user request, and deallocate the datalink IRP and buffer.
                        06F0  1485 ;
                        06F0  1486 ; Inputs:
                        06F0  1487 ;
                        06F0  1488 ;       R5 = IRP address
                        06F0  1489 ;
                        06F0  1490 ;       IPL = IPL$_POST (4)
                        06F0  1491 ;
                        06F0  1492 ; Outputs:
                        06F0  1493 ;
                        06F0  1494 ;       None
                        06F0  1495 ;-
                        06F0  1496 DLE$XMT_DONE::
                        06F0  1497          DSBINT  #NET$C_IPL              ; Raise to driver IPL
          56    DD      06F6  1498          PUSHL   R6                      ; Save registers
    50    24 A5 D0      06F8  1499          MOVL    IRP$L_IOSB(R5),R0       ; Get CXB address
                        06FC  1500          ;
                        06FC  1501          ;   Locate user's original transmit IRP, and copy the completion
                        06FC  1502          ;   status to it.
                        06FC  1503          ;
    51    14 A0 D0      06FC  1504          MOVL    CXB$L_IRP(R0),R1        ; Get user IRP address
          38 A5 D0      0700  1505          MOVL    IRP$L_IOST1(R5),-       ; Copy I/O status and xfer size
          38 A1         0703  1506                  IRP$L_IOST1(R1)         ;
                        0705  1507          ;
                        0705  1508          ;   Deallocate the CXB attached to the datalink IRP
                        0705  1509          ;
00000000'GF   16        0705  1510          JSB     G^COM$DRVDEALMEM        ; Deallocate the CXB
                        070B  1511          ;
                        070B  1512          ;   Deallocate the datalink IRP
                        070B  1513          ;
    50    55    D0      070B  1514          MOVL    R5,R0                   ; Set address of datalink IRP
00000000'GF   16        070E  1515          JSB     G^COM$DRVDEALMEM        ; Deallocate the datalink IRP
                        0714  1516          ;
                        0714  1517          ;   Make sure user transmits are completed in order, as a sanity check.
                        0714  1518          ;
 56   18 A1 D0          0714  1519          MOVL    IRP$L_WIND(R1),R6       ; Get associated DWB
 2C A6    51 D1         0718  1520          CMPL    R1,DWB$Q_XMT_PND(R6)    ; Is this the first IRP in the list ?
          04 13         071C  1521          BEQL    10$                     ; If EQL then yes
                        071E  1522          BUG_CHECK  NETNOSTATE,FATAL     ; DLE transmit completed out of order
                        0722  1523 10$:     ;
                        0722  1524          ;   Remove the user's IRP from the "transmit pending" queue,
                        0722  1525          ;   and complete it.
                        0722  1526          ;
    53    61    0F      0722  1527          REMQUE  (R1),R3                 ; Remove IRP
 55   1C A3 D0          0725  1528          MOVL    IRP$L_UCB(R3),R5        ; Get corresponding UCB address
00000000'GF   16        0729  1529          JSB     G^COM$POST              ; Complete the user's I/O request
                        072F  1530          ;
                        072F  1531          ;   Decrement the count of the number of outstanding datalink I/Os
                        072F  1532          ;
      47 A6   97        072F  1533          DECB    DWB$B_IRPCNT(R6)        ; Decrement outstanding I/O refcnt
```

```
                     0732  1534          ;
                     0732  1535          ;   If the DWB has become inactive, and was waiting for this
                     0732  1536          ;   I/O to complete before doing so, then do not issue any further
                     0732  1537          ;   I/O.
                     0732  1538          ;
          00    E0   0732  1539          BBS     #DWB$V_RUN,-                ; If DWB no longer active,
05 0E A6             0734  1540                  DWB$W_FLAGS(R6),20$
      FB84    30     0737  1541          BSBW    DEALLOC_DWB                 ; then try to deallocate DWB
        03    11     073A  1542          BRB     50$                         ; and exit without any more I/O
                     073C  1543 20$:     ;
                     073C  1544          ;   Transmit another user message, if there is one waiting
                     073C  1545          ;
      FF13    30     073C  1546          BSBW    DLE$XMT_MSG                 ; Transmit user message, if possible
        56 8ED0      073F  1547 50$:     POPL    R6                          ; Restore registers
                     0742  1548          ENBINT                              ; Restore IOPOST IPL
              05     0745  1549          RSB                                 ; Done
```

```
                                0746  1551          .SBTTL  INIT_RCV_IRP    - Initialize datalink receive IRP
                                0746  1552 ;+
                                0746  1553 ; INIT_RCV_IRP - Initialize datalink receive IRP
                                0746  1554 ;
                                0746  1555 ; This routine is called to allocate and initialize a new read request IRP
                                0746  1556 ; to the datalink driver.
                                0746  1557 ;
                                0746  1558 ; Inputs:
                                0746  1559 ;
                                0746  1560 ;       R6 = DWB address
                                0746  1561 ;
                                0746  1562 ; Outputs:
                                0746  1563 ;
                                0746  1564 ;       R0 = Status code
                                0746  1565 ;       R3 = IRP address
                                0746  1566 ;
                                0746  1567 ;       R1 is destroyed.
                                0746  1568 ;-
                                0746  1569 INIT_RCV_IRP::
                    52    DD    0746  1570          PUSHL   R2                      ; Save registers
                                0748  1571          ;
                                0748  1572          ;   Allocate and initialize an IRP to send to the datalink layer
                                0748  1573          ;
                51  C4 8F   9A  0748  1574          MOVZBL  #IRP$C_LENGTH,R1        ; Set length of block to allocate
                     0169   30  074C  1575          BSBW    DLE$ALONPGD_Z           ; Allocate/zero the IRP
                   3C 50    E9  074F  1576          BLBC    R0,90$                  ; Exit if no memory
                53  52      D0  0752  1577          MOVL    R2,R3                   ; Copy block address
              0A A3  0A      90  0755  1578          MOVB    #DYN$C_IRP,IRP$B_TYPE(R3) ; Set block type
              0B A3  08      90  0759  1579          MOVB    #NET$C_IPL,IRP$B_RMOD(R3) ; Set driver IPL
     0C A3  00000080D'EF     9E  075D  1580          MOVAB   DLE$RCV_MSG,IRP$L_PID(R3) ; Set I/O post-processing routine
              14 A3  56      D0  0765  1581          MOVL    R6,IRP$L_ASTPRM(R3)     ; Save DWB address in IRP
                     21      B0  0769  1582          MOVW    S^#IO$_READLBLK,-       ; Set I/O function code
                   20 A3             076B  1583                  IRP$W_FUNC(R3)
                     03      B0  076D  1584          MOVW    #IRP$M_FUNC!IRP$M_BUFIO,- ; Setup STS for buffered reads
                   2A A3             076F  1585                  IRP$W_STS(R3)
                                0771  1586          ;
                                0771  1587          ;   Setup diagnostics buffer and attach it to the receive IRP
                                0771  1588          ;   so that we can get back the NI datalink header to return
                                0771  1589          ;   to the DLE user, if desired.  Rather than allocate another
                                0771  1590          ;   block from pool, we re-use the end of the IRP as a diagnostics
                                0771  1591          ;   buffer.
                                0771  1592          ;
15 0E A6    03      E1          0771  1593          BBC     #DWB$V_BC,DWB$W_FLAGS(R6),20$ ; Skip if not an NI
                                0776  1594          ASSUME  IRP$C_LENGTH-IRP$C_STDLEN GE DIAG_C_LENGTH
   50    5C A3      9E          0776  1595          MOVAB   IRP$C_STDLEN(R3),R0     ; Get address of diagnostics buffer
                1A    B0        077A  1596          MOVW    #DIAG_C_LENGTH,-        ; Set size of diagnostics buffer
                   08 A0             077C  1597                  DIAG_W_SIZE(R0)
                   0C A0      9E     077E  1598          MOVAB   DIAG_G_DATA(R0),-       ; Make pointer to data area
                      60             0781  1599                  DIAG_L_DATA(R0)
   4C A3    50    D0          0782  1600          MOVL    R0,IRP$L_DIAGBUF(R3)    ; Set address of diagnostics buffer
                                0786  1601          SETBIT  #IRP$V_DIAGBUF,-        ; Indicate diagnostics buffer present
                                0786  1602                  IRP$W_STS(R3)
      50    01      9A          078B  1603 20$:     MOVZBL  S^#SS$_NORMAL,R0        ; Success
            52 8ED0             078E  1604 90$:     POPL    R2                      ; Restore registers
                  05            0791  1605          RSB                             ; Exit with status
```

```
                     0792  1607          .SBTTL  ISSUE_DLL_RCV   - Issue datalink receive request
                     0792  1608 ;+
                     0792  1609 ; ISSUE_DLL_RCV - Re-Issue receive request to datalink layer
                     0792  1610 ;
                     0792  1611 ; This routine is called to re-issue a read request to the datalink driver
                     0792  1612 ; to obtain more incoming DLE messages.  An IRP is passed to this routine
                     0792  1613 ; which is assumed to have already been setup as a datalink receive IIRP.
                     0792  1614 ; A CXB buffer may optionally still be attached to the IRP - if so, it will
                     0792  1615 ; be re-used by the datalink driver.
                     0792  1616 ;
                     0792  1617 ; Inputs:
                     0792  1618 ;
                     0792  1619 ;       R6 = DWB address
                     0792  1620 ;       R3 = IRP address to be re-cycled
                     0792  1621 ;
                     0792  1622 ; Outputs:
                     0792  1623 ;
                     0792  1624 ;       R0 = Status code
                     0792  1625 ;
                     0792  1626 ;       Nc registers are destroyed.
                     0792  1627 ;-
                     0792  1628 ISSUE_DLL_RCV::
          55    DD   0792  1629          PUSHL   R5                      ; Save registers
                     0794  1630          ;
                     0794  1631          ;   Reset the UCB/CHAN pair each time we send an IRP to the datalink
                     0794  1632          ;   because it is possible that the UCB address might have changed
                     0794  1633          ;   since the last receive if the user did a UNA LIMITED SETMODE
                     0794  1634          ;   which causes a different UCB to be used than the original.
                     0794  1635          ;
    48 A6    D0      0794  1636          MOVL    DWB$L_DLL_UCB(R6),-     ; Set address of datalink UCB
       1C A3         0797  1637                  IRP$L_UCB(R3)
       4C A6    AE   0799  1638          MNEGW   DWB$W_DLL_CHAN(R6),-    ; Set channel to datalink
       28 A3         079C  1639                  IRP$W_CHAN(R3)
          01    E1   079E  1640          BBC     #DWB$V_DLL_RBF,-        ; Branch if receiver uses direct I/O
    08 0E A6         07A0  1641                  DWB$W_FLAGS(R6),20$
                     07A3  1642          ;
                     07A3  1643          ;   The datalink does buffered receives
                     07A3  1644          ;
32 A3    3FFF 8F     07A3  1645          MOVW    #^X<3FFF>,IRP$W_BCNT(R3) ; Accept infinite size messages
             4E   11 07A9  1646          BRB     30$
                     07AB  1647          ;
                     07AB  1648          ;   The datalink does direct I/O receives.  Allocate a CXB for the
                     07AB  1649          ;   DMA receive, and attach it to the IRP.
                     07AB  1650          ;
  52   24 A3    D0   07AB  1651 20$:     MOVL    IRP$L_IOSB(R3),R2       ; Is there a buffer already allocated?
             1F   12 07AF  1652          BNEQ    25$                     ; Branch if so
  51   084C 8F     3C 07B1  1653          MOVZWL  #CXB$C_OVERHEAD+-       ; Compute total buffer size
                     07B6  1654                  MAX_DIR_RCV,R1
         0110        30 07B6  1655          BSBW    DLE$ALONONPAGED         ; Allocate nonpaged buffer
         4D 50       E9 07B9  1656          BLBC    R0,90$                  ; Branch if error detected
            3E       BB 07BC  1657          PUSHR   #^M<R1,R2,R3,R4,R5>     ; Save critical regs destroyed by MOVC
00   6E    00       2C 07BE  1658          MOVC5   #0,(SP),#0,-            ; Zero the CXB header
62   0048 8F          07C2  1659                  #CXB$C_HEADER,(R2)      ;
            3E       BA 07C6  1660          POPR    #^M<R1,R2,R3,R4,R5>     ; Restore regs
  08 A2    51       B0 07C8  1661          MOVW    R1,CXB$W_SIZE(R2)       ; Store size for deallocation
  0A A2    1B       90 07CC  1662          MOVB    #DYN$C_CXB,CXB$B_TYPE(R2) ; Setup structure type
  62   48 A2       9E 07D0  1663 25$:     MOVAB   CXB$C_HEADER(R2),(R2)   ; Store data area ptr in CXB
```

```
                24 A3    52    D0  07D4  1664          MOVL    R2,IRP$L_IOSB(R3)       ; Store CXB address in IRP
                   2A A3      B4  07D8  1665          CLRW    IRP$W_STS(R3)           ; Clear BUFIO flag
                      09      EF  07DB  1666          EXTZV   #VA$V_VPN,-             ; Get virtual page number
             51    62    15        07DD  1667                  #VA$S_VPN,(R2),R1
          52   00000000'GF    D0  07E0  1668          MOVL    G^MMG$GL_SPTBASE,R2     ; Get base of system page table
             2C A3    6241    DE  07E7  1669          MOVAL   (R2)[R1],IRP$L_SVAPTE(R3) ; Set PTE address
             62    FE00 8F    AB  07EC  1670          BICW3   #^C<VA$M_BYTE>,(R2),-   ; Set byte offset within page
                      30 A3      07F1  1671                  IRP$W_BOFF(R3)
                   0800 8F    B0  07F3  1672          MOVW    #MAX_DIR_RCV,-          ; Set maximum receive size
                      32 A3      07F7  1673                  IRP$W_BCNT(R3)
                                 07F9  1674          ;
                                 07F9  1675          ;   Queue the request to the datalink driver
                                 07F9  1676          ;
                      47 A6    96  07F9  1677 30$:    INCB    DWB$B_IRPCNT(R6)        ; Increment outstanding I/O count
                55    1C A3    D0  07FC  1678          MOVL    IRP$L_UCB(R3),R5        ; Get datalink UCB address
                00000000'GF    16  0800  1679          JSB     G^EXE$ALTQUEPKT         ; Queue request to datalink driver
                   50    01    D0  0806  1680          MOVL    S^#SS$_NORMAL,R0        ; Success
                         55 8ED0  0809  1681 90$:    POPL    R5                      ; Restore registers
                            05    080C  1682          RSB
```

```
                     080D  1684          .SBTTL  DLE$RCV_MSG     - Receive a message from datalink
                     080D  1685 ;+
                     080D  1686 ; DLE$RCV_MSG   - Receive a message from the datalink layer
                     080D  1687 ;
                     080D  1688 ; Inputs:
                     080D  1689 ;
                     080D  1690 ;       R5 = IRP address
                     080D  1691 ;
                     080D  1692 ;       IPL = IPL$_POST (4)
                     080D  1693 ;
                     080D  1694 ; Outputs:
                     080D  1695 ;
                     080D  1696 ;       None
                     080D  1697 ;-
                     080D  1698 DLE$RCV_MSG::                           ; Receive message from datalink
                     080D  1699          DSBINT  #NET$C_IPL             ; Raise to driver IPL
          56    DD   0813  1700          PUSHL   R6                     ; Save registers
56   14 A5    D0   0815  1701          MOVL    IRP$L_ASTPRM(R5),R6    ; Get DWB address
     47 A6    97   0819  1702          DECB    DWB$B_IRPCNT(R6)       ; Decrement outstanding I/O refcnt
                     081C  1703          ;
                     081C  1704          ;   If the receive was buffered I/O, then save the address of the
                     081C  1705          ;   CXB allocated by the datalink in a consistent spot for all types
                     081C  1706          ;   of datalinks.
                     081C  1707          ;
          01    E1   081C  1708          BBC     #DWB$V_DLL_RBF,-       ; If buffered I/O,
05 0E A6             081E  1709                  DWB$W_FLAGS(R6),5$
     2C A5    D0   0821  1710          MOVL    IRP$L_SVAPTE(R5),-     ; Copy the CXB address to a
     24 A5         0824  1711                  IRP$L_IOSB(R5)         ; consistent place in the IRP
                     0826  1712 5$:      ;
                     0826  1713          ;   Dequeue the next user receive IRP, and copy the message and
                     0826  1714          ;   completion status to it.  Then re-issue the read request to
                     0826  1715          ;   the datalink driver.
                     0826  1716          ;
53   1C B6    0F   0826  1717          REMQUE  @DWB$Q_USER_RCV(R6),R3 ; Dequeue next user receive IRP
          17    1D   082A  1718          BVS     70$                    ; If none, bugcheck - should be one
                     082C  1719                                         ; for every datalink I/O
          19    10   082C  1720          BSBB    RCV_DONE               ; Copy msg to user IRP and post it
                     082E  1721          ;
                     082E  1722          ;   Deallocate the datalink IRP and message.
                     082E  1723          ;
51    55    D0   082E  1724          MOVL    R5,R1                  ; Set datalink receive IRP address
     FAC2    30   0831  1725          BSBW    DEALLOC_MSG            ; Deallocate received message IRP/CXB
                     0834  1726          ;
                     0834  1727          ;   If the DWB has become inactive, and was waiting for this
                     0834  1728          ;   I/O to complete before doing so, then try to cleanup DWB.
                     0834  1729          ;
          00    E0   0834  1730          BBS     #DWB$V_RUN,-           ; If DWB no longer active,
03 0E A6             0836  1731                  DWB$W_FLAGS(R6),50$
     FA82    30   0839  1732          BSBW    DEALLOC_DWB            ; then try to deallocate DWB
          56 8ED0   083C  1733 50$:    POPL    R6                     ; Restore registers
                     083F  1734          ENBINT                         ; Restore IPL
                05   0842  1735          RSB
                     0843  1736          ;
                     0843  1737          ;   We have just received an IRP from the datalink, but there is
                     0843  1738          ;   no user request pending.  This should never happen, since for
                     0843  1739          ;   every user request, a corresponding datalink request is made.
                     0843  1740          ;
```

```
0843  1741 70$:     BUG_CHECK NETNOSTATE,FATAL
```

```
                     0847  1743          .SBTTL  RCV_DONE        - Complete User Receive IRP
                     0847  1744 ;+
                     0847  1745 ; RCV_DONE - Complete user receive IRP
                     0847  1746 ;
                     0847  1747 ; This routine is called to transfer the message from the datalink IRP
                     0847  1748 ; to the user's IRP, and to post the user's IRP.
                     0847  1749 ;
                     0847  1750 ; Inputs:
                     0847  1751 ;
                     0847  1752 ;       R5 = Datalink IRP
                     0847  1753 ;       R3 = User IRP
                     0847  1754 ;
                     0847  1755 ; Outputs:
                     0847  1756 ;
                     0847  1757 ;       R3 = 0 to indicate user IRP is posted.
                     0847  1758 ;
                     0847  1759 ;       R0 is destroyed.
                     0847  1760 ;-
                     0847  1761 RCV_DONE:                                   ; Complete User receive IRP
           3E   BB   0847  1762          PUSHR   #^M<R1,R2,R3,R4,R5>     ; Save regs
           07   E1   0849  1763          BBC     #IRP$V_DIAGBUF,-        ; If diagnostics buffer present,
     17 2A A3        084B  1764                  IRP$W_STS(R3),5$
           07   E1   084E  1765          BBC     #IRP$V_DIAGBUF,-        ; and diagnostics returned by datalink
     12 2A A5        0850  1766                  IRP$W_STS(R5),5$
     50    4C A5    D0  0853  1767          MOVL    IRP$L_DIAGBUF(R5),R0    ; Get address of datalink diag buffer
     51    4C A3    D0  0857  1768          MOVL    IRP$L_DIAGBUF(R3),R1    ; Get address of user diag buffer
           28   BB   085B  1769          PUSHR   #^M<R3,R5>              ; Save registers
           0E   28   085D  1770          MOVC    #DIAG_C_NIHDRSIZ,-      ; Copy NI datalink header
        00 B0        085F  1771                  @DIAG_L_DATA(R0),-
        00 B1        0861  1772                  @DIAG_L_DATA(R1)
           28   BA   0863  1773          POPR    #^M<R3,R5>              ; Restore registers
        38 A5    D0  0865  1774 5$:     MOVL    IRP$L_IOST1(R5),-       ; Copy I/O status and bytes xferred
        38 A3        0868  1775                  IRP$L_IOST1(R3)         ;
     50    3A A5    3C  086A  1776          MOVZWL  IRP$L_IOST1+2(R5),R0    ; Get bytes received
     32 A3    50    B1  086E  1777          CMPW    R0,IRP$W_BCNT(R3)       ; Bigger than user buffer?
           0A   1B   0872  1778          BLEQU   10$                     ; If LEQU no
     50    32 A3    3C  0874  1779          MOVZWL  IRP$W_BCNT(R3),R0       ; Use user buffer size
        0838 8F     B0  0878  1780          MOVW    #SS$_DATAOVERUN,-       ; Setup error code
        38 A3        087C  1781                  IRP$L_IOST1(R3)         ;
     32 A3    50    B0  087E  1782 10$:    MOVW    R0,IRP$W_BCNT(R3)       ; Setup bytes to xfer to user buffer
     3A A3    50    B0  0882  1783          MOVW    R0,IRP$L_IOST1+2(R3)    ; ...and in IOSB image as well
           10   13   0886  1784          BEQL    20$                     ; If EQL then no data to move
     51    24 B5    D0  0888  1785          MOVL    @IRP$L_IOSB(R5),R1      ; Point to datalink buffer's data area
     52    2C B3    D0  088C  1786          MOVL    @IRP$L_SVAPTE(R3),R2    ; Point to user buffer's data area
  62    61    50    28  0890  1787          MOVC3   R0,(R1),(R2)            ; Move data
     53    08 AE    D0  0894  1788          MOVL    8(SP),R3                ; Restore user IRP address
     55    1C A3    D0  0898  1789 20$:    MOVL    IRP$L_UCB(R3),R5        ; Get user's UCB address
  00000000'GF    16   089C  1790          JSB     G^COM$POST              ; Complete user I/O
           3E   BA   08A2  1791          POPR    #^M<R1,R2,R3,R4,R5>     ; Restore regs
           53   D4   08A4  1792          CLRL    R3                      ; Indicate user IRP gone
                05   08A6  1793          RSB                             ; Done
```

```
                     08A7  1795          .SBTTL  UNIT_INIT       - Unit initialization
                     08A7  1796 ;+
                     08A7  1797 ; UNIT_INIT - Unit initialization
                     08A7  1798 ;
                     08A7  1799 ; This routine is called by SYSGEN when a new unit (UCB) is added.
                     08A7  1800 ;
                     08A7  1801 ; Inputs:
                     08A7  1802 ;
                     08A7  1803 ;       R5 = UCB address
                     08A7  1804 ;
                     08A7  1805 ; Outputs:
                     08A7  1806 ;
                     08A7  1807 ;       None
                     08A7  1808 ;
                     08A7  1809 ;       All registers are preserved.
                     08A7  1810 ;-
                     08A7  1811 UNIT_INIT:
          50    DD   08A7  1812          PUSHL   R0                      ; Save registers
50   0090 C5    9E   08A9  1813          MOVAB   UCB$Q_DWB_LIST(R5),R0   ; Get address of listhead
     60   50    D0   08AE  1814          MOVL    R0,(R0)                 ; Initialize listhead
     60   80    DE   08B1  1815          MOVAL   (R0)+,(R0)
          50 8ED0   08B4  1816          POPL    R0                      ; Restore registers
             05     08B7  1817          RSB
```

```
                        08B8  1819          .SBTTL  DLE$ALONPGD_Z   - Allocate and zero from system pool
                        08B8  1820          .SBTTL  DLE$ALONONPAGED - Allocate from system pool
                        08B8  1821
                        08B8  1822 ;++
                        08B8  1823 ; DLE$ALONPGD_Z    - Allocate and zero system non-paged buffer
                        08B8  1824 ; DLE$ALONONPAGED  - Allocate buffer from system non-paged pool
                        08B8  1825 ;
                        08B8  1826 ; A buffer is allocated from non-paged pool and its size field is set to
                        08B8  1827 ; the size requested.  Its type field is set to DYN$C_CXB.
                        08B8  1828 ;
                        08B8  1829 ; Inputs:
                        08B8  1830 ;
                        08B8  1831 ;       R1 = Size of block to be allocated in bytes
                        08B8  1832 ;
                        08B8  1833 ; Outputs:
                        08B8  1834 ;
                        08B8  1835 ;       R0 = Status
                        08B8  1836 ;       R2 = Address of block if successful, else zero
                        08B8  1837 ;
                        08B8  1838 ;       All other registers are preserved.
                        08B8  1839 ;-
                        08B8  1840          .ENABL  LSB
                        08B8  1841
                        08B8  1842 DLE$ALONPGD_Z::                          ; Allocate and zero non-paged buffer
                    0F  10    08B8  1843          BSBB    DLE$ALONONPAGED         ; Allocate the buffer
                  2A 50   E9  08BA  1844          BLBC    R0,20$                  ; If LBC then error
                    3F    BB  08BD  1845          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save regs
    62  51  00  6E   00    2C  08BF  1846          MOVC5   #0,(SP),#0,R1,(R2)      ; Zero the entire buffer
                    3F    BA  08C5  1847          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore regs
                    16    11  08C7  1848          BRB     10$                     ; Setup the type and size fields (again)
                        08C9  1849
                        08C9  1850
                        08C9  1851 DLE$ALONONPAGED::                        ; Allocate non-paged memory
                    0A    BB  08C9  1852          PUSHR   #^M<R1,R3>              ; Save regs
    00000000'GF   16    08CB  1853          JSB     G^EXE$ALONONPAGED       ; Allocate memory
                    0A    BA  08D1  1854          POPR    #^M<R1,R3>              ; Restore regs
                  09 50   E8  08D3  1855          BLBS    R0,10$                  ; If LBS then success
50  0124 8F   3C    08D6  1856          MOVZWL  #SS$_INSFMEM,R0         ; Return insufficient memory
                    52    D4  08DB  1857          CLRL    R2                      ; Zero the buffer pointer
                    08    11  08DD  1858          BRB     20$                     ; Take common exit
  08 A2     51    B0  08DF  1859 10$:    MOVW    R1,CXB$W_SIZE(R2)       ; Set size for deallocation
                    1B    90  08E3  1860          MOVB    #DYN$C_CXB,-            ;
                0A A2       08E5  1861                  CXB$B_TYPE(R2)          ; Set tentative buffer type
                    05    E7  08E7  1862 20$:    RSB                             ; Return with status in R0
                        08E8  1863
                        08E8  1864          .DSABL  LSB
```

```
08E8  1866
08E8  1867 ;
08E8  1868 ; Mark end of driver
08E8  1869 ;
08E8  1870
08E8  1871 DLE$END::
08E8  1872          .END
```

```
$$$                             = 00000020 R      02
$$OP                            = 00000002
ABT$L_USERVA                    = 00000004
ABD$W_COUNT                     = 00000002
ABD$W_TEXT                      = 00000000
ABORTIO                           00000110 R      03
ABORT_IRPS                        00000493 R      03
ACP$ACCESSNET                     ********    X   03
ACP$C_STA_F                     = 00000004
ACP$C_STA_H                     = 00000005
ACP$C_STA_I                     = 00000000
ACP$C_STA_N                     = 00000001
ACP$C_STA_R                     = 00000002
ACP$C_STA_S                     = 00000003
ACP$DEACCESS                      ********    X   03
ACP$MODIFY                        ********    X   03
AQB$L_ACPPID                    = 0000000C
ATS_NOLL                          ********    X   02
BUG$_NETNOSTATE                   ********    X   03
CAN$C_CANCEL                    = 00000000
CANCEL_ALL                        00000443 RG     03
CCB$L_WIND                      = 00000004
CNF$_ADVANCE                    = 00000000
CNF$_QUIT                       = 00000002
CNF$_TAKE_CURR                  = 00000003
CNF$_TAKE_PREV                  = 00000001
COM$DRVDEALMEM                    ********    X   03
COM$POST                          ********    X   03
CRB$L_INTD                      = 00000024
CXB$B_TYPE                      = 0000000A
CXB$C_HEADER                    = 00000048
CXB$C_OVERHEAD                  = 0000004C
CXB$L_IRP                       = 00000014
CXB$W_LENGTH                    = 0000000C
CXB$W_SIZE                      = 00000008
DDB$L_DDT                       = 0000000C
DDB$L_UCB                       = 00000004
DDT$L_CANCEL                    = 0000000C
DDT$L_FDT                       = 00000008
DEALLOC_DWB                       000002BE R      03
DEALLOC_MSG                       000002F6 R      03
DEV$M_AVL                       = 00040000
DEV$M_IDV                       = 04000000
DEV$M_MBX                       = 00100000
DEV$M_NET                       = 00002000
DEV$M_ODV                       = 08000000
DEV$M_SHR                       = 00010000
DEV$V_DMT                       = 00000015
DEV$V_MNT                       = 00000013
DEVNOTMNT                         00000108 R      03
DIAG_B_TYPE                       0000000A
DIAG_C_LENGTH                     0000001A
DIAG_C_NIHDRSIZ                 = 0000000E
DIAG_G_DATA                       0000000C
DIAG_L_DATA                       00000000
DIAG_L_USERBUF                    00000004
DIAG_W_SIZE                       00000008

DLE$ACCESS                        000001B4 RG     03
DLE$ALONONPAGED                   000008C9 RG     03
DLE$ALONPGD_Z                     000008B8 RG     03
DLE$CANCEL                        00000442 RG     03
DLE$CONTROL                       00000428 RG     03
DLE$DEACCESS                      0000029B RG     03
DLE$END                           000008E8 RG     03
DLE$FDT_ACCESS                    00000116 RG     03
DLE$FDT_BYTQUO                    0000060E RG     03
DLE$FDT_CONTROL                   00000419 RG     03
DLE$FDT_DEACCESS                  00000269 RG     03
DLE$FDT_RCV                       000004E7 R      03
DLE$FDT_RW                        0000052E RG     03
DLE$FDT_SETMODE                   00000327 RG     03
DLE$FDT_XMT                       00000525 R      03
DLE$LPD_DOWN                      000004AC RG     03
DLE$RCV_MSG                       0000080D RG     03
DLE$SETMODE                       00000376 RG     03
DLE$STARTIO                       00000090 RG     03
DLE$XMT_DONE                      000006F0 RG     03
DLE$XMT_MSG                       00000652 RG     03
DPT$C_LENGTH                    = 00000038
DPT$C_VERSION                   = 00000004
DPT$INITAB                        00000038 R      02
DPT$REINITAB                      0000006D R      02
DPT$TAB                           00000000 R      02
DWB$B_IRPCNT                    = 00000047
DWB$B_SUBSTA                    = 00000046
DWB$B_TYPE                      = 0000000A
DWB$C_LENGTH                    = 00000050
DWB$G_REMNOD                    = 00000040
DWB$L_DLL_UCB                   = 00000048
DWB$L_PID                       = 00000034
DWB$L_UCBO                      = 00000038
DWB$Q_RCV_MSG                   = 00000014
DWB$Q_USER_RCV                  = 0000001C
DWB$Q_USER_XMT                  = 00000024
DWB$Q_XMT_PND                   = 0000002C
DWB$V_BC                        = 00000003
DWB$V_DELETE                    = 00000004
DWB$V_DLL_RBF                   = 00000001
DWB$V_DLL_XBF                   = 00000002
DWB$V_RUN                       = 00000000
DWB$W_CHAN                      = 0000003C
DWB$W_DLL_CHAN                  = 0000004C
DWB$W_FLAGS                     = 0000000E
DWB$W_ID                        = 0000004E
DWB$W_PATH                      = 0000003E
DWB$W_REFCNT                    = 0000000C
DYN$C_BUFIO                     = 00000013
DYN$C_CRB                       = 00000005
DYN$C_CXB                       = 0000001B
DYN$C_DDB                       = 00000006
DYN$C_DPT                       = 0000001E
DYN$C_IRP                       = 0000000A
DYN$C_ORB                       = 00000049
DYN$C_UCB                       = 00000010
```

```
EXE$ABORTIO                     ********    X   03
EXE$ALLOCBUF                    ********    X   03
EXE$ALONONPAGED                 ********    X   03
EXE$ALTQUEPKT                   ********    X   03
EXE$QIOACPPKT                   ********    X   03
EXE$QIORETURN                   ********    X   03
EXE$READCHKR                    ********    X   03
EXE$WRITECHKR                   ********    X   03
FUNCTABLE                       00000038 R      03
FUNCTAB_LEN                   = 00000058
GET_P1DSC                       00000247 R      03
GET_P2DSC                       0000024C R      03
GET_P3DSC                       00000251 R      03
GET_P4DSC                       00000256 R      03
GET_STATUS                      0000040D R      03
GET_WNDSC                       00000243 R      03
INIT_RCV_IRP                    00000746 RG     03
IO$M_FCODE                    = 0000003F
IO$V_NOW                      = 00000006
IO$_ACCESS                    = 00000032
IO$_ACPCONTROL                = 00000038
IO$_DEACCESS                  = 00000034
IO$_READLBLK                  = 00000021
IO$_READVBLK                  = 00000031
IO$_SETMODE                   = 00000023
IO$_VIRTUAL                   = 0000003F
IO$_WRITELBLK                 = 00000020
IO$_WRITEVBLK                 = 00000030
IOC$INITIATE                    ********    X   03
IOC$MNTVER                      ********    X   03
IOC$REQCOM                      ********    X   03
IOC$RETURN                      ********    X   03
IPL$_ASTDEL                   = 00000002
IPL$_SYNCH                    = 00000008
IRP$B_PRI                     = 00000023
IRP$B_RMOD                    = 0000000B
IRP$B_TYPE                    = 0000000A
IRP$C_LENGTH                  = 000000C4
IRP$C_STDLEN                    0000005C
IRP$L_ARB                     = 00000058
IRP$L_ASTPRM                  = 00000014
IRP$L_DIAGBUF                 = 0000004C
IRP$L_EXTEND                  = 00000054
IRP$L_IOSB                    = 00000024
IRP$L_IOST1                   = 00000038
IRP$L_IOST2                   = 0000003C
IRP$L_PID                     = 0000000C
IRP$L_SVAPTE                  = 0000002C
IRP$L_UCB                     = 0000001C
IRP$L_WIND                    = 00000018
IRP$M_BUFIO                   = 00000001
IRP$M_FUNC                    = 00000002
IRP$Q_STATION                 = 00000040
IRP$V_COMPLX                  = 00000003
IRP$V_DIAGBUF                 = 00000007
IRP$V_FUNC                    = 00000001
IRP$V_PHYSIO                  = 00000008
IRP$W_BCNT                    = 00000032
IRP$W_BOFF                    = 00000030
IRP$W_CHAN                    = 00000028
IRP$W_FUNC                    = 00000020
IRP$W_STS                     = 0000002A
ISSUE_DLL_RCV                   00000792 RG     03
JIB$L_BYTCNT                  = 00000020
JIB$L_BYTLM                   = 00000024
JIB$W_FILCNT                  = 00000030
MASKH                         = 01000000
MASKL                         = 00000000
MAX_DIR_RCV                   = 00000800
MMG$GL_SPTBASE                  ********    X   03
ND$DDT                          00000000 RG     03
NET$C_ACT_TIMER               = 0000001E
NET$C_EFN_ASYN                = 00000002
NET$C_EFN_WAIT                = 00000001
NET$C_IPL                     = 00000008
NET$C_MAXACCFLD               = 00000027
NET$C_MAXLINNAM               = 0000000F
NET$C_MAXLNK                  = 000003FF
NET$C_MAXNODNAM               = 00000006
NET$C_MAXOBJNAM               = 0000000C
NET$C_MAX_AREAS               = 0000003F
NET$C_MAX_LINES               = 00000040
NET$C_MAX_NCB                 = 0000006E
NET$C_MAX_NODES               = 000003FF
NET$C_MAX_OBJ                 = 000000FF
NET$C_MAX_WQE                 = 00000014
NET$C_MINBUFSIZ               = 000000C0
NET$C_TID_ACT                 = 00000003
NET$C_TID_RUS                 = 00000001
NET$C_TID_XRT                 = 00000002
NET$C_TRCTL_CEL               = 00000002
NET$C_TRCTL_OVR               = 00000005
NET$C_UTLBUFSIZ               = 00001000
NET$M_MAXLNKMSK               = 000003FF
NOOPER                          00000104 R      03
NSP$C_EXT_LNK                 = 0000001E
NSP$C_MAXADR                  = 00000009
ORB$B_FLAGS                   = 0000000B
ORB$L_OWNER                   = 00000000
ORB$M_PROT_16                 = 00000001
ORB$W_PROT                    = 00000018
P1                            = 00000000
P2                            = 00000004
P3                            = 00000008
P4                            = 0000000C
P5                            = 00000010
P6                            = 00000014
PCB$L_JIB                     = 00000080
PCB$L_PID                     = 00000060
PCB$L_STS                     = 00000024
PCB$Q_PRIV                    = 00000084
PCB$V_SSRWAIT                 = 0000000A
PR$_IPL                       = 00000012
PROC_IO                         000000BC R      03
```

```
PRV$V_OPER                      = 00000012
RCB$W_MCOUNT                    = 00000054
RCV_DONE                          00000847 R      03
RESTORE_QUOTA                     00000308 R      03
RSN$_ASTWAIT                    = 00000001
SCH$GL_PCBVEC                     ********   X    03
SCH$RWAIT                         ********   X    03
SETMODE_ACPBUF                    00000382 R      03
SIZ...                          = 00000001
SS$_ACCVIO                      = 0000000C
SS$_CANCEL                      = 00000830
SS$_DATAOVERUN                  = 00000838
SS$_DEVNOTMOUNT                 = 0000007C
SS$_ENDOFFILE                   = 00000870
SS$_EXBYTLM                     = 00002A14
SS$_FILNOTACC                   = 000000AC
SS$_ILLIOFUNC                   = 000000F4
SS$_INSFMEM                     = 00000124
SS$_NOOPER                      = 00002894
SS$_NORMAL                      = 00000001
STAT_V_BC                       = 00000000
TR$C_MAXHDR                     = 0000001C
TR$C_NI_ALLEND1                 = 040000AB
TR$C_NI_ALLEND2                 = 00000000
TR$C_NI_ALLROU1                 = 030000AB
TR$C_NI_ALLROU2                 = 00000000
TR$C_NI_PREFIX                  = 000400AA
TR$C_NI_PROT                    = 00000360
TR$C_PRI_ECL                    = 0000001F
TR$C_PRI_RTHRU                  = 0000001F
UCB$B_DIPL                      = 0000005E
UCB$B_FIPL                      = 0000000B
UCB$C_LENGTH                    = 00000090
UCB$C_ND_LENGTH                   0000009A
UCB$L_DDB                       = 00000028
UCB$L_DDT                       = 00000088
UCB$L_DEVCHAR                   = 00000038
UCB$L_IOQFL                     = 0000004C
UCB$L_IRP                       = 00000058
UCB$L_VCB                       = 00000034
UCB$M_ONLINE                    = 00000010
UCB$Q_DWB_LIST                    00000090
UCB$V_BSY                       = 00000008
UCB$W_DEVBUFSIZ                 = 00000042
UCB$W_NEXT_ID                     00000098
UCB$W_STS                       = 00000064
UNIT_INIT                         000008A7 R      03
VA$M_BYTE                       = 000001FF
VA$S_VPN                        = 00000015
VA$V_VPN                        = 00000009
VCB$L_AQB                       = 00000010
VCB$W_TRANS                     = 0000000C
VEC$L_ADP                       = 00000014
VEC$L_UNITINIT                  = 00000018
_$$_                            = 00000000
```

### Psect synopsis

| PSECT name | Allocation | PSECT No. | Attributes |
|---|---|---|---|
| . ABS . | 00000000 ( 0.) | 00 ( 0.) | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| $ABS$ | 0000009A ( 154.) | 01 ( 1.) | NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE |
| $$$105_PROLOGUE | 00000078 ( 120.) | 02 ( 2.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| $$$115_DRIVER | 000008E8 ( 2280.) | 03 ( 3.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG |

### Performance indicators

| Phase | Page faults | CPU Time | Elapsed Time |
|---|---|---|---|
| Initialization | 35 | 00:00:00.09 | 00:00:00.55 |
| Command processing | 161 | 00:00:01.02 | 00:00:03.55 |
| Pass 1 | 750 | 00:00:31.06 | 00:01:01.07 |
| Symbol table sort | 0 | 00:00:04.02 | 00:00:08.25 |
| Pass 2 | 589 | 00:00:07.02 | 00:00:16.97 |
| Symbol table output | 35 | 00:00:00.31 | 00:00:00.73 |
| Psect synopsis output | 4 | 00:00:00.02 | 00:00:00.23 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 1577 | 00:00:43.54 | 00:01:31.35 |

The working set limit was 1350 pages.
170238 bytes (333 pages) of virtual memory were used to buffer the intermediate code.
There were 140 pages of symbol table space allocated to hold 2604 non-local and 80 local symbols.
1872 source lines were read in Pass 1, producing 22 object records in Pass 2.
66 pages of virtual memory were used to define 60 macros.

### Macro library statistics

| Macro library name | Macros defined |
|---|---|
| _$255$DUA28:[SHRLIB]NMALIBRY.MLB;1 | 0 |
| _$255$DUA28:[SHRLIB]EVCDEF.MLB;1 | 0 |
| _$255$DUA28:[NETACP.OBJ]NETDRV.MLB;1 | 0 |
| _$255$DUA28:[NETACP.OBJ]NET.MLB;1 | 6 |
| _$255$DUA28:[SYS.OBJ]LIB.MLB;1 | 32 |
| _$255$DUA28:[SYSLIB]STARLET.MLB;2 | 12 |
| TOTALS (all libraries) | 50 |

3019 GETS were required to define 50 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:NDDRIVER/OBJ=OBJ$:NDDRIVER MSRC$:NDDRIVER/UPDATE=(ENH$:NDDRIVER)+EXECML$/LIB+LIB$:NET/LIB+LIB$:NETDRV/LIB+SHRLIB$:EVC
```

NETUSR
SDL

LIBTAIL
B32

XWBDEF
SDL

NETDEFS
MAR

PSIUSR
SDL

NETDRVMAC
MAR

NODRIVER
LIS

NSPMSGDEF
SDL

LIBHEAD
B32

NET
LIS

NETMACROS
MAR

NETACPTRN
LIS