```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
```

```
NN       NN    000000    DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
NN       NN    000000    DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
NN       NN  00      00  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
NN       NN  00      00  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
NNNN     NN  00      00  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
NNNN     NN  00      00  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
NN  NN   NN  00      00  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
NN  NN   NN  00      00  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
NN    NNNN  00      00  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
NN    NNNN  00      00  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
NN      NN  00      00  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
NN      NN  00      00  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
NN      NN    000000    DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....
NN      NN    000000    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  NODRIVER- VAX/VMS DMF32 Async DDMCP Line Driver
0000     2          .IDENT  'V04-001'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984                                     *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;++
0000    28 ; FACILITY:
0000    29 ;
0000    30 ;       VAX/VMS Async DDCMP line driver
0000    31 ;
0000    32 ; ABSTRACT:
0000    33 ;
0000    34 ;       This module contains the Async DDCMP line driver FDT routines,
0000    35 ;       interrupt dispatcher, interrupt service and fork routines.
0000    36 ;
0000    37 ; AUTHOR:
0000    38 ;
0000    39 ;       Meg Dumont 1-OCT-83
0000    40 ;
0000    41 ; MODIFIED BY:
0000    42 ;
0000    43 ;       V04-001 MMD0328         Meg Dumont,     5-Sep-1984  18:16
0000    44 ;               Fix bugs in the xmt error, error paths.
0000    45 ;
0000    46 ;       V03-008 MMD0322         Meg Dumont,     13-Aug-1984  8:55
0000    47 ;               Fix to SENSEMODE_CTRL where DDCMP could be called without
0000    48 ;               the protocol/NOB buffers having been set up.
0000    49 ;
0000    50 ;       V03-007 MMD0318         Meg Dumont,     25-Jul-1984  11:34
0000    51 ;               Fix to clear the GFB before giving it to protocol and fix
0000    52 ;               to DEVTIMER, we were using the wrong register in testing the
0000    53 ;               timeout conditions.
0000    54 ;
0000    55 ;       V03-006 LMP0275         L. Mark Pilant,         12-Jul-1984  12:30
0000    56 ;               Initialize the ACL info in the ORB to be a null descriptor
0000    57 ;               list rather than an empty queue.  This avoids the overhead
```

```
0000     58 ;                  of locking and unlocking the ACL mutex, only to find out
0000     59 ;                  that the ACL was empty.
0000     60 ;
0000     61 ;          V03-005 MMD0299         Meg Dumont,     18-May-1984  10:26
0000     62 ;                  Fix to returning of quota on line startup. Also fix to check
0000     63 ;                  that the buffers are allocated before using them in the
0000     64 ;                  sensemode paths'.
0000     65 ;
0000     66 ;          V03-004 MMD0297         Meg Dumont,     14-May-1984  14:54
0000     67 ;                  Fix bug in COM_XMITFDT error path
0000     68 ;
0000     69 ;          V03-003 MMD0294         Meg Dumont,     23-Apr-1984  16:02
0000     70 ;                  Fix so that if the TFB address is not valid the driver won't
0000     71 ;                  try to do an XMIT. Fix to support a new interface with the
0000     72 ;                  protocol. This interface is designed to fix problems with
0000     73 ;                  retransmitting messages out of order.
0000     74 ;
0000     75 ;          V03-002 LMP0221         L. Mark Pilant,         27-Mar-1984  10:13
0000     76 ;                  Change UCB$L_OWNUIC to ORB$L_OWNER and UCB$W_VPROT to
0000     77 ;                  ORB$W_PROT.
0000     78 ;
0000     79 ;          V03-001 MMD0262         Meg Dumont,     22-Mar-1984  14:40
0000     80 ;                  Fix so that the driver doesn't crash if an XMT or RCV
0000     81 ;                  is queued to the device before the device is started.
0000     82 ;                  Add support for timers on receives and transmits.
0000     83 ;
0000     84 ;--
```

```
0000     86
0000     87 ; System definitions
0000     88
0000     89          $ACBDEF                                 ; AST control
0000     90          $CRBDEF                                 ; Controller request block
0000     91          $CXBDEF                                 ; Complex buffer block
0000     92          $DDBDEF                                 ; Device data block
0000     93          $DDCMPDEF                               ; Constant def's for DDCMP
0000     94          $DEVDEF                                 ; Device charateristics
0000     95          $DLADEF                                 ; Driver/protocol command defs
0000     96          $DLKDEF                                 ; Driver/protocol command defs
0000     97          $DYNDEF                                 ; Dynamic data structures
0000     98          $DPTDEF                                 ; Driver prologue table
0000     99          $GFDEF                                  ; Global field definitions
0000    100          $IODEF                                  ; I/O function codes
0000    101          $IPLDEF                                 ; Interrupt priority levels
0000    102          $IRPDEF                                 ; I/O packets
0000    103          $JIBDEF                                 ; Job information block
0000    104          $MFDDEF                                 ; Message field descriptor
0000    105          $NMADEF                                 ; Network management def's
0000    106          $ORBDEF                                 ; Object's Rights Block offsets
0000    107          $PCBDEF                                 ; Process control block
0000    108          $SSDEF                                  ; System service status
0000    109          $TFDEF                                  ; DDCMP buffer def's
0000    110          $TTDEF                                  ; Terminal characteristics
0000    111          $TT2DEF                                 ;    ''         ''
0000    112          $TTYDEF                                 ;    ''         ''
0000    113          $TTYMACS                                ; Def terminal macros
0000    114          $TTYDEFS                                ; Define terminal definitions
0000    115          $TTYMODEM                               ; Define terminal modem signals
0000    116          $TTYUCBDEF                              ; Terminal UCB extension
0000    117          $TTYVECDEF                              ; Define terminal vectors
0000    118          $TQEDEF                                 ; Timer queue element def's
0000    119          $UCBDEF                                 ; Unit control block
0000    120          $VECDEF
0000    121          $XMDEF                                  ; XMDRIVER symbols
0000    122          $XMTQDEF                                ; Transmit Q def's
0000    123
0000    124
0000    125 ; Local macros
0000    126
0000    127          .MACRO  SETBIT  POS,BAS,?L              ; Set a single bit
0000    128                  BBSS    POS,BAS,L
0000    129          L:
0000    130          .ENDM   SETBIT
0000    131
0000    132          .MACRO  CLRBIT  POS,BAS,?L              ; Clear a single bit
0000    133                  BBCC    POS,BAS,L
0000    134          L:
0000    135          .ENDM   CLRBIT
0000    136
0000    137 ; Bit Definitions
0000    138
0000    139          .MACRO  BITDEF  BLK,SYM,BITVAL
0000    140
0000    141          'BLK'_V_'SYM' = BITVAL
0000    142          'BLK'_M_'SYM' = 1@<BITVAL>
```

```
0000    143
0000    144          .ENDM
0000    145
0000    146 ; Constant definitions
0000    147
0000    148          .MACRO  CONSTDEF BLK,SYM,CONSTVAL
0000    149
0000    150          'BLK'$C_'SYM' = CONSTVAL
0000    151          .ENDM
0000    152
0000    153 ; Set up the duetim table
0000    154
0000    155          .MACRO  STORE_DUETIM BAUDRATE,BITS,RESULT1,RESULT2,TABLE
0000    156
0000    157                  DIVW3   #'BAUDRATE',BITS,RESULT1
0000    158                  ADDW2   #1,RESULT1
0000    159                  MOVZBL  #TT$C_BAUD_'BAUDRATE',RESULT2
0000    160                  MOVW    RESULT1,TABLE[RESULT2]
0000    161
0000    162          .ENDM   STORE_DUETIM
```

```
0000   164          .MACRO  PARAM   TYPE,OFFSET,WIDTH,MIN,MAX,INVALID,BASE
0000   165 ;++
0000   166 ;        INPUTS:
0000   167 ;
0000   168 ;                TYPE = Parameter type
0000   169 ;                OFFSET = Offset in the data structure to current value
0000   170 ;                WIDTH = Width of field in the data structure (B,W,L)
0000   171 ;                MIN = Minimum value parameter is allowed to take
0000   172 ;                MAX = Maximum value parameter is allowed to take
0000   173 ;                INVALID = Invalid flags in status word
0000   174 ;                BASE = Data base (LINE,TRIB)
0000   175 ;--
0000   176          .IF BLANK TYPE
0000   177          .WORD   0
0000   178          .IF FALSE
0000   179          $$$TYP = TYPE & PRM_M_TYPE                ; Isolate type code
0000   180          .IIF NOT_BLANK  <MIN>, $$$TYP = $$$TYP!PRM_M_MIN
0000   181          .IIF NOT_BLANK  <MAX>, $$$TYP = $$$TYP!PRM_M_MAX
0000   182          .IIF NOT_BLANK  <INVALID>, $$$TYP = $$$TYP!PRM_M_INVALID
0000   183                  .WORD   $$$TYP
0000   184          $$$OFF = OFFSET & OFF_M_VALUE             ; Isolate offset only
0000   185          $$$WID = 0                                ; Set null width
0000   186          .IIF IDN <WIDTH><B>, $$$WID = <1@OFF_V_WIDTH>
0000   187          .IIF IDN <WIDTH><W>, $$$WID = <2@OFF_V_WIDTH>
0000   188          .IIF IDN <WIDTH><L>, $$$WID = <3@OFF_V_WIDTH>
0000   189                  .WORD   $$$OFF!$$$WID
0000   190          .IIF NOT_BLANK  <MIN>, .WORD    MIN
0000   191          .IIF NOT_BLANK  <MAX>, .WORD    MAX
0000   192          .IIF NOT_BLANK  <INVALID>, .WORD         INVALID
0000   193          'BASE'_PRM_BUFSIZ = 'BASE'_PRM_BUFSIZ + 6
0000   194          .ENDC
0000   195          .ENDM   PARAM
0000   196
0000   197          .MACRO  SKIP    BIT,LOC,REG,CONTEXT=W,?L        ; SKIP FIELD
0000   198          BBC     #BIT,LOC,L                ; Br if field not present
0000   199          TST'CONTEXT     (REG)+            ; Skip next field
0000   200 L:
0000   201          .ENDM   SKIP
```

```
                                D 13
0000    203
0000    204 ;++
0000    205 ; This macro translates into the CASEx instruction.  It calculates the
0000    206 ; "base" and "limit" parameters from the <index,displacement> list
0000    207 ; specfied in the 'vector' parameter.  The dispatch table is set up
0000    208 ; such that any unspecified index value within the bounds of the
0000    209 ; transfer vector is associated with a diplacement which transfers
0000    210 ; control to the first location after the CASE statement, i.e., behaves
0000    211 ; as if the index were out of bounds.
0000    212 ;
0000    213 ; Example:
0000    214 ;       $DISPATCH       R0,<-                           ; Message type in R0
0000    215 ;
0000    216 ;               ;index  displacement
0000    217 ;
0000    218 ;               <CI,    NSP$RCV_CI>,-                   ; Process CI message
0000    219 ;               <CC,    NSP$RCV_CC>,-                   ; Process CC message
0000    220 ;               <DI,    NSP$RCV_DI>,-                   ; Process DI message
0000    221 ;               <DC,    NSP$RCV_DI>,-                   ; Process DC message
0000    222 ;       >
0000    223 ;       BRW     NSP$RCV_ILLMSG                          ; Message type unknown
0000    224 ;--
0000    225 .MACRO  $DISPATCH,      INDX,VECTOR,TYPE=W,NMODE=S^#,?MN,?MX,?S,?SS,?ZZ
0000    226 SS:
0000    227
0000    228         .MACRO  $DSP1,$DSP1_1
0000    229                 .IRP    $DSP1_2,$DSP1_1
0000    230                         $DSP2   $DSP1_2
0000    231                 .ENDR
0000    232         .ENDM
0000    233
0000    234         .MACRO  $DSP2,$DSP2_1,$DSP2_2
0000    235                 .=<$DSP2_1-MN>*2 + S
0000    236                 .WORD   $DSP2_2-S
0000    237         .ENDM
0000    238
0000    239
0000    240         .MACRO  $BND1,$BND1_1,$BND1_2,$BND1_3
0000    241                 $BND2   $BND1_1,$BND1_2
0000    242         .ENDM
0000    243
0000    244         .MACRO  $BND2,$BND2_1,$BND2_2
0000    245                 .IIF    $BND2_1,$BND2_2-.,      .=$BND2_2
0000    246         .ENDM
0000    247
0000    248         .MACRO  $BND    $BND_1,$BND_2
0000    249                 .IRP    $BND_3,<$BND_2>
0000    250                         $BNDT   $BND_1,$BND_3
0000    251                 .ENDR
0000    252         .ENDM
0000    253
0000    254         .=0
0000    255 ZZ:
0000    256         $BND    GT,<VECTOR>
0000    257 MX:
0000    258         $BND    LT,<VECTOR>
0000    259 MN:
```

```
0000    260          .=SS
0000    261
0000    262 CASE'TYPE        INDX,#<MN-ZZ>,NMODE'<MX-MN>
0000    263 S:
0000    264          .REPT    MX-MN+1
0000    265          .WORD    <MX-MN>*2 + 2
0000    266          .ENDR
0000    267
0000    268          .=S
0000    269
0000    270          $DSP1    <<VECTOR>>
0000    271
0000    272          .=<MX-MN>*2 + S + 2
0000    273
0000    274 .ENDM
0000    275
0000    276
```

```
                        .SBTTL  Standard Driver Tables

                        .PSECT  $$$105_PROLOGUE

                ; Driver Prologue Table

                NO$DPT:

                        DPTAB   -
                                END=DRIVER_END,-                ; End of driver plus DDCMP
                                FLAGS=DPT$M_NOUNLOAD,-          ; Set driver can't be reloaded
                                ADAPTER=NULL,-                  ; no hardware assoc with driver
                                UCBSIZE=UCB$C_TL_LENGTH,-       ; UCB size
                                NAME=NODRIVER,-                 ; Driver name
                                VECTOR=ADDCMP_VECTOR            ; Asynch DDCMP vector table

                        DPT_STORE INIT                          ; Initialization data
                        DPT_STORE UCB,UCB$B_FIPL,B,8            ; Fork IPL
                        DPT_STORE UCB,UCB$B_DIPL,B,21           ; Device IPL
                        DPT_STORE UCB,UCB$L_DEVCHAR,L,<-        ; Device characteristics
                                DEV$M_NET!DEV$M_IDV!DEV$M_ODV>
                        DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_SCOM ; Device class
                        DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$_ADDCMP ; Device type
                        DPT_STORE UCB,UCB$W_DEVSTS,W,0          ; Internal status
                        DPT_STORE DDB,DDB$L_DDT,D,NO$DDT        ; Driver dispatch table
                        DPT_STORE UCB,UCB$L_NO_AST,L,0          ; Attention AST listhead
                        DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,256     ; Default buffer size
                        DPT_STORE UCB,UCB$L_DEVDEPEND,L,0       ; Clear devdepend field
                        DPT_STORE UCB,UCB$L_DEVDEPND2,L,0       ; Clear devdepend field
                        DPT_STORE ORB,ORB$B_FLAGS,B,-           ; Protection block flags
                                        <ORB$M_PROT_16>         ; SOGW protection word
                        DPT_STORE ORB,ORB$W_PROT,W,0            ; default protection
                        DPT_STORE ORB,ORB$L_OWNER,L,<^X010001>  ; [1,1] owns the device
                        DPT_STORE UCB,UCB$W_STS,0

                        DPT_STORE REINIT                        ; Init data also for reload
                        DPT_STORE CRB,CRB$L_INTD+VEC$L_INITIAL,D,NO$CONTROL_INIT ; Controller init
                        DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,D,NO$UNIT_INIT ; Unit init routine
                        DPT_STORE END

0000    317
```

```
                     0000   319
                     0000   320 ; $QIO parameter offsets
                     0000   321
00000000             0000   322 P1      = 0                                     ; Parameter 1
00000004             0000   323 P2      = 4                                     ; Parameter 2
00000008             0000   324 P3      = 8                                     ; Parameter 3
0000000C             0000   325 P4      = 12                                    ; Parameter 4
00000010             0000   326 P5      = 16                                    ; Parameter 5
                     0000   327
                     0000   328 ; & temp until CLASS_LENGTH is define in TTYDEF.SDL
                     0000   329
00000028             0000   330 CLASS_LENGTH = CLASSS_CLASS_DEF
                     0000   331
                     0000   332 ; & temp until defined in system
                     0000   333
000000C8             0000   334 DT$_ADDCMP = 200
                     0000   335
00000004             0000   336 DSC$A_POINTER   = 4                             ; Descriptor buffer address
00000004             0000   337 MAX_RCVS = 4                                    ; Max number of RCVS before
FFF3F300             0000   338 NO$C_LINE_PAR = ^X<FFF3F300>                     ; Bits to clear in DEVDEPEND
                     0000   339                                                 ; on start of line
FFFFF700             0000   340 NO$C_CIR_PAR = ^X<FFFFF700>                     ; Bits to clear in DEVDEPEND
                     0000   341                                                 ; on start of circuit
0000FFFF             0000   342 NO$C_PAD = ^X<FFFF>                            ; Pads to send at end of message
                     0000   343
                     0000   344 ; Bits set in this constant indicate the parity and the chaarcter length
                     0000   345 ; that we want the port driver to run at.
                     0000   346
00000018             0000   347 NO$C_SET_LINE = ^X<18>
                     0000   348
                     0000   349 ; NODRIVER constants
                     0000   350
                     0000   351 ; Message fields with constant values
                     0000   352
                     0000   353 CONSTDEF        NO,ENQ,5                        ; ENQ message identifier
                     0000   354 CONSTDEF        NO,SOH,129                      ; Data message identifier
                     0000   355 CONSTDEF        NO,DLE,144                      ; Maintenace message identifier
                     0000   356 CONSTDEF        NO,SYN,150                      ; SYN byte identifier
                     0000   357 CONSTDEF        NO,DEL,255                      ; DEL byte identifier
                     0000   358
                     0000   359 ; Receiver and transmitter states
                     0000   360
                     0000   361 CONSTDEF        NO,SEAR_MSG,0                   ; Receive state sear for start
                     0000   362                                                 ; of message
                     0000   363 CONSTDEF        NO,EMPTY,0                      ; Tranmit state nothing being send
                     0000   364 CONSTDEF        NO,HEADER,1                     ; Send header; recv header+crc
                     0000   365 CONSTDEF        NO,HCRC,2                       ; Send header crc
                     0000   366 CONSTDEF        NO,DATA,3                       ; Send data; recv data+crc
                     0000   367 CONSTDEF        NO,NOBUFFER,4                   ; Recv no buffer for data
                     0000   368 CONSTDEF        NO,DCRC,4                       ; Send data crc
                     0000   369 CONSTDEF        NO,PADS,5                       ; Send PADS on end of message
                     0000   370 CONSTDEF        NO,XMTERR,10                    ; Send enough data to clear xmt erro
                     0000   371
                     0000   372 ; Misc constants
                     0000   373
                     0000   374 CONSTDEF        NO,DEF_BUFSIZ,256               ; Default buffer size
                     0000   375 CONSTDEF        NO,HEADER_LEN,DDCMP$C_HEADER    ; Header length
```

```
                0000   376 CONSTDEF        NO,HEADER_HCRC,DDCMP$C_HEADER+2 ; Header length plus crc length
                0000   377 CONSTDEF        NO,DUETIM_TABLE_SIZE,256        ; Default number of baud rates avail
                0000   378
                0000   379 ; Bit definitions
                0000   380
                0000   381 BITDEF          NO_FS,PORTFORK,0                ; Fork on port
                0000   382 BITDEF          NO_FS,IOFORK,1                  ; Fork for IO completion
                0000   383 BITDEF          NO_FS,POWERFAIL,2               ; Fork for powerfail completion
                0000   384
                0000   385 ; Overlay of IRP
                0000   386
                0000   387
                0000   388         $DEFINI IRP
                0000   389
00000021        0000   390 . = IRP$W_FUNC+1                                ; Overlay function word
                0021   391
                0021   392 $DEF     IRP$B_NOFUNC    .BLKB   1               ; NODRIVER internal func code
                0022   393
00000040        0022   394 . = IRP$Q_STATION
                0040   395
                0040   396 $DEF     IRP$W_QUOTA     .BLKW   1               ; NODRIVER quota used for receives
                0042   397
                0042   398 ; Define driver internal function codes stored in IRP$B_NOFUNC of IRP.
                0042   399 ; NOTE:  These are not used as bit offsets, but as values.
                0042   400
                0042   401         _VIELD  NO_FC,0,<-                      ; Internal function codes
                0042   402                 <STRT_CIR>,-                    ; Start a tributary
                0042   403                 <STRT_LIN>,-                    ; Init the device inc UCB
                0042   404                 <STOP_CIR>,-                    ; Stop a tributary
                0042   405                 <STOP_LIN>,-                    ; Stop the device (inc stopping
                0042   406                 <READ_MODEM>,-                  ; Read the modem status
                0042   407                 >
                0042   408
                0042   409         $DEFEND IRP                             ; End of IRP overlays
                0000   410
                0000   411
                0000   412 ; NODRIVER UCB OVERLAYS
                0000   413
                0000   414
                0000   415         $DEFINI UCB
                0000   416
000000A8        0000   417 .=UCB$Q_TL_BRKTHRU
                00A8   418
                00A8   419 ; Because we are limited to the size of the terminal driver UCB for our UCB
                00A8   420 ; then when the device is started (the first QIO is issued to it) then we
                00A8   421 ; will allocate a block from nonpaged pool which we will use as our UCB
                00A8   422 ; extension. This block will be deallocate on the last DEASSIGN to the device.
                00A8   423
                00A8   424 $DEF     UCB$L_NO_BUFFER .BLKL 1                 ; Address of NOB buffer
                00AC   425 $DEF     UCB$L_NO_AST    .BLKL 1                 ; Attention AST list
                00B0   426
                00B0   427         $DEFEND UCB                             ; End of UCB overlays
                0000   428
                0000   429 ; These assumptions are made in case the terminal people decide to change
                0000   430 ; the use of the UCB$W_DEVSTS field. Chagnes in bit definitions of that
                0000   431 ; field could drastically affect the operation of this driver.
                0000   432
```

```
                     0000   433          ASSUME  UCB$V_TT_TIMO EQ 1
                     0000   434          ASSUME  UCB$V_TT_NOTIF EQ 2
                     0000   435          ASSUME  UCB$V_TT_HANGUP EQ 3
                     0000   436          ASSUME  UCB$V_TT_NOLOGINS EQ 15
                     0000   437          ASSUME  UCB$V_TT_DEVSTS_FILL EQ 4
                     0000   438          ASSUME  UCB$S_TT_DEVSTS_FILL EQ 11
                     0000   439
                     0000   440 ; Device status bits
                     0000   441
            00000000 0000   442 .=0
                     0000   443          _VIELD  NO_DS,4,<-                      ; UCB$W_DEVSTS bits
                     0000   444                  <XMTING,,M>,-                   ; Device XMTing if 0
                     0000   445                  <RCVING,,M>,-                   ; Device RCVing if 0
                     0000   446                  <INITED,,M>,-                   ; Unit initialized
                     0000   447                  <FORK_PEND,,M>,-                ; Fork pending
                     0000   448                  <ILOOP_SUP,,M>,-                ; Internal loop supported on device
                     0000   449                  <MSG_SENT,,M>,-                 ; Set when a msg was xmted
                     0000   450                  <SHUTDOWN,,M>,-                 ; Set when circuit going thru shutdo
                     0000   451                  <XMT_TIME,,M>,-                 ; Set if XMT_DUETIM is valid
                     0000   452                  <RCV_TIME,,M>,-                 ; Set if RCV_DUETIM is valid
                     0000   453                  >
                     0000   454
                     0000   455 ; Receive Flag field bit definitions
                     0000   456
                     0000   457 BITDEF           RCV,CNTL,0                      ; Set if rcv msg is an ENQ msg
                     0000   458
                     0000   459 ; Receive buffer definition
                     0000   460 ;
                     0000   461 ; This structure must be the same size as the CXB
                     0000   462
                     0000   463          $DEFINI RCV
            00000000 0000   464 .=0
                     0000   465 $DEF     RCV_L_LINK      .BLKL   2               ; Forward and backward links
                     0008   466 $DEF     RCV_W_BLKSIZE   .BLKW   1               ; Total block size
                     000A   467 $DEF     RCV_B_BLKTYPE   .BLKB   1               ; Block type
                     000B   468 $DEF     RCV_B_FIPL      .BLKB   1               ; Fork IPL
                     000C   469 $DEF     RCV_W_MSGSIZ    .BLKW   1               ; Size of this part of the msg to  r
                     000E   470 $DEF     RCV_W_INDEX     .BLKW   1               ; Index into rcved message
                     0010   471 $DEF     RCV_W_ERROR     .BLKW   1               ; Error status
                     0012   472 $DEF     RCV_W_FLAGS     .BLKW   1               ; Receive message flags
                     0014   473 $DEF     RCV_CXB_SPARE   .BLKB   CXB$K_HEADER-NO$C_HEADER_HCRC-.
                     0040   474                                                 ; Spare bytes to allow for CXB
                     0040   475 $DEF     RCV_Z_HEADER    .BLKB   NO$C_HEADER_HCRC ; Size in bytes of msg header
                     0048   476 $DEF     RCV_T_DATA                              ; Receive data
                     0048   477
                     0048   478          ASSUME  RCV_Z_HEADER+8 EQ CXB$K_HEADER
                     0048   479          ASSUME  RCV_T_DATA GE CXB$K_HEADER
                     0048   480
                     0048   481          $DEFEND RCV
                     0000   482
                     0000   483 ; P2 buffer header definition
                     0000   484
                     0000   485          $DEFINI P2B
            00000000 0000   486 .=0
                     0000   487 $DEF     P2B_L_POINTER   .BLKL   1       ; Pointer to start of data
                     0004   488 $DEF     P2B_L_BUFFER    .BLKL   1       ; Address of user's data buffer
                     0008   489 $DEF     P2B_W_SIZE      .BLKW   1       ; Size of P2 buffer
```

```
                     000A   490 $DEF     P2B_B_TYPE      .BLKB   1       ; Type of structure
                     000B   491 $DEF     P2B_B_SPARE     .BLKB   1       ; Spare byte
                     000C   492 $DEF     P2B_C_LENGTH                    ; Size of P2 buffer header
                     000C   493 $DEF     P2B_T_DATA                      ; Start of data
                     000C   494
                     000C   495          $DEFEND P2B
                     0000   496
                     0000   497
                     0000   498 ; NODRIVER buffer definitions
                     0000   499
                     0000   500          $DEFINI NOB
                 00000000 0000   501 .=0
                     0000   502 $DEF     NOB$L_FLINK     .BLKL           ; Nob buffer forward link
                     0004   503 $DEF     NOB$L_BLINK     .BLKL           ; Nob buffer backward link
                     0008   504 $DEF     NOB$W_SIZE      .BLKW           ; Size of Nob
                     000A   505 $DEF     NOB$B_TYPE      .BLKB           ; Type of buffer
                 0000000C 000B   506                     .BLKB           ; reserved
                     000C   507 $DEF     NOB$A_PRO_BUFFER .BLKL  1       ; Address of the DDCMP buffer
                     0010   508 $DEF     NOB$Q_ATTN      .BLKQ   1       ; Received messaeg list
                     0018   509 $DEF     NOB$Q_RCVS      .BLKQ   1       ; Received I/O queue
                     0020   510 $DEF     NOB$Q_FREE      .BLKQ   1       ; Free receive buffer queue
                     0028   511 $DEF     NOB$Q_POST      .BLKQ   1       ; Receive and transmit buffers to comp
                     0030   512 $DEF     NOB$L_RCV_INPR  .BLKL   1       ; Receive buffer awaiting data
                     0034   513 $DEF     NOB$L_XMT_INPR  .BLKL   1       ; Transmit buffer to be sent
                     0038   514 $DEF     NOB$L_RCV_DUETIM .BLKL  1       ; Receive message duetime
                     003C   515 $DEF     NOB$L_XMT_DUETIM .BLKL  1       ; Transmit message duetime
                     0040   516 $DEF     NOB$W_QUOTA     .BLKW   1       ; Byte quota for receive's
                     0042   517 $DEF     NOB$B_XSTATE    .BLKB   1       ; Transmitter state
                     0043   518 $DEF     NOB$B_RSTATE    .BLKB   1       ; Receiver state
                     0044   519 $DEF     NOB$W_PADS      .BLKW   1       ; Pads to send at end of messages
                     0046   520 $DEF     NOB$W_XMTERR_SIZE .BLKW 1       ; Number of pads to send on an error on an x
                     0048   521 $DEF     NOB$Z_HEADER    .BLKB   NO$C_HEADER_HCRC ; Area to receive header and crc
                     0050   522 $DEF     NOB$W_MSGSIZ    .BLKW   1       ; Current number of bytes left in rec hdr
                     0052   523 $DEF     NOB$W_INDEX     .BLKW   1       ; Index into receive header
                     0054   524 $DEF     NOB$W_ERROR     .BLKW   1       ; Error on receive if any
                     0056   525 $DEF     NOB$W_FLAGS     .BLKW   1       ; Flags for receive buffer after it is alloc
                     0058   526 $DEF     NOB$L_PID       .BLKL   1       ; Process ID
                     005C   527 $DEF     NOB$W_CHANL     .BLKW   1       ; Line channel number
                     005E   528 $DEF     NOB$W_CHANC     .BLKW   1       ; Circuit channel number
                     0060   529 $DEF     NOB$Z_DDCMP     .BLKL   4       ; Block for setable DDCMP parameters
                     0070   530 $DEF     NOB$C_SETPRM                    ; Start of UCB/LINE paramters
                     0070   531 $DEF     NOB$B_PRO       .BLKB   1       ; Protocol selection
                     0071   532 $DEF     NOB$B_DUP       .BLKB   1       ; Duplex setting
                     0072   533 $DEF     NOB$B_CON       .BLKB   1       ; Controller loopback
                     0073   534 $DEF     NOB$B_BFN       .BLKB   1       ; Number of receive buffers
                     0074   535 $DEF     NOB$W_DEVBUFSIZ .BLKW   1       ; Size of receive buffers
                     0076   536 $DEF     NOB$B_SPD       .BLKB   1       ; Set line speed
                 00000078 0077   537                     .BLKB   1       ; reserved
                     0078   538 $DEF     NOB$Z_DLA_ADDR  .BLKB   DLA$C_ADDR_LENGTH ; Buffer to pass to prot with
                     0080   539                                             ; address of queues prot needs
                     0080   540
                     0080   541 ; Block to copy the control message into on transmit
                     0080   542 $DEF     NOB$Z_CTL_MSG   .BLKB   XMTQ$K_LENGTH
                     00AA   543
                     00AA   544 $DEF     NOB$C_LENGTH                    ; Length of the NODRIVER buffer
                     00AA   545 $DEF     NOB$K_LENGTH                    ; Length of the NODRIVER buffer
                     00AA   546
```

```
                        $DEFEND NOB
00000000                .PSECT  $$$115_DRIVER,LONG
                        
                        
                        ; Driver Dispatch Table
                        
                        DDTAB   DEVNAM=NO,-                     ; Device name
                                START=NO$STARTIO,-              ; Start I/O routine
                                FUNCTB=NO$FUNCTABLE,-           ; Function decision table
                                CANCEL=NO$CANCEL,-              ; Cancel I/O routine
                                REGDMP=NO$REGDUMP,-             ; Register dump routine
                                DIAGBF=<36+12>,-                ; Diagnostic buffer size
                                ALTSTART=NO$ALT_ENTRY           ; Alternate entry routine
```

```
                        ; Function Decision Table
                        ;
                        NO$FUNCTABLE:
                                FUNCTAB,-                               ; Legal functions
                                        <WRITEVBLK,WRITELBLK,WRITEPBLK,-; Transmit functions
                                         READVBLK,READLBLK,READPBLK,-   ; Receive functions
                                         SETMODE,SENSEMODE,SETCHAR,-    ; Set mode functions
                                         >
                                FUNCTAB,-                               ; Buffered I/O functions
                                        <WRITEVBLK,WRITELBLK,WRITEPBLK,-; Transmit functions
                                         READLBLK,READPBLK,READVBLK,-
                                         SETMODE,SENSEMODE,SETCHAR,-    ; Set mode functions
                                         >

                                .IF     DF JNX$$$
                                functab fillbuffer,-                    ; Buffered I/O functions
                                        <writevblk,writelblk,writepblk,-; Transmit functions
                                         readvblk,readlblk,readpblk,-   ; Receive functions
                                         setmode,sensemode,setchar,-    ; Set mode functions
                                         >
                                .ENDC   ;DF JNX$$$

                                FUNCTAB NO$XMITFDT,-                    ; Transmit function dispatcher
                                        <WRITELBLK,WRITEPBLK,WRITEVBLK>
                                FUNCTAB NO$RCVFDT,-                     ; Receive function dispatcher
                                        <READLBLK,READPBLK,READVBLK>
                                FUNCTAB NO$SETMODEFDT,-                 ; FDT for set mode and set char
                                        <SETMODE,SETCHAR>               ;
                                FUNCTAB NO$SENSEMODEFDT,-               ; FDT sensemode routine
                                        <SENSEMODE>
```

```
                     0078   594
                     0078   595 ; The following is a table of services that the ASYNCH  DDCMP driver uses to
                     0078   596 ; interface with the terminal lines' port driver. It initially contains
                     0078   597 ; relative offsets to various routines and data structures needed by the
                     0078   598 ; terminal port driver. At driver load these relative offsets are reloacted
                     0078   599 ; to actual virtual addresses. The list is terminated by a 0 longword to
                     0078   600 ; signal te relocation routine where the list terminates.
                     0078   601 ;
                     0078   602 ; NOTE the because of how this table is indexed into the following order
                     0078   603 ; can not be changed, and if the placement of the routines are changed we
                     0078   604 ; must know about it.
                     0078   605
                     0078   606         ASSUME  CLASS_GETNXT     EQ 0
                     0078   607         ASSUME  CLASS_PUTNXT     EQ 4
                     0078   608         ASSUME  CLASS_SETUP_UCB  EQ 8
                     0078   609         ASSUME  CLASS_DS_TRAN    EQ 12
                     0078   610         ASSUME  CLASS_DDT        EQ 16
                     0078   611         ASSUME  CLASS_READERROR  EQ 20
                     0078   612         ASSUME  CLASS_DISCONNECT EQ 24
                     0078   613         ASSUME  CLASS_FORK       EQ 28
                     0078   614         ASSUME  CLASS_POWERFAIL  EQ 32
                     0078   615
                     0078   616 ADDCMP_VECTOR:
000010FA'            0078   617         .LONG   NO$GETNXT -              NO$DPT  ; Port driver calls to get next
                     007C   618                                                  ; character to output
00001274'            007C   619         .LONG   NO$PUTNXT -              NO$DPT  ; Port driver calls whenever a
                     0080   620                                                  ; character is received
000013E8'            0080   621         .LONG   NO$SETUP_UCB -           NO$DPT  ; Called to reset units' UCB,
                     0084   622                                                  ; called at powerfail and unit init
000013E9'            0084   623         .LONG   NO$PORT_TRANSITION -     NO$DPT  ; Handles modem transitions
                     0088   624                                                  ; on the line
00000000'            0088   625         .LONG   NO$DDT -                 NO$DPT  ; Class drivers DDT
000013EA'            008C   626         .LONG   NO$READERROR -           NO$DPT  ; Port driver detected an error
                     0090   627                                                  ; on the line
00001449'            0090   628         .LONG   NO$CLASS_DISCONNECT -    NO$DPT  ; Indicates that the terminal
                     0094   629                                                  ; is no longer connect to the system
000013A0'            0094   630         .LONG   NO$CLASS_PORTFORK -      NO$DPT  ; Used when the port driver
                     0098   631                                                  ; needs to issue a fork
0000144A'            0098   632         .LONG   NO$CLASS_POWERACTION -   NO$DPT  ; Called in unit init when a
                     009C   633                                                  ; powerfail occured
00000000             009C   634         .LONG   0
```

```
                     00A0    636          .SBTTL  P2 buffer verification tables
                     00A0    637
                     00A0    638
                     00A0    639 ; Define P2 buffer verification offsets
                     00A0    640
                     00A0    641          $DEFINI PARAM
                     0000    642
                     0000    643          _VIELD  PRM,0,<-                        ; Parameter bits and sizes
                     0000    644                  <TYPE,12,M>,-                   ; Parameter type
                     0000    645                  <MIN,1,M>,-                     ; Parameter minimum value
                     0000    646                  <MAX,1,M>,-                     ; Parameter maximum value
                     0000    647                  <INVALID,1,M>,-                 ; Parameter invalid flags
                     0000    648                  >
                     0000    649
                     0000    650          _VIELD  OFF,0,<-                        ; Offset word fields
                     0000    651                  <VALUE,14,M>,-                  ; Offset value
                     0000    652                  <WIDTH,2,M>,-                   ; Size of field in structure
                     0000    653                  >
                     0000    654
                     0000    655          $DEFEND PARAM
                     00A0    656
                     00A0    657
                     00A0    658 ; Define UCB (line) parameters
                     00A0    659
                     00A0    660 LINE_PRM_BUFSIZ = 0                             ; Line parameter buffer size
                     00A0    661 LINE_PARAM:                                     ; Start of line parameters
                     00A0    662
                     00A0    663          PARAM   NMA$C_PCLI_PRO,-                ; Protocol selection
                     00A0    664                  OFFSET=NOB$B_PRO,WIDTH=B,-
                     00A0    665                  MAX=NMA$C_LINPR_TRI,-
                     00A0    666                  INVALID=NO_DS_M_INITED,-        ; Device can't be ON
                     00A0    667                  BASE=LINE
                     00A8    668
                     00A8    669          PARAM   NMA$C_PCLI_DUP,-                ; Duplex mode
                     00A8    670                  OFFSET=NOB$B_DUP,WIDTH=B,-
                     00A8    671                  MAX=NMA$C_DPX_HAL,-
                     00A8    672                  INVALID=NO_DS_M_INITED,-        ; Device can't be ON
                     00A8    673                  BASE=LINE
                     00B0    674
                     00B0    675          PARAM   NMA$C_PCLI_CON,-                ; Controller mode
                     00B0    676                  OFFSET=NOB$B_CON,WIDTH=B,-
                     00B0    677                  MAX=NMA$C_LINCN_LOO,-
                     00B0    678                  INVALID=NO_DS_M_INITED,-        ; Device can't be On
                     00B0    679                  BASE=LINE
                     00B8    680
                     00B8    681          PARAM   NMA$C_PCLI_BFN,-                ; Number of receives
                     00B8    682                  OFFSET=NOB$B_BFN,WIDTH=B,-
                     00B8    683                  MIN=1,MAX=255,-
                     00B8    684                  INVALID=NO_DS_M_INITED,-        ; Device can't be ON
                     00B8    685                  BASE=LINE
                     00C2    686
                     00C2    687          PARAM   NMA$C_PCLI_BUS,-                ; Buffer size
                     00C2    688                  OFFSET=NOB$W_DEVBUFSIZ,WIDTH=W,-
                     00C2    689                  MIN=1,MAX=16383,-
                     00C2    690                  INVALID=NO_DS_M_INITED,-        ; Device can't be ON
                     00C2    691                  BASE=LINE
                     00CC    692
```

```
                     00CC   693          PARAM   NMA$C_PCLI_RTT,-                    ; Retransmit timer
                     00CC   694                   OFFSET=NOB$Z_DDCMP+DLK$W_REPWAIT,-
                     00CC   695                   WIDTH=W,-
                     00CC   696                   MIN=50,-
                     00CC   697                   BASE=LINE
                     00D2   698
                     00D2   699          PARAM                                       ; End of table
                     00D4   700
                     00D4   701 ; Define Trib parameters
                     00D4   702
                     00D4   703 TRIB_PRM_BUFSIZ  = 0                                 ; Trib parameter buffer size
                     00D4   704 TRIB_PARAM:                                          ; Start of trib parameters
                     00D4   705
                     00D4   706          PARAM   NMA$C_PCCI_TRI,-                    ; Trib address
                     00D4   707                   OFFSET=DLK$B_TRIB,WIDTH=B,-
                     00D4   708                   MIN=1,-
                     00D4   709                   INVALID=XM$M_STS_ACTIVE,-          ; Trib can't be established
                     00D4   710                   BASE=TRIB
                     00DC   711
                     00DC   712          PARAM   NMA$C_PCCI_MTR,-                    ; Max number of messages sent
                     00DC   713                   OFFSET=DLK$B_MSGCNT,WIDTH=B,-      ; in a selection interval
                     00DC   714                   MIN=1,-
                     00DC   715                   MAX=100,-
                     00DC   716                   BASE=TRIB
                     00E4   717
                     00E4   718          PARAM   NMA$C_PCCI_MST,-                    ; Maintenance state
                     00E4   719                   OFFSET=DLK$B_MAINT,WIDTH=B,-
                     00E4   720                   MAX=NMA$C_STATE_OFF,-
                     00E4   721                   BASE=TRIB
                     00EA   722
                     00EA   723          PARAM   NMA$C_PCCI_MRB,-                    ; Maxium receive buffers
                     00EA   724                   OFFSET=DLK$B_MRB,WIDTH=B,-
                     00EA   725                   MIN=1,-
                     00EA   726                   INVALID=XM$M_STS_ACTIVE,-
                     00EA   727                   BASE=TRIB
                     00F2   728
                     00F2   729          PARAM                                       ; End of trib tables
                     00F4   730
```

```
                     00F4   732
                     00F4   733
                     00F4   734 ; DEFAULT TRIBUTARY PARAMETERS for DDCMP
                     00F4   735
                     00F4   736 DEF_TRIB_PARAM::
                     00F4   737          ASSUME  DLK$B_MSGCNT EQ DLK$B_TRIB+1
                     00F4   738          ASSUME  DLK$B_MAXREP EQ DLK$B_MSGCNT+1
                     00F4   739          ASSUME  DLK$B_MAXSEL EQ DLK$B_MAXREP+1
                     00F4   740          ASSUME  DLK$W_REPWAIT EQ DLK$B_MAXSEL+1
                     00F4   741          ASSUME  DLK$W_SELWAIT EQ DLK$W_REPWAIT+2
                     00F4   742          ASSUME  DLK$B_MAINT EQ DLK$W_SELWAIT+2
                     00F4   743          ASSUME  DLK$B_MRB EQ DLK$B_MAINT+1
                  01 00F4   744          .BYTE   1                               ; Default Trib address
                  04 00F5   745          .BYTE   4                               ; Max nmb of msgs sent / select
                  04 00F6   746          .BYTE   4                               ; Max nmb of sel intls allowed
                  03 00F7   747          .BYTE   3                               ; Max times to reXMT a msg
                0BB8 00F8   748          .WORD   3000                            ; Reply timeout timer
                0BB8 00FA   749          .WORD   3000                            ; Selection timer in sec
                  01 00FC   750          .BYTE   NMA$C_STATE_OFF                 ; Default is no maint mode
                  FF 00FD   751          .BYTE   255                             ; Default to unlimited rcv buff
          0000000A 00FE   752          DEF_TRIB_PARAMSZ = .-DEF_TRIB_PARAM
                     00FE   753
                     00FE   754 ; Default line parameter values
                     00FE   755
                     00FE   756 DEF_LINE_PARAM::
                     00FE   757          ASSUME  NOB$B_PRO EQ NOB$C_SETPRM
                     00FE   758          ASSUME  NOB$B_DUP EQ NOB$B_PRO+1
                     00FE   759          ASSUME  NOB$B_CON EQ NOB$B_DUP+1
                     00FE   760          ASSUME  NOB$B_BFN EQ NOB$B_CON+1
                     00FE   761          ASSUME  NOB$W_DEVBUFSIZ EQ NOB$B_BFN+1
                     00FE   762
                  00 00FE   763          .BYTE   NMA$C_LINPR_POI                 ; Protocol is point-point
                  00 00FF   764          .BYTE   NMA$C_DPX_FUL                   ; Duplex is full
                  00 0100   765          .BYTE   NMA$C_LINCN_NOR                 ; Controller mode is normal
                  01 0101   766          .BYTE   1                               ; Number of receive buffers
                0100 0102   767          .WORD   NO$C_DEF_BUFSIZ                 ; Size of receive buffers
                     0104   768
          00000006 0104   769          DEF_LINE_PARAMSZ = .-DEF_LINE_PARAM
                     0104   770
                     0104   771 ; Table to store all possible duetimes for receive and tranmsit messages.
                     0104   772 ; The time to wait is store in this table and is calculated as follows:
                     0104   773 ; ((default buffer size + the ddcmp overhead bytes) * eight bits per byte )
                     0104   774 ; divided the baud rate. This table is calculated each time the line is
                     0104   775 ; started so we will have the most recent information about the default buffer
                     0104   776 ; size. The byte value of the terminal line speed is used to index into this
                     0104   777 ; table.
                     0104   778
                     0104   779 DUETIM_TABLE:
          00000304   0104   780          .BLKW   NO$C_DUETIM_TABLE_SIZE
                     0304   781
```

```
                     0304   783          .SBTTL  NO$CONTROL_INIT - Initialize Sync line device
                     0304   784 ;++
                     0304   785 ; NO$CONTROL_INIT - Initialize Sync line unit
                     0304   786 ;
                     0304   787 ; FUNCTINAL DESCRIPTION:
                     0304   788 ;
                     0304   789 ; This routine is called at driver load it sets up the global DPT for the
                     0304   790 ; driver adn relocates the drivers vector tables.
                     0304   791 ;
                     0304   792 ;       INPUTS:
                     0304   793 ;               R4 = Address of device CSR
                     0304   794 ;               R5 = Address of device IDB
                     0304   795 ;               R6 = Address of device DDB
                     0304   796 ;               R8 = Address of device CRB
                     0304   797 ;
                     0304   798 ;       OUTPUTS:
                     0304   799 ;               R4,R5,R8 are preserved
                     0304   800 ;--
                     0304   801 NO$CONTROL_INIT:
51   00000000'EF  DE  0304   802          MOVAL   NO$DPT,R1
00000000'GF   51  D0  030B   803          MOVL    R1,G^NO$GL_DPT
                     0312   804
      50   1E A1  3C  0312   805          MOVZWL  DPT$W_VECTOR(R1),R0
         50   51  C0  0316   806          ADDL2   R1,R0
                     0319   807
                     0319   808 ; Relocate the asynch ddcmp vector table
                     0319   809
      60  D5  0319   810 5$:     TSTL    (R0)
      05  15  031B   811         BLEQ    10$
80    51  C0  031D   812         ADDL2   R1,(R0)+
      F7  11  0320   813         BRB     5$
                     0322   814
          05  0322   815 10$:    RSB
                     0323   816
```

```
                     0323   818          .SBTTL  NO$UNIT_INIT - Initialize the device unit
                     0323   819 ;++
                     0323   820 ; NO$UNIT_INIT - Initialize the device unit
                     0323   821 ;
                     0323   822 ; FUNCTIONAL DESCRIPTION:
                     0323   823 ;
                     0323   824 ; This routine is called when the driver is loaded and during powerfailure
                     0323   825 ; recovery.  It sets the unit status to ONLINE.  Also, if called during
                     0323   826 ; powerfail recovery, it shuts down the device.
                     0323   827 ;
                     0323   828 ;       INPUTS:
                     0323   829 ;               R4 = Address of the device CSR
                     0323   830 ;               R5 = UCB address
                     0323   831 ;
                     0323   832 ;       OUTPUTS:
                     0323   833 ;               R5 preserved
                     0323   834 ;--
                     0323   835 NO$UNIT_INIT:                                    ; Initialize the unit
11 64 A5    05   E1 0323   836          BBC     #UCB$V_POWER,UCB$W_STS(R5),10$  ; Br if not powerfail recovery
            0B   E1 0328   837          BBC     #XM$V_STS_ACTIVE,-              ; Br if not previously active
      0C 44 A5        032A   838                  UCB$L_DEVDEPEND(R5),10$
            0F   BB 032D   839          PUSHR   #^M<R0,R1,R2,R3>                ; Save all registers
          109E   30 032F   840          BSBW    SCHED_FORK_POWERFAIL
            0F   BA 0332   841          POPR    #^M<R0,R1,R2,R3>
            05   0334   842          RSB
                     0335   843
                     0335   844 5$:     BUG_CHECK NOBUFPCKT,FATAL
                     0339   845
                     0339   846
                     0339   847 ; This driver makes the assumption that IPL$_SYNC, IPL$_TIMER and the drivers
                     0339   848 ; fork IPL are all equal. If any of these change the integrity of the driver
                     0339   849 ; can not be assured. THIS DRIVER WILL NOT WORK ON PRE VERSION 4 SYSTEMS.
                     0339   850
                     0339   851
                     0339   852          ASSUME IPL$_SYNCH EQ IPL$_TIMER
                     0339   853
                     0339   854
                     0339   855 ; We must also be sure that SYNCH and the drivers FIPL are the same. If this
                     0339   856 ; changes the integrity of the driver can not be assured with out some
                     0339   857 ; major changes.
                     0339   858
0B A5       08   91 0339   859 10$:     CMPB    #IPL$_SYNCH,UCB$B_FIPL(R5)      ; If the IPLs are not equal then
            F6   12 033D   860          BNEQ    5$                              ; cause a fatal bugcheck
64 A5       10   A8 033F   861          BISW    #UCB$M_ONLINE,UCB$W_STS(R5)     ; Set software status ONLINE
                 05 0343   862          RSB
                     0344   863
```

```
0344   865          .IF     DF JNX$$$
0344   866 fillbuffer:
0344   867          movzbl  ucb$b_last(r5),r0
0344   868          movl    r3,ucb$l_buffer(r5)[r0]
0344   869          movq    #0,irp$l_media(r3)
0344   870          incl    r0
0344   871          bicl    #^c<jnx_size-1>,r0
0344   872          movb    r0,ucb$b_last(r5)
0344   873          rsb
0344   874          .ENDC   ;DF JNX$$$
0344   875
```

```
                                        .SBTTL NO$XMITFDT - Transmit I/O FDT routine
                                ;++
                                ; NO$XMITFDT - Transmit I/O FDT routine
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ; This routine allocates a system buffer for the QIO. Then calls the
                                ; common FDT routine give the buffer to DDCMP.
                                ;
                                ;       INPUTS:
                                ;               R3 = I/O packet address
                                ;               R4 = Current PCB address
                                ;               R5 = UCB address
                                ;               R6 = CCB address
                                ;
                                ;       OUTPUTS:
                                ;               R3,R4,R5,R6,R7,R8,R9 are preserved
                                ;
                                ;--
                                NO$XMITFDT:                                             ; Transmit FDT routine
    03F8 8F    BB  0344   897           PUSHR   #^M<R3,R4,R5,R6,R7,R8,R9>
    57   6C    D0  0348   898           MOVL    P1(AP),R7                              ; Get address of buffer
59   04 AC    D0  034B   899           MOVL    P2(AP),R9                              ; Get the size of buffer
         55    13  034F   900           BEQL    10$                                    ; If EQL then size is zero
    50   57    D0  0351   901           MOVL    R7,R0                                  ; Set up R0 and R1 for
    51   59    D0  0354   902           MOVL    R9,R1                                  ; write access check
00000000'GF    16  0357   903           JSB     G^EXE$WRITECHK                         ; Check the acc of users buffer
51   59   2A    C1  035D   904           ADDL3   #XMTQ$K_LENGTH,R9,R1                   ; Get length of buffer to alloc
    56   53    D0  0361   905           MOVL    R3,R6                                  ; Save the IRP address
00000000'GF    16  0364   906           JSB     G^EXE$BUFFRQUOTA                       ; Check quota
       3C 50    E9  036A   907           BLBC    R0,20$                                 ; If LBC not enough quota
00000000'GF    16  036D   908           JSB     G^EXE$ALLOCBUF                         ; Allocated the buffer
       33 50    E9  0373   909           BLBC    R0,20$                                 ; If LBC not buffer allocated
    53   56    D0  0376   910           MOVL    R6,R3                                  ; Retreive the IRP
    58   52    D0  0379   911           MOVL    R2,R8                                  ; Get buffer address in R8
5C   0080 C4    D0  037C   912           MOVL    PCB$L_JIB(R4),R0                       ; Get the JIB address
  20 A0   51    C2  0381   913           SUBL2   R1,JIB$L_BYTCNT(R0)                    ; Adjust buffered I/O
  30 A3   51    B0  0385   914           MOVW    R1,IRP$W_BOFF(R3)                      ; Save byte offset
  2C A3   52    D0  0389   915           MOVL    R2,IRP$L_SVAPTE(R3)                    ; Save system PTE
       1F A8    94  038D   916           CLRB    XMTQ$B_FLAG(R8)                        ; Clr bits, not "Internal" IRP
        0076    30  0390   917 5$:       BSBW    COP_BUFF                               ; Branch to copy into my buff
                    0393   918           SETIPL  UCB$B_FIPL(R5)                         ; Sync to FIPL
         1A    10  0397   919           BSBB    COM_XMITFDT                            ; Branch to common processing
       0D 50    E9  0399   920           BLBC    R0,20$                                 ; If BC then error
    03F8 8F    BA  039C   921 8$:       POPR    #^M<R3,R4,R5,R6,R7,R8,R9>
00000000'GF    17  03A0   922           JMP     G^EXE$QIORETURN
                   03A6   923
    50   14    3C  03A6   924 10$:      MOVZWL  S^#SS$_BADPARAM,R0                     ; Set status for abort
    03F8 8F    BA  03A9   925 20$:      POPR    #^M<R3,R4,R5,R6,R7,R8,R9>
00000000'GF    17  03AD   926           JMP     G^EXE$ABORTIO
                   03B3   927
```

```
                        03B3    929          .SBTTL  COM_XMITFDT - Common transmit FDT routine
                        03B3    930 ;++
                        03B3    931 ;COM_XMITFDT -  Common transmit FDT routine
                        03B3    932 ;
                        03B3    933 ; The allocated buffer is given to DDCMP where a header is added and if
                        03B3    934 ; possible the buffer is added to the transmit queue. The buffer keeps various
                        03B3    935 ; information on the transmit as it progresses from the transmit to the
                        03B3    936 ; completion queues.
                        03B3    937 ;
                        03B3    938 ;       INPUTS:
                        03B3    939 ;               R3 = IRP address
                        03B3    940 ;               R5 = UCB address
                        03B3    941 ;               R7 = Address of the user/"Internal" IRP buffer
                        03B3    942 ;               R8 = Address of allocated buffer
                        03B3    943 ;               R9 = Size of user/"Internal" IRP buffer
                        03B3    944 ;
                        03B3    945 ;       IPL = Fork IPL
                        03B3    946 ;
                        03B3    947 ;       OUTPUTS:
                        03B3    948 ;               R0 = Status of operation
                        03B3    949 ;               R5,R8 are preserved
                        03B3    950 ;
                        03B3    951 ;--
                        03B3    952 COM_XMITFDT:
    54   00A8 C5   D0   03B3    953          MOVL    UCB$L_NO_BUFFER(R5),R4          ; Get NOB address
                45   13   03B8    954          BEQL    45$                             ; If eql device not started
                30   BB   03BA    955          PUSHR   #^M<R4,R5>                      ; Save these registers
    55     0C A4   D0   03BC    956          MOVL    NOB$A_PRO_BUFFER(R4),R5         ; Get addr of start of TFB
                3B   13   03C0    957          BEQL    43$
        56   02   9A   03C2    958          MOVZBL  #DLK$C_XMTMSG,R6                ; Set that this is a msg to XMT
                57   D4   03C5    959          CLRL    R7                              ; Clear error bits
             FC36'   30   03C7    960          BSBW    DDCMP                           ; Branch to set up the header
                30   BA   03CA    961          POPR    #^M<R4,R5>                      ; Restore the registers
        56   09   91   03CC    962          CMPB    #DLK$C_ACTNOTCOM,R6             ; If EQL then protocol has not
                2E   13   03CF    963          BEQL    45$                             ; been started
  1C 57   04   E0   03D1    964          BBS     #DLK$V_XMTERR,R7,35$            ; If BS then problem with XMT
  03 57   0A   E1   03D5    965          BBC     #DLK$V_XMTCMP,R7,20$            ; If BC no XMT's to complete
           1233   30   03D9    966          BSBW    FINISH_XMT_IO                   ; Complete all XMT's
  16 57   01   E0   03DC    967 20$:     BBS     #DLK$V_PRSTERR,R7,40$           ; Branch BS persisent error
57   0220 8F   B3   03E0    968          BITW    #<DLK$M_XMTACK!-                ; If EQL then XMT not put on
                        03E5    969                  DLK$M_QFULERR>,R7          ; either queue so abort it
                0A   13   03E5    970          BEQL    35$
                51   D4   03E7    971          CLRL    R1                              ; Set no status for start_transmit
             085B   30   03E9    972          BSBW    START_TRANSMIT                  ; Else start the xmt'r
        50   01   3C   03EC    973 30$:     MOVZWL  S^#SS$_NORMAL,R0                ; Set normal return
                13   11   03EF    974          BRB     50$
                        03F1    975
        50   14   3C   03F1    976 35$:     MOVZWL  S^#SS$_BADPARAM,R0              ; Set status
                0E   11   03F4    977          BRB     50$
                        03F6    978
                        03F6    979 40$:     SETBIT  #XM$V_ERR_FATAL,UCB$L_DEVDEPEND(R5) ; Set fatal error
                EF   11   03FB    980          BRB     30$                             ; Complete as normal timer
                        03FD    981                                                  ; will shutdown the device
                30   BA   03FD    982 43$:     POPR    #^M<R4,R5>                      ; Restore the registers
50   20D4 8F   3C   03FF    983 45$:     MOVZWL  #SS$_DEVINACT,R0                ; Set the device is not active
  51   44 A5   D0   0404    984 50$:     MOVL    UCB$L_DEVDEPEND(R5),R1          ; Set status for abort
                05   0408    985          RSB
```

```
                          0409   986
                          0409   987 ;++
                          0409   988 ;COP_BUFF
                          0409   989 ;
                          0409   990 ; This routine takes the user or "Internal" IRP buffer which contains the
                          0409   991 ; message to send and writes it into the allocated buffer.
                          0409   992 ;
                          0409   993 ;       INPUTS:
                          0409   994 ;               R1 = Size of allocated buffer
                          0409   995 ;               R3 = IRP address
                          0409   996 ;               R8 = Address of allocated buffer
                          0409   997 ;               R7 = Address of buffer from which to move data
                          0409   998 ;               R9 = Size of user/"Internal" IRP buffer
                          0409   999 ;
                          0409  1000 ;       IPL = FIPL from ALT_ENTRY and ASTDEL from XMITFDT
                          0409  1001 ;
                          0409  1002 ;       OUTPUTS:
                          0409  1003 ;               Buffer is copied into allocated buffer
                          0409  1004 ;               R5,R8,R9 are preserved
                          0409  1005 ;
                          0409  1006 ;--
                          0409  1007 COP_BUFF:
      08 A8   51   B0     0409  1008         MOVW    R1,XMTQ$W_BUFLEN(R8)            ; Save the size of the buffer
              13   90     040D  1009         MOVB    S^#DYN$C_BUFIO,-                ; Set that this is an XMT
           0A A8           040F  1010                 XMTQ$B_BUFTYP(R8)
      0C A8   53   D0     0411  1011         MOVL    R3,XMTQ$L_IRP(R8)               ; Save address of the IRP
1A A8   59   06   A1     0415  1012         ADDW3   #MFD$K_LENGTH,R9,-              ; Get the msg size plus header
                         041A  1013                 XMTQ$W_MSGSIZE(R8)              ; for character count
              55   DD     041A  1014         PUSHL   R5                              ; Save R5 before the MOV
2A A8   67   59   28     041C  1015         MOVC    R9,(R7),XMTQ$K_LENGTH(R8)       ; Move data into system buffer
              55 8ED0     0421  1016         POPL    R5                              ; Restore R5
                   05     0424  1017         RSB
```

```
                                0425  1019          .SBTTL  NO$ALT_ENTRY - Alternate I/O entry
                                0425  1020 ;++
                                0425  1021 ; NO$ALT_ENTRY - Alternate I/O entry point
                                0425  1022 ;
                                0425  1023 ; This routine is called by the other drivers to pass an "internal" I/O
                                0425  1024 ; request to the driver.  "Internal" IRP's are not built via $QIO.
                                0425  1025 ; The action here is to setup the IRP fields as if the packet had been
                                0425  1026 ; processed by the FDT routines.
                                0425  1027 ;
                                0425  1028 ; In this driver, the alternate entry point is called by the DECnet
                                0425  1029 ; Transport layer driver.
                                0425  1030 ;
                                0425  1031 ;       INPUTS:
                                0425  1032 ;               R3 = IRP address
                                0425  1033 ;               R5 = UCB address
                                0425  1034 ;               All pertinent fields of the IRP are assumed to be valid.
                                0425  1035 ;
                                0425  1036 ;       IPL = Fork IPL
                                0425  1037 ;
                                0425  1038 ;       OUTPUTS:
                                0425  1039 ;               R0 = Status of the request
                                0425  1040 ;               R3 and R5 preserved
                                0425  1041 ;--
                                0425  1042 NO$ALT_ENTRY:                                      ; Accept an "internal" IRP
                                0425  1043          .IF     DF JNX$$$
                                0425  1044          bsbw    fillbuffer
                                0425  1045          .ENDC   ;DF jnx$$$
                                0425  1046
38 A3    20D4 8F    3C          0425  1047          MOVZWL  #SS$_DEVINACT,I.[illegible]SL_MEDIA(R3)   ; Assume device inactive
                      0B    E0  042B  1048          BBS     #XM$V_STS_ACTIVE,-                      ; If BS status active
                03 44 A5          042D  1049                  UCB$L_DEVDEPEND(R5),5$                  ; device is on line
                    1198  31    0430  1050          BRW     IO_DONE                                 ; Else comp request in error
                      01  E0    0433  1051 5$:      BBS     #IRP$V_FUNC,-                           ; If BS then receive function
                50 2A A3          0435  1052                  IRP$W_STS(R3),ALT_RCVFDT
                 03F8 8F    BB    0438  1053          PUSHR   #^M<R3,R4,R5,R6,R7,R8,R9>               ; Save registers
             59    32 A3    3C    043C  1054          MOVZWL  IRP$W_BCNT(R3),R9                       ; Get the length of the buffer
                      2D  13    0440  1055          BEQL    10$                                     ; If EQL then bad parameter
          51    2A    59    C1  0442  1056          ADDL3   R9,#XMTQ$K_LENGTH,R1                    ; Add the XMTQ length
                56    53    D0  0446  1057          MOVL    R3,R6                                   ; Save the IRP
           00000000'GF    16    0449  1058          JSB     G^EXE$ALONONPAGED                       ; Allocated the buffer
                  21 50    E9  044F  1059          BLBC    R0,20$                                  ; If LBC not buffer allocated
                53    56    D0  0452  1060          MOVL    R6,R3                                   ; Retreive the IRP
                58    52    D0  0455  1061          MOVL    R2,R8                                   ; Get buffer address in R8
                  80 8F    90  0458  1062          MOVB    #XMTQ$M_INTERNAL,-                      ; Set that the I/O is from
                  1F A8         045B  1063                  XMTQ$B_FLAG(R8)                         ; an "Internal" IRP
                                045D  1064
                                045D  1065          .IF     DF JNX$$$
                                045D  1066          movl    r2,irp$L_fr4(r3)
                                045D  1067          .ENDC   ;DF jnx$$$
                                045D  1068
             57    2C B3    D0  045D  1069          MOVL    @IRP$L_SVAPTE(R3),R7                    ; Get interal IRP buffer addr
                   FFA5 30    0461  1070          BSBW    COP_BUFF                                ; Copy buffer into allcd buff
                   FF4C 30    0464  1071          BSBW    COM_XMITFDT                             ; Branch for common processng
                  10 50 E9    0467  1072          BLBC    R0,30$                                  ; If BC complete in error
                 03F8 8F BA   046A  1073          POPR    #^M<R3,R4,R5,R6,R7,R8,R9>               ; Restore registers
                         05   046E  1074          RSB
                                046F  1075
```

```
                38 A3    14    3C  046F  1076 10$:     MOVZWL  S^#SS$_BADPARAM,IRP$L_MEDIA(R3) ; Set abort status
                   03F8  8F    BA  0473  1077 20$:     POPR    #^M<R3,R4,R5,R6,R7,R8,R9>       ; Restore registers
                      1151     31  0477  1078          BRW     IO_DONE                         ; Complete the request
                                   047A  1079
                   51    58     D0  047A  1080 30$:     MOVL    R8,R1                           ; Save R8 it contains XMTQ buff
                   03F8  8F    BA  047D  1081          POPR    #^M<R3,R4,R5,R6,R7,R8,R9>       ; Restore registers
                38 A3    50     D0  0481  1082          MOVL    R0,IRP$L_MEDIA(R3)              ; Set status
                      1135     31  0485  1083          BRW     TRANSMIT_IO_DONE                ; Complete deall the buffer
                                   0488  1084
                                   0488  1085 ALT_RCVFDT:
                52    2C A3     D0  0488  1086          MOVL    IRP$L_SVAPTE(R3),R2             ; Is there a buffer to reuse
                         19     13  048C  1087          BEQL    10$                             ; If EQL then no
                      2C A3     D4  048E  1088          CLRL    IRP$L_SVAPTE(R3)                ; Clear so not deallocated
                         54     DD  0491  1089          PUSHL   R4
          50    000020D4  8F     D0  0493  1090          MOVL    #SS$_DEVINACT,R0                ; Assume failure
              54    00A8  C5     D0  049A  1091          MOVL    UCB$L_NO_BUFFER(R5),R4          ; Get NOB address
                         0D     13  049F  1092          BEQL    20$
                      0BD4     30  04A1  1093          BSBW    ADDFREELIST                     ; Else add it to the free list
                         54  8ED0  04A4  1094          POPL    R4
                      003B     30  04A7  1095 10$:     BSBW    COM_RCVFDT                      ; Do common processing
                   01 50       E9  04AA  1096          BLBC    R0,20$                          ; Br if unsuccessful
                               05  04AD  1097          RSB
                                   04AE  1098
                38 A3    50     D0  04AE  1099 20$:     MOVL    R0,IRP$L_MEDIA(R3)              ; Set status
                      1116     31  04B2  1100          BRW     IO_DONE                         ; Complete IRP in error
                                   04B5  1101
```

```
                     04B5  1103          .SBTTL  NO$RCVFDT - Receive I/O FDT routine
                     04B5  1104 ;++
                     04B5  1105 ; NO$RCVFDT - Receive I/O FDT routine
                     04B5  1106 ;
                     04B5  1107 ; FUNCTIONAL DESCRIPTION:
                     04B5  1108 ;
                     04B5  1109 ; The specified buffer is checked for accessibility. The buffer address and
                     04B5  1110 ; count are saved in the packet. Then IPL is set to device fork IPL and if
                     04B5  1111 ; a message is available the operation is completed.  Otherwise the packet
                     04B5  1112 ; is queued onto the waiting receive list.
                     04B5  1113 ;
                     04B5  1114 ; For requests specifing IO$M_NOW, the I/O is completed with status of
                     04B5  1115 ; SS$_ENDOFILE if no message is available when the test is made.
                     04B5  1116 ;
                     04B5  1117 ;
                     04B5  1118 ;       INPUTS:
                     04B5  1119 ;               R3 = I/O packet address
                     04B5  1120 ;               R4 = PCB address
                     04B5  1121 ;               R5 = UCB address
                     04B5  1122 ;               R6 = CCB address
                     04B5  1123 ;               R7 = Function code
                     04B5  1124 ;               AP = Address of first I/O request parameter
                     04B5  1125 ;
                     04B5  1126 ;       OUTPUTS:
                     04B5  1127 ;               R0 = Status of the receive request
                     04B5  1128 ;               R3-R7 preserved.
                     04B5  1129 ;--
                     04B5  1130 NO$RCVFDT:
          50   14   3C  04B5  1131          MOVZWL  #SS$_BADPARAM,R0                ; Assume illegal size
     51   04 AC   3C  04B8  1132          MOVZWL  P2(AP),R1                       ; Get size
          1F   13  04BC  1133          BEQL    ABORTIO                         ; Br if none specified
          50   6C  D0  04BE  1134          MOVL    P1(AP),R0                       ; Get buffer address
     38 A3   50   D0  04C1  1135          MOVL    R0,IRP$L_MEDIA(R3)              ; Save address
        30 A3   B4  04C5  1136          CLRW    IRP$W_BOFF(R3)                  ; No quota to return during
                     04C8  1137                                                  ; completion
00000000'GF   16  04C8  1138          JSB     G^EXE$READCHK                   ; Check buffer accessibility
                     04CE  1139                                                  ; (no return on no access)
                     04CE  1140          SETIPL  UCB$B_FIPL(R5)                  ; Synchronize access to the UCB
          11   10  04D2  1141          BSBB    COM_RCVFDT                      ; Process the request
        06 50   E9  04D4  1142          BLBC    R0,ABORTIO                      ; Br if error
00000000'GF   17  04D7  1143          JMP     G^EXE$QIORETURN                 ; Return to await completion
                     04DD  1144 ABORTIO:                                        ; Abort the I/O request
          51   D4  04DD  1145          CLRL    R1                              ; Don't return device status
00000000'GF   17  04DF  1146          JMP     G^EXE$ABORTIO                   ;
                     04E5  1147
                     04E5  1148
                     04E5  1149 ; Common receive processing
                     04E5  1150
                     04E5  1151 COM_RCVFDT:                                     ; Common receive processing
          0B   E0  04E5  1152          BBS     #XM$V_STS_ACTIVE,-              ; Br if device active
     06 44 A5        04E7  1153                  UCB$L_DEVDEPEND(R5),10$
50   20D4 8F   3C  04EA  1154          MOVZWL  #SS$_DEVINACT,R0                ; Set return status
          05   04EF  1155          RSB
                     04F0  1156
                     04F0  1157
                     04F0  1158 ; Check for an avaiable message and complete the receive
                     04F0  1159
```

```
                54   DD  04F0  1160 10$:     PUSHL   R4
          54    00A8 C5  D0  04F2  1161         MOVL    UCB$L_NO_BUFFER(R5),R4          ; Get NOB address
                     2D  13  04F7  1162         BEQL    30$                             ; If eql device not active
            52    10 B4   0F  04F9  1163         REMQUE  @NOB$Q_ATTN(R4),R2              ; Dequeue a received message
                     07  1D  04FD  1164         BVS     15$                             ; Br if none
                   1077  30  04FF  1165         BSBW    FINISH_RCV_IO                   ; Comp the I/O request
                      54 8ED0  0502  1166         POPL    R4
                         05  0505  1167         RSB
                             0506  1168
                             0506  1169 ; Queue the requst for future message arrival unless IO$M_NOW specified.
                             0506  1170 ; Receives are queued to the special receive wait queue.
                             0506  1171
          0B 20 A3   06   E0  0506  1172 15$:     BBS     #IO$V_NOW,IRP$W_FUNC(R3),20$    ; Br BS read NOW
             1C B4   63   0E  050B  1173         INSQUE  (R3),@NOB$Q_RCVS+4(R4)          ; Queue the I/O packet
                      54 8ED0  050F  1174         POPL    R4
                50    01   3C  0512  1175         MOVZWL  S^#SS$_NORMAL,R0                ; Set QIO status
                         05  0515  1176         RSB                                     ;
                             0516  1177
                      54 8ED0  0516  1178 20$:     POPL    R4
       38 A3    0870 8F   3C  0519  1179         MOVZWL  #SS$_ENDOFFILE,IRP$L_MEDIA(R3)  ; Set no message status
                   10A9  30  051F  1180         BSBW    IO_DONE                         ; Complete the I/O
                50    01   3C  0522  1181         MOVZWL  S^#SS$_NORMAL,R0                ; Set normal completetion
                         05  0525  1182         RSB                                     ; And return
                             0526  1183
                      54 8ED0  0526  1184 30$:     POPL    R4
             50    20D4 8F   3C  0529  1185         MOVZWL  #SS$_DEVINACT,R0                ; Set error
                         05  052E  1186         RSB                                     ; And return
                             052F  1187
```

```
                     052F  1189          .SBTTL  NO$SETMODEFDT,  Set mode I/O operation FDT routine
                     052F  1190
                     052F  1191  ;++
                     052F  1192  ; NO$SETMODEFDT - Set mode I/O operation FDT routine
                     052F  1193  ;
                     052F  1194  ; Functional description:
                     052F  1195  ;
                     052F  1196  ; This routine is used to set the configuration of the terminal device
                     052F  1197  ; including configuration of software DDCMP. The first SETMODE done to the
                     052F  1198  ; device causes a the NOB to be allocated. This buffer is used in place
                     052F  1199  ; of an extension to the devices UCB. Subfunction modifier bits are used
                     052F  1200  ; to specify the type of action to be taken. The two characteristics buffers
                     052F  1201  ; (P1 and P2) are used to describe specific characteristics.
                     052F  1202  ;
                     052F  1203  ; The QIO parameters for SETMODE are:
                     052F  1204  ;
                     052F  1205  ;       P1 = Optional address of quadword or longword buffer
                     052F  1206  ;       P2 = Optional address of buffer descriptor for extended characteristics
                     052F  1207  ;       P3 = Number of receive buffers to pre-allocate. Required on
                     052F  1208  ;            controller startup.
                     052F  1209  ;
                     052F  1210  ;
                     052F  1211  ; The subfunction modifiers are as follows:
                     052F  1212  ;
                     052F  1213  ;
                     052F  1214  ;        o STARTUP - start the device - this modifier is used to
                     052F  1215  ;              start the device.
                     052F  1216  ;
                     052F  1217  ;       o SHUTDOWN - shutdown the device - this modifier
                     052F  1218  ;              is used to stop the device.
                     052F  1219  ;
                     052F  1220  ;       o ATTNAST - request an attention AST - this modifier is used
                     052F  1221  ;              to set up an AST to be delivered when a change of
                     052F  1222  ;              status occurs on this device.
                     052F  1223  ;
                     052F  1224  ;       o CTRL - perform the request on the Controller not the
                     052F  1225  ;              tributary.
                     052F  1226  ;
                     052F  1227  ;       o SET_MODEM - set line unit's mode register.
                     052F  1228  ;
                     052F  1229  ;       INPUTS:
                     052F  1230  ;              R3 = IRP address
                     052F  1231  ;              R4 = PCB address
                     052F  1232  ;              R5 = UCB address
                     052F  1233  ;              R6 = CCB address
                     052F  1234  ;              R7 = Function code
                     052F  1235  ;              AP = address of first QIO parameter
                     052F  1236  ;
                     052F  1237  ;       OUTPUTS:
                     052F  1238  ;              R0 = status of setmode request
                     052F  1239  ;              R3-R5 are preserved.
                     052F  1240  ;              R7-R9 = destroyed
                     052F  1241  ;
                     052F  1242  ;--
                     052F  1243  NO$SETMODEFDT:
50   00A8 C5   D0    052F  1244          MOVL    UCB$L_NO_BUFFER(R5),R0          ; If NEQ then BUFFER allocated
          27   12    0534  1245          BNEQ    3$
```

```
                51   000000AA 8F    D0  0536  1246          MOVL    #NOB$K_LENGTH,R1                ; Else set up to allocate the buffer
                           0265    30  053D  1247          BSBW    ALLOC_BUFFER                    ; Call to allocate the buffer
                         57 50     E9  0540  1248          BLBC    R0,28$                          ; If LBC buffer could not be allocat
                            54     DD  0543  1249          PUSHL   R4
                        54   52     D0  0545  1250          MOVL    R2,R4                           ; Setup NOB address
                                       0548  1251          DSBINT  UCB$B_FIPL(R5)
                           027F    30  054F  1252          BSBW    INIT_NO_BUFFER                  ; Setup initial states in NOB
                    00A8 C5    54     D0  0552  1253          MOVL    R4,UCB$L_NO_BUFFER(R5)          ; Set address of NOB
                                       0557  1254          ENBINT
                            54 8ED0    055A  1255          POPL    R4
                     57   20 A3    3C  055D  1256  3$:     MOVZWL  IRP$W_FUNC(R3),R7               ; Get entire function code
                     03 57    09    E1  0561  1257          BBC     #IO$V_CTRL,R7,5$                ; Br if not controller request
                           00AD    31  0565  1258          BRW     SETMODE_CTRL                    ; Process controller request
                                       0568  1259
                                       0568  1260 ; Perform setmode request on a tributary
                                       0568  1261
                     34 57    08    E1  0568  1262  5$:     BBC     #IO$V_ATTNAST,R7,30$            ; Branch if not attention AST
                                       056C  1263
                                       056C  1264
                                       056C  1265 ; User is requesting an attention AST.
                                       056C  1266
                57   00AC C5    DE  056C  1267          MOVAL   UCB$L_NO_AST(R5),R7             ; Get addr of AST list
                 00000000'GF    16  0571  1268          JSB     G^COM$SETATTNAST                ; Set up attention AST
                50   00AB C5    D0  0577  1269          MOVL    UCB$L_NO_BUFFER(R5),R0          ; Get NOB address
                  51    10 A0    9E  057C  1270          MOVAB   NOB$Q_ATTN(R0),R1               ; Check for empty rcv list
                     61    51    D1  0580  1271          CMPL    R1,(R1)                         ; Empty?
                              08    13  0583  1272          BEQL    20$                             ; Yes, no need to inform user
                              53    DD  0585  1273 10$:     PUSHL   R3                              ; Else, save IRP address
                            1157    30  0587  1274          BSBW    POKE_USER                       ; Inform the user
                              53 8ED0    058A  1275          POPL    R3                              ; Restore IRP address
                  51    44 A5    D0  058D  1276 20$:     MOVL    UCB$L_DEVDEPEND(R5),R1          ; Get device characteristics
                     50    01    3C  0591  1277 23$:     MOVZWL  S^#SS$_NORMAL,R0                ; Set success
                00000000'GF    17  0594  1278 25$:     JMP     G^EXE$FINISHIO                  ; Complete the I/O
                                       059A  1279
                00000000'GF    17  059A  1280 28$:     JMP     G^EXE$ABORTIO                   ; Abort the request
                                       05A0  1281
                                       05A0  1282
                           04DD    30  05A0  1283 30$:     BSBW    GET_CHAR_BUFS                   ; Get P1 and P2 characteristics
                        30 50      E9  05A3  1284          BLBC    R0,40$                          ; Br if error - abort I/O
                32 57    07       E1  05A6  1285          BBC     #IO$V_SHUTDOWN,R7,50$           ; Branch if not trib shutdown
                                       05AA  1286
                                       05AA  1287 ; Shutdown tributary modifier specified.
                                       05AA  1288 ;
                                       05AA  1289 ; Validate P2 buffer. Then update trib parameter block.
                                       05AA  1290
                              02    90  05AA  1291          MOVB    S^#NO_FC_V_STOP_CIR,-           ; Set internal function code
                        21 A3          05AC  1292                  IRP$B_NOFUNC(R3)
                52   44 A5    3C  05AE  1293          MOVZWL  UCB$L_DEVDEPEND(R5),R2          ; Else, get status
                                       05B2  1294 35$:     DSBINT  UCB$B_FIPL(R5)                  ; Sync to get the UCB
                           144C    30  05B9  1295          BSBW    VALIDATE_P2_TRIB                ; Validate the P2 buffer
                                       05BC  1296          ENBINT                                  ; Restore IPL
                51   44 A5    D0  05BF  1297          MOVL    UCB$L_DEVDEPEND(R5),R1          ; Assume no error
                     CE 50    E9  05C3  1298          BLBC    R0,25$                          ; Br if error
                                       05C6  1299          DSBINT  UCB$B_FIPL(R5)                  ; Sync to get the UCB
                           0617    30  05CD  1300          BSBW    CHG_TRIB                        ; Change trib parameters
                                       05D0  1301          ENBINT                                  ; Restore IPL
                           049A    31  05D3  1302          BRW     QUEPKT                          ; Queue packet to driver
```

```
                        05D6  1303
00000000'GF   17        05D6  1304 40$:    JMP     G^EXE$ABORTIO                   ; Abort the I/O request
                        05DC  1305
 52   44 A5    3C        05DC  1306 50$:    MOVZWL  UCB$L_DEVDEPEND(R5),R2          ; Get status
                        05E0  1307         DSBINT  UCB$B_FIPL(R5)                  ; Sync to get the UCB
       141E     30        05E7  1308         BSBW    VALIDATE_P2_TRIB                ; Validate the P2 buffer
                        05EA  1309         ENBINT                                  ; Restore IPL
       E6 50    E9        05ED  1310         BLBC    R0,40$                          ; Br if error
                        05F0  1311 60$:    DSBINT  UCB$B_FIPL(R5)                  ; Sync to get the UCB
       05ED     30        05F7  1312         BSBW    CHG_TRIB                        ; Change trib parameters
                        05FA  1313         ENBINT                                  ; Restore IPL
 0D 57    06    E0        05FD  1314         BBS     #IO$V_STARTUP,R7,80$            ; Br if startup request
    50    01    3C        0601  1315         MOVZWL  S^#SS$_NORMAL,R0                ; Else, set successful return
 51   44 A5    D0        0604  1316         MOVL    UCB$L_DEVDEPEND(R5),R1          ; Set IOSB return status
00000000'GF   17        0608  1317         JMP     G^EXE$FINISHIO                  ; Finish the I/O request
          00    90        060E  1318 80$:    MOVB    S^#NO_FC_V_STRT_CIR,-           ; Set internal function code
       21 A3              0610  1319                 IRP$B_NOFUNC(R3)
       045B     31        0612  1320         BRW     QUEPKT
                        0615  1321
```

```
                                0615  1323          .SBTTL  SETMODE_CTRL,  Perform setmode FDT operation on controller
                                0615  1324
                                0615  1325 ;++
                                0615  1326 ; SETMODE_CTRL - Perform setmode FDT operation on controller
                                0615  1327 ;
                                0615  1328 ; Functional description:
                                0615  1329 ;
                                0615  1330 ; This routine performs the SETMODE FDT setup for the controller.
                                0615  1331 ;
                                0615  1332 ;       INPUTS:
                                0615  1333 ;               R3 = IRP address
                                0615  1334 ;               R4 = PCB address
                                0615  1335 ;               R5 = UCB address
                                0615  1336 ;               R7 = IRP function word
                                0615  1337 ;
                                0615  1338 ;       OUTPUTS:
                                0615  1339 ;               R0 = status of setmode request
                                0615  1340 ;               R3-R5 are preserved.
                                0615  1341 ;
                                0615  1342 ;--
                                0615  1343 SETMODE_CTRL:                                   ; Perform setmode on controller
                         0468   30  0615  1344          BSBW    GET_CHAR_BUFS           ; Get P1 and P2 characteristics
                        6E 50   E9  0618  1345          BLBC    R0,20$                  ; Br if error - Abort I/O
                                    061B  1346
                                B3  061B  1347          BITW    #<XM$M_CHR_CTRL!-       ; If NEQ then not multipoint
                                    061C  1348                  XM$M_CHR_DMC>,-         ; control or DMC mode
                 3C A3  0060 8F     061C  1349                  IRP$C_MEDIA+4(R3)
                             05  13  0621  1350          BEQL    5$
                          50 14  3C  0623  1351          MOVZWL  #SS$_BADPARAM,R0        ; Else set bad parameter
                             61  11  0626  1352          BRB     20$
                             56  D4  0628  1353 5$:      CLRL    R6                      ; Assume no protocol specified
                       51    0458 8F  3C  062A  1354          MOVZWL  #NMA$C_PCLI_PRO,R1      ; Check P2 buffer for multi
                                    062F  1355          DSBINT  UCB$B_FIPL(R5)          ; Sync to get the UCB
                                1564  30  0636  1356          BSBW    UNPACK_P2_BUF           ; point control specification
                                    0639  1357          ENBINT                          ; Restore IPL
                              0C 50   E9  063C  1358          BLBC    R0,14$                  ; If BC then no prot specified
                          52    01  91  063F  1359          CMPB    #NMA$C_LINPR_CON,R2     ; If EQL then multipoint contrl
                             02  13  0642  1360          BEQL    10$                     ; specified
                             05  11  0644  1361          BRB     14$
                          50 14  3C  0646  1362 10$:      MOVZWL  #SS$_BADPARAM,R0        ; Else set bad parameter
                             38  11  0649  1363          BRB     17$
                        52   68 A5  3C  064B  1364 14$:      MOVZWL  UCB$W_DEVSTS(R5),R2     ; Get device status
                                    064F  1365          DSBINT  UCB$B_FIPL(R5)          ; Sync to get the UCB
                                13C1  30  0656  1366          BSBW    VALIDATE_P2_UCB         ; Validate the P2 buffer
                                    0659  1367          ENBINT                          ; Restore IPL
                              2A 50   E9  065C  1368          BLBC    R0,20$                  ; Br if error
                        2F 57    07  E1  065F  1369          BBC     #IO$V_SHUTDOWN,R7,30$   ; Br if not shutdown request
                                    0663  1370
                                    0663  1371 ; Shutdown modifier specified
                                    0663  1372
                                 03 90  0663  1373          MOVB    S^#NO_FC_V_STOP_LIN,-   ; Set internal function code
                              21 A3     0665  1374                  IRP$B_NOFUNC(R3)
                                    0667  1375          DSBINT  UCB$B_FIPL(R5)          ; Sync to get NOB
                                 050B  30  066E  1376          BSBW    CHG_UCB_NOB             ; Update the UCB and the NOB
                                    0671  1377          ENBINT                          ; Lower IPL
                                 06 E1  0674  1378          BBC     #NO_DS_V_INITED,-       ; Br if controller not up
                           03 68 A5     0676  1379                  UCB$W_DEVSTS(R5),15$
```

```
                03F4   31  0679  1380          BRW     QUEPKT                          ; Queue packet to driver
                       0067C 1381
      51   44 A5   D0  067C  1382 15$:        MOVL    UCB$L_DEVDEPEND(R5),R1          ; Get IOSB1 return
           50   01   3C  0680  1383           MOVZWL  S^#SS$_NORMAL,R0                ; Set success
00000000'GF     17   0683  1384 17$:        JMP     G^EXE$FINISHIO                  ; Complete the I/O request
                       0689  1385
00000000'GF     17   0689  1386 20$:        JMP     G^EXE$ABORTIO                   ; Abort the I/O request
                       068F  1387
                00C2   31  068F  1388 25$:        BRW     50$                             ; Branch to compl request
                       0692  1389
   F9 57   06   E1  0692  1390 30$:        BBC     #IO$V_STARTUP,R7,25$            ; Br if not startup request
                01   90  0696  1391           MOVB    S^#NO_FC_V_STRT_LIN,-           ; Set internal function code
                 21 A3     0698  1392                   IRP$B_NOFUNC(R3)
   51   08 AC   3C  069A  1393           MOVZWL  P3(AP),R1                       ; If EQL then P3 not set up
                28   13  069E  1394           BEQL    35$
50   00A8 C5   D0  06A0  1395           MOVL    UCB$L_NO_BUFFER(R5),R0          ; Get NOB address
                06   E1  06A5  1396           BBC     #NO_DS_V_INITED,-               ; If BC then device not inited
     10 68 A5     06A7  1397                   UCB$W_DEVSTS(R5),33$            ; set number of buffers
   73 A0   51   91  06AA  1398           CMPB    R1,NOB$B_BFN(R0)                ; Are the buffer nmb's the same
                18   13  06AE  1399           BEQL    35$                             ; Br if yes
51   0451 8F   3C  06B0  1400           MOVZWL  #NMA$C_PCLI_BFN,R1              ; Set IOSB1 return
     50   14   3C  06B5  1401           MOVZWL  #SS$_BADPARAM,R0                ; Set error return
                C9   11  06B8  1402           BRB     17$                             ; Finish the I/O request
                       06BA  1403 33$:        DSBINT  UCB$B_FIPL(R5)                  ; Sync to get NOB
   73 A0   51   90  06C1  1404           MOVB    R1,NOB$B_BFN(R0)                ; Store new rcve buffer number
                       06C5  1405           ENBINT                                  ; Restore IPL
                       06C8  1406
                       06C8  1407 ; Set new P1, P2 parameters
                       06C8  1408
                06   E1  06C8  1409 35$:        BBC     #NO_DS_V_INITED,-               ; Br if device not already
      03 68 A5         06CA  1410                   UCB$W_DEVSTS(R5),36$            ; inited
          00AA   31  06CD  1411           BRW     70$
                       06D0  1412 36$:        DSBINT  UCB$B_FIPL(R5)                  ; Sync to get UCB
      03 38 A3   E9  06D7  1413           BLBC    IRP$L_MEDIA(R3),37$             ; Br if no P1
         44 A5   94  06DB  1414           CLRB    UCB$L_DEVDEPEND(R5)             ; Clear old characteristics
          049B   30  06DE  1415 37$:        BSBW    CHG_UCB_NOB                     ; Update the UCB and the NOB
                       06E1  1416           ENBINT                                  ; Restore IPL
50   00A8 C5   D0  06E4  1417           MOVL    UCB$L_NO_BUFFER(R5),R0          ; Get NOB address
  51   73 A0   9A  06E9  1418           MOVZBL  NOB$B_BFN(R0),R1                ; Get number of receive buffers
  52   3A A3   3C  06ED  1419           MOVZWL  IRP$L_MEDIA+2(R3),R2            ; Get message size from P1
    04 38 A3   E8  06F1  1420           BLBS    IRP$L_MEDIA(R3),40$             ; Br if P1 buffer is valid
  52   42 A5   3C  06F5  1421           MOVZWL  UCB$W_DEVBUFSIZ(R5),R2          ; Else, get buff size from UCB
     50   14   3C  06F9  1422 40$:        MOVZWL  #SS$_BADPARAM,R0                ; Assume bad parameter
     52   51   C4  06FC  1423           MULL    R1,R2                           ; Compute total needed for buff
                73   13  06FF  1424           BEQL    60$                             ; Br if zero - error
     57   52   3C  0701  1425           MOVZWL  R2,R7                           ; Copy quota
     57   52   D1  0704  1426           CMPL    R2,R7                           ; Overflow?
                6B   12  0707  1427           BNEQ    60$                             ; Br if error
                53   DD  0709  1428           PUSHL   R3                              ; Save R3
00000000'GF     16   070B  1429           JSB     G^EXE$BUFQUOPRC                 ; Check caller's quota
                53 8ED0  0711  1430           POPL    R3                              ; Restore R3
      5D 50   E9  0714  1431           BLBC    R0,60$                          ; Br if error
   40 A3   57   B0  0717  1432           MOVW    R7,IRP$W_QUOTA(R3)              ; Save quota in packet
50   0080 C4   D0  071B  1433           MOVL    PCB$L_JIB(R4),R0                ; Get JIB address
  20 A0   57   C2  0720  1434           SUBL    R7,JIB$L_BYTCNT(R0)             ; Charge user for rec. bufs
                       0724  1435
                       0724  1436 ; The following call is used to allocate a buffer for the protocol, DDCMP.
```

```
                                0724  1437 ; If the routine returns with a LBC the the buffer could not be allocated and
                                0724  1438 ; the device can not be started.
                                0724  1439
        51    2C    000001C8 8F    C1  0724  1440          ADDL3   #TF$K_LENGTH,#GF$K_LENGTH,R1
                          0076     30  072C  1441          BSBW    ALLOC_BUFFER
                        42 50      E9  072F  1442          BLBC    R0,60$                          ; If LBC then abort startup
                                   0732  1443          DSBINT  UCB$B_FIPL(R5)
                    50    00A8 C5    D0  0739  1444          MOVL    UCB$L_NO_BUFFER(R5),R0          ; Get NOB address
                      0C  A0    52    D0  073E  1445          MOVL    R2,NOB$A_PRO_BUFFER(R0)         ; Set protocol addess
                                3F    BB  0742  1446          PUSHR   #^M<R0,R1,R2,R3,R4,R5>          ; Before queueing request zero
    01C8 C2     2C    00    65    00    2C  0744  1447          MOVC5   #0,(R5),#0,#GF$K_LENGTH,TF$K_LENGTH(R2) ; the GFB portion of
                                3F    BA  074C  1448          POPR    #^M<R0,R1,R2,R3,R4,R5>          ; buffer
                                   074E  1449          ENBINT
                          031C     31  0751  1450          BRW     QUEPKT                          ; Queue request to driver
                                   0754  1451
                                   0754  1452 ; No modifier specified - change controller parameters
                                   0754  1453
                                   0754  1454 50$:     SETIPL  UCB$B_FIPL(R5)                  ; Sync access to UCB
                             06    E0  0758  1455          BBS     #NO_DS_V_INITED,-                ; Br if already inited
                      1D 68 A5         075A  1456                  UCB$W_DEVSTS(R5),70$
                      03 38 A3    E9  075D  1457          BLBC    IRP$L_MEDIA(R3),53$             ; Br if no P1 buffer
                         44 A5    94  0761  1458          CLRB    UCB$L_DEVDEPEND(R5)             ; Clear old UCB characteristics
                          0415    30  0764  1459 53$:     BSBW    CHG_UCB_NOB                     ; Update the UCB and the NOB
                       50   01    3C  0767  1460          MOVZWL  S^#SS$_NORMAL,R0                ; Else, set success
                     51    44 A5    D0  076A  1461          MOVL    UCB$L_DEVDEPEND(R5),R1          ; Set IOSB1 return
                00000000'GF       17  076E  1462 55$:     JMP     G^EXE$FINISHIO                  ; Finish the I/O request
                                   0774  1463
                00000000'GF       17  0774  1464 60$:     JMP     G^EXE$ABORTIO                   ; Abort the I/O request
                                   077A  1465
                                   077A  1466 ; Device already inited - set new parameters and give them to device
                                   077A  1467
                   0F 38 A3       E9  077A  1468 70$:     BLBC    IRP$L_MEDIA(R3),80$             ; Br if no P1 buffer
                      3C A3       91  077E  1469          CMPB    IRP$L_MEDIA+4(R3),-             ; Are characteristics okay?
                      44 A5           0781  1470                  UCB$L_DEVDEPEND(R5)
                         08       13  0783  1471          BEQL    80$                             ; Yes - let it go
                    50   14       3C  0785  1472          MOVZWL  #SS$_BADPARAM,R0                ; Return error
                    51   01       CE  0788  1473          MNEGL   S^#1,R1                         ; No specific parameter
                         E1       11  078B  1474          BRB     55$                             ; Complete the I/O
                                   078D  1475
                                   078D  1476 80$:     SETIPL  UCB$B_FIPL(R5)                  ; Sync to get UCB
                          03E8     30  0791  1477          BSBW    CHG_UCB_NOB                     ; Update the UCB and the NOB
                    50    F0 8F    90  0794  1478          MOVB    #<15@4>,R0.                     ; Set only four parameters
                       50   01    B0  0798  1479          MOVW    S^#SS$_NORMAL,R0
                51    44 A5       D0  079B  1480          MOVL    UCB$L_DEVDEPEND(R5),R1
                00000000'GF       17  079F  1481          JMP     G^EXE$FINISHIO
```

```
                     07A5  1483          .SBTTL  ALLOC_BUFFER, Allocate a buffer
                     07A5  1484 ;++
                     07A5  1485 ;ALLOC_BUFFER - Allocate protocol buffer routine
                     07A5  1486 ;
                     07A5  1487 ; This routine does all the necesary operations to allocate a buffer from
                     07A5  1488 ; nonpaged pool.
                     07A5  1489 ;
                     07A5  1490 ;         INPUTS:
                     07A5  1491 ;                 R1 = Buffer size
                     07A5  1492 ;                 R4 = Address of the PCB
                     07A5  1493 ;
                     07A5  1494 ;         OUTPUTS:
                     07A5  1495 ;                 R2 = Address of buffer allocated
                     07A5  1496 ;                 R3 is preserved
                     07A5  1497 ;
                     07A5  1498 ;--
                     07A5  1499 ALLOC_BUFFER:
                 53   DD  07A5  1500          PUSHL   R3
  00000000'GF    16   07A7  1501          JSB     G^EXE$BUFFRQUOTA                ; Does the user have quota
        1D 50    E9   07AD  1502          BLBC    R0,10$                          ; If LBC no quota
  00000000'GF    16   07B0  1503          JSB     G^EXE$ALLOCBUF                  ; Allocate the buffer
        14 50    E9   07B6  1504          BLBC    R0,10$                          ; If LBC then not allocated
50   0080 C4    D0   07B9  1505          MOVL    PCB$L_JIB(R4),R0                ; Get the users JIB
  20 A0    51    C2   07BE  1506          SUBL2   R1,JIB$L_BYTCNT(R0)             ; Subtract the quota
  08 A2    51    B0   07C2  1507          MOVW    R1,UCB$W_SIZE(R2)               ; Save the size of BUFFER allocated
  0A A2    13    B0   07C6  1508          MOVW    S^#DYN$C_BUFIO,UCB$B_TYPE(R2)   ; Set buffer type
     50    01    3C   07CA  1509          MOVZWL  S^#SS$_NORMAL,R0                ; Set successful return
           53 8ED0   07CD  1510 10$:      POPL    R3
                05   07D0  1511          RSB
                     07D1  1512
```

```
                        07D1  1514          .SBTTL  INIT_NO_BUFFER - Initialize NOB extension
                        07D1  1515
                        07D1  1516 ;++
                        07D1  1517 ; INIT_NO_BUFFER - Initialize NOB extension
                        07D1  1518 ;
                        07D1  1519 ; Functional description:
                        07D1  1520 ;
                        07D1  1521 ; This routine is called to initialize queues and reset parameters when the
                        07D1  1522 ; NOB has been allocated on a startup.
                        07D1  1523 ;
                        07D1  1524 ;       INPUTS:
                        07D1  1525 ;               R4 = NOB address
                        07D1  1526 ;               R5 = UCB address
                        07D1  1527 ;
                        07D1  1528 ;       IPL = FIPL
                        07D1  1529 ;
                        07D1  1530 ;       OUTPUTS:
                        07D1  1531 ;               R0-R2 are destroyed.
                        07D1  1532 ;
                        07D1  1533 ;--
                        07D1  1534 INIT_NO_BUFFER:
                        07D1  1535
                        07D1  1536 ; Initialize queue headers
                        07D1  1537
18 A4    18 A4    9E    07D1  1538          MOVAB   NOB$Q_RCVS(R4),NOB$Q_RCVS(R4)   ; Receive list
1C A4    18 A4    9E    07D6  1539          MOVAB   NOB$Q_RCVS(R4),NOB$Q_RCVS+4(R4)
20 A4    20 A4    9E    07DB  1540          MOVAB   NOB$Q_FREE(R4),NOB$Q_FREE(R4)   ; Free buffer list
24 A4    20 A4    9E    07E0  1541          MOVAB   NOB$Q_FREE(R4),NOB$Q_FREE+4(R4)
28 A4    28 A4    9E    07E5  1542          MOVAB   NOB$Q_POST(R4),NOB$Q_POST(R4)   ; Post list
2C A4    28 A4    9E    07EA  1543          MOVAB   NOB$Q_POST(R4),NOB$Q_POST+4(R4)
10 A4    10 A4    9E    07EF  1544          MOVAB   NOB$Q_ATTN(R4),NOB$Q_ATTN(R4)   ; Full buffer list
14 A4    10 A4    9E    07F4  1545          MOVAB   NOB$Q_ATTN(R4),NOB$Q_ATTN+4(R4)
                        07F9  1546
                        07F9  1547          ASSUME  NOB$L_XMT_INPR EQ NOB$L_RCV_INPR+4
                        07F9  1548
          30 A4    7C   07F9  1549          CLRQ    NOB$L_RCV_INPR(R4)
          42 A4    94   07FC  1550          CLRB    NOB$B_XSTATE(R4)                ; Clear xmter state
44 A4   FFFF 8F    B0   07FF  1551          MOVW    #NO$C_PAD,NOB$W_PADS(R4)        ; Set up field with pads to send
                        0805  1552
                        0805  1553
                        0805  1554 ; Enable the receiver. DDCMP STRT messages could be coming in on the
                        0805  1555 ; wire and we will start looking at them as soon as this buffer is
                        0805  1556 ; given to the device. ( The address is stored in UCB$L_NO_BUFFER.)
                        0805  1557 ; Since that is the case then we might as well start framing receive messages
                        0805  1558 ; as well. The protocol will not be brought up however until both
                        0805  1559 ; the line and circuit are turned on.
                        0805  1560
                        0805  1561          DSBINT  UCB$B_DIPL(R5)
            08BE    30  080C  1562          BSBW    START_RECEIVE
                        080F  1563          ENBINT
                        0812  1564
                        0812  1565
                        0812  1566 ; Set up the block allocated in the NOB to look like a DDCMP control message
                        0812  1567 ; buffer
                        0812  1568
50    0080 C4    DE    0812  1569          MOVAL   NOB$Z_CTL_MSG(R4),R0
        0C A0    D4    0817  1570          CLRL    XMTQ$C_IRP(R0)
```

```
                1F A0    04    90  081A  1571          MOVB    #XMTQ$M_CONTROL,XMTQ$B_FLAG(R0)
                                   081E  1572
                                   081E  1573 ; Please note that setting this of this parameter on device startup is
                                   081E  1574 ; accomplished via the NOB$W_DEVBUFSIZ field in the NOB buffer. This
                                   081E  1575 ; was done to make the validating and the setting of this field simpler
                                   081E  1576
          42 A5   0100 8F    B0  081E  1577          MOVW    #NO$C_DEF_BUFSIZ,UCB$W_DEVBUFSIZ(R5) ; Set default buffer size
                50    06    3C  0824  1578          MOVZWL  #DEF_LINE_PARAMSZ,R0                  ; Set size of defaults in bytes
             51   F8D3 CF    9E  0827  1579          MOVAB   DEF_LINE_PARAM,R1                     ; Set address of defaults
               52    70 A4    9E  082C  1580          MOVAB   NOB$C_SETPRM(R4),R2                   ; Set address of parameters
                82    81    90  0830  1581 10$:     MOVB    (R1)+,(R2)+                           ; Set next default
                  FA 50    F5  0833  1582          SOBGTR  R0,10$                                ; Loop on all parameters
                50    0A    3C  0836  1583          MOVZWL  #DEF_TRIB_PARAMSZ,R0                  ; Set size of defaults in bytes
             51   F8B7 CF    9E  0839  1584          MOVAB   DEF_TRIB_PARAM,R1                     ; Set address of defaults
               52    60 A4    9E  083E  1585          MOVAB   NOB$Z_DDCMP(R4),R2                    ; Set address of parameters
                82    81    90  0842  1586 20$:     MOVB    (R1)+,(R2)+                           ; Set next default
                  FA 50    F5  0845  1587          SOBGTR  R0,20$                                ; Loop on all parameters
                            05  0848  1588          RSB
                                0849  1589
```

```
                     0849  1591          .SBTTL  NO$SENSEMODEFDT,  Sense Mode I/O operation FDT routine
                     0849  1592
                     0849  1593 ;++
                     0849  1594 ; NO$SENSEMODEFDT - Sense Mode FDT routine
                     0849  1595 ;
                     0849  1596 ; Functional Description:
                     0849  1597 ;
                     0849  1598 ;  This routine returns information to the caller about the configuration
                     0849  1599 ;  and status of the DMF32 device.  Depending on the function modifier,
                     0849  1600 ;  either the device characteristics, error counters contents are returned.
                     0849  1601 ;
                     0849  1602 ;
                     0849  1603 ;  The QIO parameters for SENSEMODE are:
                     0849  1604 ;
                     0849  1605 ;       P1 = optional address of quadword or longword buffer
                     0849  1606 ;       P2 = optional address of buffer descriptor for extended characteristics
                     0849  1607 ;
                     0849  1608 ;
                     0849  1609 ;       INPUTS:
                     0849  1610 ;               R3 = IRP address
                     0849  1611 ;               R4 = PCB address
                     0849  1612 ;               R5 = UCB address
                     0849  1613 ;               R6 = CCB address
                     0849  1614 ;               R7 = Function code
                     0849  1615 ;               AP = Address of first function-dependent QIO parameter
                     0849  1616 ;
                     0849  1617 ;       OUTPUTS:
                     0849  1618 ;               R0 = status return of sensemode request
                     0849  1619 ;               R3,R5 are preserved.
                     0849  1620 ;
                     0849  1621 ;--
                     0849  1622 NO$SENSEMODEFDT:
OD 68 A5    06   E0 0849  1623          BBS     #NO_DS_V_INITED,UCB$W_DEVSTS(R5),2$ ; If inited then ok to read
                     084E  1624                                                      ; errors and characteristics
    50    01   3C 084E  1625          MOVZWL  #SS$_NORMAL,R0                      ; Else return no information
 51    44 A5   D0 0851  1626          MOVL    UCB$L_DEVDEPEND(R5),R1
 00000000'GF   17 0855  1627          JMP     G^EXE$FINISHIO
                     085B  1628
 57    20 A3   B0 085B  1629 2$:      MOVW    IRP$W_FUNC(R3),R7                   ; Get entire function code
                     085F  1630
 03 57    09   E1 085F  1631          BBC     #IO$V_CTRL,R7,5$                    ; Br if not controller request
       00F0   31 0863  1632          BRW     SENSEMODE_CTRL                      ; Else, process controller req.
57    0500 8F   B3 0866  1633 5$:     BITW    #<IO$M_RD_COUNT!-                   ; Check to see if either bit is
                     086B  1634                  IO$M_CLR_COUNT>,R7              ; Is set
          7C   13 086B  1635          BEQL    40$                                 ; If EQL then read parameters
          50   D4 086D  1636          CLRL    R0                                  ; Assume clear count
    51    01   3C 086F  1637          MOVZWL  S^#SS$_NORMAL,R1                    ; Assume success
 3F 57    08   E1 0872  1638          BBC     #IO$V_RD_COUNT,R7,15$               ; If BC then not read count
                     0876  1639
                     0876  1640 ; Read tributary counters - modifier RD_COUNT
                     0876  1641
       026E   30 0879  1642          BSBW    CHECK_P2                            ; Check P2 buffer
    50    14   3C 0879  1643          MOVZWL  #SS$_BADPARAM,R0                    ; Assume zero length buffer
 3C A3    51   B0 087C  1644          MOVW    R1,IRP$L_MEDIA+4(R3)                ; Save user size of P2 buffer
          5D   13 0880  1645          BEQL    30$                                 ; Br if none
          54   DD 0882  1646          PUSHL   R4
    51    01   3C 0884  1647          MOVZWL  S^#SS$_NORMAL,R1                    ; Assume success
```

```
                54   00A8 C5    D0   0887  1648          MOVL    UCB$L_NO_BUFFER(R5),R4          ; Get NOB address
                          57    13   088C  1649          BEQL    35$                             ; If eql no info
                  54    0C A4    D0   088E  1650          MOVL    NOB$A_PRO_BUFFER(R4),R4         ; Get prot buffer address
                          51    13   0892  1651          BEQL    35$                             ; If eql no info
                  50    1E A4    3C   0894  1652          MOVZWL  TF$W_TEB(R4),R0                 ; Get size of buffer needed
                  50    3C A3    B1   0898  1653          CMPW    IRP$C_MEDIA+4(R3),R0            ; IF GEQU then buffer ok
                          09    1E   089C  1654          BGEQU   10$
                51    0601 8F    3C   089E  1655          MOVZWL  #SS$_BUFFEROVF,R1               ; Return partial success
                  50    3C A3    3C   08A3  1656          MOVZWL  IRP$C_MEDIA+4(R3),R0            ; Set size of copy
                          3B    BB   08A7  1657  10$:      PUSHR   #^M<R0,R1,R3,R4,R5>             ; Save the registers
              62  018A C4     50   28   08A9  1658          MOVC3   R0,TF$K_ERRSTRT(R4),(R2)        ; Get the errors
                          3B    BA   08AF  1659          POPR    #^M<R0,R1,R3,R4,R5>
                16 57     0A    E1   08B1  1660          BBC     #IO$V_CLR_COUNT,R7,20$          ; Br if not clear counts
                          57    03   3C   08B5  1661  15$:      MOVZWL  #<DLK$M_TRIB!DLK$M_CLEAR>,R7    ; Set counters to read/clear
                                     08B8  1662          SETIPL  UCB$B_FIPL(R5)                  ; Set to fork IPL
                          2B    BB   08BC  1663          PUSHR   #^M<R0,R1,R3,R5>                ; Save registers
                          58    D4   08BE  1664          CLRL    R8                              ; Set no parameters to return
                          55    54   D0   08C0  1665          MOVL    R4,R5                           ; Set address of trib block
                          56    03   9A   08C3  1666          MOVZBL  #DLK$C_REQEBA,R6                ; Set operation to perform
                             F737' 30   08C6  1667          BSBW    DDCMP                           ; Branch to protocol
                          2B    BA   08C9  1668          POPR    #^M<R0,R1,R3,R5>
                50    50    10   78   08CB  1669  20$:      ASHL    #16,R0,R0                       ; Shift size of buffer return
                          50    51   B0   08CF  1670          MOVW    R1,R0                           ; Set success status
                          54 8ED0    08D2  1671          POPL    R4
                  51    44 A5    D0   08D5  1672          MOVL    UCB$L_DEVDEPEND(R5),R1          ; Set devdepend info
                 00000000'GF     17   08D9  1673          JMP     G^EXE$FINISHIO                  ; Complete the request
                                     08DF  1674
                 00000000'GF     17   08DF  1675  30$:      JMP     G^EXE$ABORTIO                   ; Abort the I/O request
                                     08E5  1676
                          50    D4   08E5  1677  35$:      CLRL    R0
                          E2    11   08E7  1678          BRB     20$
                                     08E9  1679
                                     08E9  1680 ; Read tributary parameters
                                     08E9  1681
                     50    01    3C   08E9  1682  40$:      MOVZWL  S^#SS$_NORMAL,R0                ; Assume success
                     51    08    3C   08EC  1683          MOVZWL  S^#8,R1                         ; Size of P1 buffer if present
                        01F3    30   08EF  1684          BSBW    CHECK_BUFS                      ; Check P1 and P2 buffers
                  3C A3    51    B0   08F2  1685          MOVW    R1,IRP$L_MEDIA+4(R3)            ; Save user P2 buffer length
                          45    13   08F6  1686          BEQL    60$                             ; Br if no P2 buffer present
                                     08F8  1687
                  3E A3    01    B0   08F8  1688          MOVW    S^#SS$_NORMAL,IRP$L_MEDIA+6(R3) ; Assume success
                  32 A3    18    B0   08FC  1689          MOVW    #TRIB_PRM_BUFSIZ,IRP$W_BCNT(R3) ; Set size of required buffer
                     18    51    B1   0900  1690          CMPW    R1,#TRIB_PRM_BUFSIZ             ; If GEQU then user buffer is
                          0A    1E   0903  1691          BGEQU   50$                             ; large enough
              3E A3    0601 8F    B0   0905  1692          MOVW    #SS$_BUFFEROVF,IRP$L_MEDIA+6(R3) ; Return partial success
                 32 A3    51    B0   090B  1693          MOVW    R1,IRP$W_BCNT(R3)               ; Set size to use in move
               51    F7C1 CF    9E   090F  1694  50$:      MOVAB   TRIB_PARAM,R1                   ; Get address of return table
                          54    DD   0914  1695          PUSHL   R4
               54    00A8 C5    D0   0916  1696          MOVL    UCB$L_NO_BUFFER(R5),R4          ; Get NOB address
                          06    13   091B  1697          BEQL    55$                             ; If Neql buffer is available
                  54    60 A4    9E   091D  1698          MOVAB   NOB$Z_DDCMP(R4),R4              ; Get trib param block address
                          08    12   0921  1699          BNEQ    56$                             ; If Neql buffer is available
                 00000032'EF     B4   0923  1700  55$:      CLRW    IRP$W_BCNT
                          03    11   0929  1701          BRB     58$
                       1211    30   092B  1702  56$:      BSBW    RETURN_P2                       ; Return the P2 parameters
                          54 8ED0    092E  1703  58$:      POPL    R4
                  50    32 A3    B0   0931  1704          MOVW    IRP$W_BCNT(R3),R0               ; Return size of buffer
```

```
                50    50    10    78  0935  1705          ASHL    #16,R0,R0                       ; Shift size of buffer return
                  50    3E A3     B0  0939  1706          MOVW    IRP$L_MEDIA+6(R3),R0            ; Set size of return
                                      093D  1707
                  52    38 A3     D0  093D  1708 60$:     MOVL    IRP$L_MEDIA(R3),R2              ; Retrieve P1 buffer address
                              09    13  0941  1709          BEQL    70$                             ; Br if none
                  62    40 A5     7D  0943  1710          MOVQ    UCB$B_DEVCLASS(R5),(R2)         ; Else, return characteristics
              04 A2     44 A5     C8  0947  1711          BISL    UCB$L_DEVDEPEND(R5),4(R2)       ;       ...
                  51    44 A5     D0  094C  1712 70$:     MOVL    UCB$L_DEVDEPEND(R5),R1          ; Get device dependend info
                00000000'GF       17  0950  1713          JMP     G^EXE$FINISHIO                  ; Complete the I/O request
```

```
                                0956  1715          .SBTTL  SENSEMODE_CTRL,  Perform SENSEMODE FDT processing for controller
                                0956  1716
                                0956  1717 ;++
                                0956  1718 ; SENSEMODE_CTRL - Perform SENSEMODE FDT processing for controller
                                0956  1719 ;
                                0956  1720 ; Functional description:
                                0956  1721 ;
                                0956  1722 ; This routine performs all FDT checking for a controller SENSEMODE request.
                                0956  1723 ; The P2 buffer if present is check for write access and if okay, a system
                                0956  1724 ; buffer is allocated for temporarily saving the needed information. The
                                0956  1725 ; P1 sensemode information is returned through IRP$L_MEDIA and IRP$L_MEDIA+4.
                                0956  1726 ;
                                0956  1727 ;       INPUTS:
                                0956  1728 ;               R3 = IRP address
                                0956  1729 ;               R4 = PCB address
                                0956  1730 ;               R5 = UCB address
                                0956  1731 ;               R7 = Function code
                                0956  1732 ;
                                0956  1733 ;       OUTPUTS:
                                0956  1734 ;               R0 = status return for request
                                0956  1735 ;               R3-R5 are preserved.
                                0956  1736 ;
                                0956  1737 ;--
                                0956  1738 SENSEMODE_CTRL:                                ; Process controller sensemode FDT
        57   0500 8F    B3      0956  1739          BITW    #<IO$M_RD_COUNT!-             ; Check to see if either bit is
                                095B  1740                  IO$M_CLR_COUNT>,R7            ; Is set
                  03    12      095B  1741          BNEQ    5$                            ; If NEQ then not read parameter
                0092    31      095D  1742          BRW     40$                           ; else read parameters
          03 57   07    E1      0960  1743 5$:      BBC     #IO$V_RD_MODEM,R7,8$          ; If BC not a read modem IO
                00EA    31      0964  1744          BRW     SENSE_MODEM                   ; Else branch to read modem FDT
                  50    D4      0967  1745 8$:      CLRL    R0                            ; Assume clear count
              51  01    3C      0969  1746          MOVZWL  S^#SS$_NORMAL,R1              ; Assume success
        52    00A8 C5    D0      096C  1747          MOVL    UCB$L_NO_BUFFER(R5),R2        ; Get NOB address
                  7B    13      0971  1748          BEQL    35$                           ; If eql no buffer available
                 0C A2    D5      0973  1749          TSTL    NOB$A_PRO_BUFFER(R2)          ; Check for address of protocol buff
                  76    13      0976  1750          BEQL    35$                           ; If eql no buffer available
          3F 57   08    E1      0978  1751          BBC     #IO$V_RD_COUNT,R7,15$         ; Br if not read counters
                                097C  1752
                                097C  1753 ; Read controller counters - modifier RD_COUNT for DDCMP mode only
                                097C  1754
                0168    30      097C  1755          BSBW    CHECK_P2                      ; Check P2 buffer
              50  14    3C      097F  1756          MOVZWL  #SS$_BADPARAM,R0              ; Assume zero length buffer
          3C A3   51    B0      0982  1757          MOVW    R1,IRP$L_MEDIA+4(R3)          ; Save user size of P2 buffer
                  60    13      0986  1758          BEQL    30$                           ; Br if none
                  54    DD      0988  1759          PUSHL   R4
        54    00A8 C5    D0      098A  1760          MOVL    UCB$L_NO_BUFFER(R5),R4        ; Get NOB address
             54   0C A4    D0      098F  1761          MOVL    NOB$A_PRO_BUFFER(R4),R4       ; Set address of protocol buffer
              51  01    3C      0993  1762          MOVZWL  S^#SS$_NORMAL,R1              ; Assume success
        50    01D3 C4    3C      0996  1763          MOVZWL  GF$W_GEB+TF$K_LENGTH(R4),R0   ; Get size of buffer needed
             50   3C A3    B1      099B  1764          CMPW    IRP$L_MEDIA+4(R3),R0          ; IF GEQU then buffer ok
                  09    1E      099F  1765          BGEQU   10$
        51    0601 8F    3C      09A1  1766          MOVZWL  #SS$_BUFFEROVF,R1             ; Return partial success
             50   3C A3    3C      09A6  1767          MOVZWL  IRP$L_MEDIA+4(R3),R0          ; Set size of copy
                  3B    BB      09AA  1768 10$:      PUSHR   #^M<R0,R1,R3,R4,R5>           ; Save the registers
    62    01E7 C4    50    28      09AC  1769          MOVC3   R0,GF$K_ERRSRT+TF$K_LENGTH(R4),(R2) ; Get the errors
                  3B    BA      09B2  1770          POPR    #^M<R0,R1,R3,R4,R5>
                  54 8ED0      09B4  1771          POPL    R4
```

```
                1C 57    0A   E1  09B7  1772          BBC     #IO$V_CLR_COUNT,R7,20$                  ; Br if not clear counts
                   57    05   3C  09BB  1773 15$:     MOVZWL  #<DLK$M_GLOB!DLK$M_CLEAR>,R7            ; Set counters to read/clear
                                   09BE  1774          SETIPL  UCB$B_FIPL(R5)                          ; Set to fork IPL
                         3B   BB  09C2  1775          PUSHR   #^M<R0,R1,R3,R4,R5>                     ; Save registers
                         58   D4  09C4  1776          CLRL    R8                                      ; Set no parameters to return
             55    00A8  C5   D0  09C6  1777          MOVL    UCB$L_NO_BUFFER(R5),R5                  ; Get NOB address
               55    0C  A5   D0  09CB  1778          MOVL    NOB$A_PRO_BUFFER(R5),R5                 ; Set address of protocol buffer
                   56    03   9A  09CF  1779          MOVZBL  #DLK$C_REQEBA,R6                        ; Set operation to perform
                      F62B'   30  09D2  1780          BSBW    DDCMP                                   ; Branch to protocol
                         3B   BA  09D5  1781          POPR    #^M<R0,R1,R3,R4,R5>
             50    50    10   78  09D7  1782 20$:     ASHL    #16,R0,R0                               ; Shift size of buffer return
                   50    51   B0  09DB  1783          MOVW    R1,R0                                   ; Set success status
               51    44  A5   D0  09DE  1784          MOVL    UCB$L_DEVDEPEND(R5),R1                  ; Set devdepend info
            00000000'GF    17  09E2  1785          JMP     G^EXE$FINISHIO                          ; Complete the request
                                   09E8  1786
            00000000'GF    17  09E8  1787 30$:     JMP     G^EXE$ABORTIO                           ; Abort the I/O request
                                   09EE  1788
                         50   D4  09EE  1789 35$:     CLRL    R0
                         E5   11  09F0  1790          BRB     20$
                                   09F2  1791
                                   09F2  1792 ; Read controller parameters
                                   09F2  1793
                   50    01   3C  09F2  1794 40$:     MOVZWL  S^#SS$_NORMAL,R0                        ; Assume success
                   51    08   3C  09F5  1795          MOVZWL  S^#8,R1                                 ; Size of P1 buffer if present
                      00EA   30  09F8  1796          BSBW    CHECK_BUFS                              ; Check P1 and P2 buffers
                3C A3    51   D0  09FB  1797          MOVL    R1,IRP$L_MEDIA+4(R3)                    ; Save user P2 buffer length
                         3C   13  09FF  1798          BEQL    60$                                     ; Br if no P2 buffer
                                   0A01  1799
                3E A3    01   B0  0A01  1800          MOVW    S^#SS$_NORMAL,IRP$L_MEDIA+6(R3)         ; Assume success
                32 A3    24   B0  0A05  1801          MOVW    #LINE_PRM_BUFSIZ,IRP$W_BCNT(R3)         ; Set size of required buffer
                   24    51   B1  0A09  1802          CMPW    R1,#LINE_PRM_BUFSIZ                     ; If GEQU then user buffer is
                         0A   1E  0A0C  1803          BGEQU   50$                                     ; large enough
          3E A3    0601   8F   B0  0A0E  1804          MOVW    #SS$_BUFFEROVF,IRP$L_MEDIA+6(R3)        ; Return partial success
                32 A3    51   B0  0A14  1805          MOVW    R1,IRP$W_BCNT(R3)                       ; Set size to use in move
             51    F684  CF   9E  0A18  1806 50$:     MOVAB   LINE_PARAM,R1                           ; Get address of return table
                         54   DD  0A1D  1807          PUSHL   R4
             54    00A8  C5   D0  0A1F  1808          MOVL    UCB$L_NO_BUFFER(R5),R4                  ; Get line param block address
                         05   12  0A24  1809          BNEQ    52$                                     ; If neq buffer available
                   32 A3    B4  0A26  1810          CLRW    IRP$W_BCNT(R3)
                         03   11  0A29  1811          BRB     54$
                      1111   30  0A2B  1812 52$:     BSBW    RETURN_P2                               ; Return the P2 parameters
                         54 8ED0  0A2E  1813 54$:     POPL    R4
                50    32 A3    B0  0A31  1814 55$:     MOVW    IRP$W_BCNT(R3),R0                       ; Return size of buffer
             50    50    10   78  0A35  1815          ASHL    #16,R0,R0                               ; Shift size of buffer return
                50    3E A3    B0  0A39  1816          MOVW    IRP$L_MEDIA+6(R3),R0                    ; Set size of return
                                   0A3D  1817
                52    38 A3    D0  0A3D  1818 60$:     MOVL    IRP$L_MEDIA(R3),R2                      ; Retrieve P1 buffer address
                         04   13  0A41  1819          BEQL    70$                                     ; Br if none
                62    40 A5    7D  0A43  1820          MOVQ    UCB$B_DEVCLASS(R5),(R2)                 ; Else, return characteristics
                51    44 A5    D0  0A47  1821 70$:     MOVL    UCB$L_DEVDEPEND(R5),R1                  ; Get device dependend info
            00000000'GF    17  0A4B  1822          JMP     G^EXE$FINISHIO                          ; Complete the I/O request
                                   0A51  1823
```

```
                        0A51  1825          .SBTTL  SENSE_MODEM - Perform SENSEMODE READ_MODEM FDT processing
                        0A51  1826 ;++
                        0A51  1827 ;SENSE_MODEM - Perform SENSEMODE READ_MODEM FDT processing
                        0A51  1828 ;
                        0A51  1829 ; This routine performs all FDT checking for a SENSEMODE read modem reqeust.
                        0A51  1830 ; The P1 buffer is checked for write access and the request is queued to
                        0A51  1831 ; the driver.
                        0A51  1832 ;
                        0A51  1833 ;       INPUTS:
                        0A51  1834 ;               R3 = IRP address
                        0A51  1835 ;               R4 = PCB address
                        0A51  1836 ;               R5 = UCB address
                        0A51  1837 ;               R7 = Fucntion code
                        0A51  1838 ;
                        0A51  1839 ;       OUTPUTS:
                        0A51  1840 ;               R0 = Status for return
                        0A51  1841 ;               R3-R5 are preserved
                        0A51  1842 ;--
                        0A51  1843 SENSE_MODEM:
              51   04  3C  0A51  1844          MOVZWL  S^#4,R1                         ; Size of P1 buffer
                 00BA  30  0A54  1845          BSBW    CHECK_P1                        ; Check access to P1
                        0A57  1846                                                    ; (no return on no access)
                 00CA  30  0A57  1847          BSBW    ALLOC_P2BUF                     ; Allocate a P2 buffer
                03 50  E8  0A5A  1848          BLBS    R0,10$                          ; Branch if success
                 FA7D  31  0A5D  1849          BRW     ABORTIO                         ; Else abort the request
                        0A60  1850
                        0A60  1851 ;& WHAT DO WE DO HERE?????
                        0A60  1852
              51   2C A3  D0  0A60  1853 10$:    MOVL    IRP$L_SVAPTE(R3),R1             ; Get system buffer address
        04 A1    38 A3  D0  0A64  1854          MOVL    IRP$L_MEDIA(R3),P2B_L_BUFFER(R1) ; Set user buffer VA
                 38 A3  D4  0A69  1855          CLRL    IRP$L_MEDIA(R3)                 ; Clear buffer
           21 A3    04  90  0A6C  1856          MOVB    #NO_FC_V_READ_MODEM,IRP$B_NOFUNC(R3) ; Set funcition
                        0A70  1857          ;BRB    QUEPKT                          ;Give request o driver
                        0A70  1858
                        0A70  1859
                        0A70  1860
                        0A70  1861 ; I/O request packet to driver
                        0A70  1862
                        0A70  1863 QUEPKT:
                        0A70  1864          SETIPL  UCB$B_FIPL(R5)                  ; Queue packet to driver
     00000000'GF    16  0A74  1865          JSB     G^IOC$INITIATE                  ; Initiate I/O request
     00000000'GF    17  0A7A  1866          JMP     G^EXE$QIORETURN                 ; Lower IPL, and return
                        0A80  1867
```

```
                        0A80  1869          .SBTTL  GET_CHAR_BUFS,  Get P1 and P2 characteristics buffers
                        0A80  1870
                        0A80  1871 ;++
                        0A80  1872 ; GET_CHAR_BUFS - Get P1 and P2 characteristics buffers
                        0A80  1873 ;
                        0A80  1874 ; Functional description:
                        0A80  1875 ;
                        0A80  1876 ; This routine saves the P1 and P2 buffers for later use by the driver.
                        0A80  1877 ; The P1 buffer is saved in the IRP (IRP$L_MEDIA) as a quadword value.
                        0A80  1878 ; The P2 buffer is saved by allocating the appropriate amount of memory from
                        0A80  1879 ; non-paged pool.   The user's quota is checked before the allocation is made.
                        0A80  1880 ; And the non-paged pool buffer is charged against the user's quota. The P2
                        0A80  1881 ; system buffer address is passed in IRP$L_SVAPTE of the IRP.
                        0A80  1882 ;
                        0A80  1883 ;
                        0A80  1884 ;       INPUTS:
                        0A80  1885 ;               R3 = IRP address
                        0A80  1886 ;               R4 = PCB address
                        0A80  1887 ;               R5 = UCB address
                        0A80  1888 ;
                        0A80  1889 ;       OUTPUTS:
                        0A80  1890 ;               R0 = status of buffers
                        0A80  1891 ;               R3-R5 are preserved.
                        0A80  1892 ;
                        0A80  1893 ;--
                        0A80  1894 GET_CHAR_BUFS:                                       ; Get characteristics buffers
                        0A80  1895
                        0A80  1896 ; Check access to P1 buffer and save P1 characteristics
                        0A80  1897
      38 A3     7C      0A80  1898          CLRQ    IRP$L_MEDIA(R3)                   ; Reset P1 chars
   52   6C     D0      0A83  1899          MOVL    P1(AP),R2                         ; Get address of P1 char buf
        11     13      0A86  1900          BEQL    10$                               ; Branch if no P1 buffer
   50   0C     3C      0A88  1901          MOVZWL  #SS$_ACCVIO,R0                    ; Assume access violation
                       0A8B  1902          IFNORD  #8,(R2),20$                       ; Check access
38 A3   62     7D      0A91  1903          MOVQ    (R2),IRP$L_MEDIA(R3)              ; Save P1 characteristics
38 A3   01     90      0A95  1904          MOVB    #1,IRP$L_MEDIA(R3)                ; Indicate valid P1 buffer
                       0A99  1905
                       0A99  1906 ; Check access to P2 buffer and check process's buffer quota
                       0A99  1907
52   04 AC     D0      0A99  1908 10$:     MOVL    P2(AP),R2                         ; Get address P2 char buf desc
        42     13      0A9D  1909          BEQL    40$                               ; Br if no P2 buffer
   50   0C     3C      0A9F  1910          MOVZWL  #SS$_ACCVIO,R0                    ; Assume access violation
                       0AA2  1911          IFNORD  #8,(R2),20$                       ; Check access to descriptor
   51   62     3C      0AA8  1912          MOVZWL  (R2),R1                           ; Get buffer length in bytes
   51   01     CA      0AAB  1913          BICL    #1,R1                             ; Must be multiple of 2 bytes
        31     13      0AAE  1914          BEQL    40$                               ; Br if size is zero
52   04 A2     D0      0AB0  1915          MOVL    DSC$A_POINTER(R2),R2              ; Get buffer address
   50   52     D0      0AB4  1916          MOVL    R2,R0                             ; Copy buffer address
        0C     BB      0AB7  1917          PUSHR   #^M<R2,R3>                        ; Save R2, R3
00000000'GF     16      0AB9  1918          JSB     G^EXE$WRITECHKR                   ; Check entire buffer
      06   50   E9      0ABF  1919          BLBC    R0,15$                            ; Branch if error
0000000C'GF     16      0AC2  1920          JSB     G^EXE$BUFQUOPRC                   ; Check for buffered quota
        0C     BA      0AC8  1921 15$:     POPR    #^M<R2,R3>                        ; Restore R2, R3
      01   50   E8      0ACA  1922          BLBS    R0,30$                            ; Branch if quota ok
                   05  0ACD  1923 20$:     RSB                                       ; Return
                       0ACE  1924
                       0ACE  1925
```

```
                                0ACE  1926 ; Quota OKAY, allocate buffer and copy info.
                                0ACE  1927
                          0053  30  0ACE  1928 30$:    BSBW    ALLOC_P2BUF                     ; Allocate buffer
                        10 50   E9  0AD1  1929         BLBC    R0,50$                          ; Br if error
                     50   2C A3  D0  0AD4  1930         MOVL    IRP$L_SVAPTE(R3),R0             ; Get P2 buffer address
                             38  BB  0AD8  1931         PUSHR   #^M<R3,R4,R5>                   ; Save sacred registers
                0C A0   62   51  28  0ADA  1932         MOVC3   R1,(R2),P2B_T_DATA(R0)          ; Save P2 char buffer
                             38  BA  0ADF  1933         POPR    #^M<R3,R4,R5>                   ; Restore registers
                        50   01  3C  0AE1  1934 40$:    MOVZWL  S^#SS$_NORMAL,R0                ; Set success
                                 05  0AE4  1935 50$:    RSB                                     ; Return
```

```
                     0AE5  1937          .SBTTL  CHECK_BUFS, Check P1 and P2 buffers for write access
                     0AE5  1938
                     0AE5  1939 ;++
                     0AE5  1940 ; CHECK_BUFS - Check P1 and P2 buffers for write access
                     0AE5  1941 ;
                     0AE5  1942 ; Functional description:
                     0AE5  1943 ;
                     0AE5  1944 ; This routines checks the P1 and P2 buffers for write access if supplied.
                     0AE5  1945 ;
                     0AE5  1946 ;       INPUTS:
                     0AE5  1947 ;               R1 = Size of P1 buffer needed for write access
                     0AE5  1948 ;               R3 = IRP address
                     0AE5  1949 ;               R4 = PCB address
                     0AE5  1950 ;               R5 = UCB address
                     0AE5  1951 ;               R7 = Function code
                     0AE5  1952 ;               R9 = CDB address
                     0AE5  1953 ;
                     0AE5  1954 ;       OUTPUTS:
                     0AE5  1955 ;               R1 = Length of P2 buffer (zero if no P2 buffer)
                     0AE5  1956 ;               R2 = Address of P2 buffer in user's process space
                     0AE5  1957 ;               R0-R1 are destroyed.
                     0AE5  1958 ;
                     0AE5  1959 ;       No RETURN on NO ACCESS
                     0AE5  1960 ;
                     0AE5  1961 ;       IMPLICIT OUTPUTS:
                     0AE5  1962 ;
                     0AE5  1963 ;               IRP$V_FUNC bit set in IRP$W_STS by EXE$READCHK subroutine.
                     0AE5  1964 ;
                     0AE5  1965 ;--
                     0AE5  1966 CHECK_BUFS:
          2A   10    0AE5  1967          BSBB    CHECK_P1                        ; Check P1 buffer
                     0AE7  1968 CHECK_P2:
          51   D4    0AE7  1969          CLRL    R1                              ; Assume no P2 buffer desc
    52    04   AC    D0    0AE9  1970    MOVL    P2(AP),R2                       ; Get address of P2 desc
          1B   13    0AED  1971          BEQL    10$                             ; Br if no P2
                     0AEF  1972          IFNORD  #8,(R2),ACCESS                  ; Br if no access
       51   62   3C  0AF5  1973          MOVZWL  (R2),R1                         ; Get length of buffer
       51   01   CA  0AF8  1974          BICL    #1,R1                           ; Must be multiple of 2 bytes
            0D   13  0AFB  1975          BEQL    10$                             ; Br if zero
    50    04   A2    D0    0AFD  1976    MOVL    DSC$A_POINTER(R2),R0            ; Get buffer address
00000000'GF      16  0B01  1977          JSB     G^EXE$READCHK                   ; Check write access to buffer
                     0B07  1978                                                  ; (no return no access)
                     0B07  1979                                                  ; Also sets IRP$V_FUNC in IRP
       52   50   D0  0B07  1980          MOVL    R0,R2                           ; Copy buffer address
            05   0B0A  1981 10$:        RSB                                     ; Return to caller
                     0B0B  1982
       50   0C   3C  0B0B  1983 ACCESS: MOVZWL  #SS$_ACCVIO,R0                  ; Return access violation
          F9CC   31  0B0E  1984          BRW     ABORTIO                         ; Abort the I/O request
```

```
                     0B11  1986          .SBTTL  CHECK_P1,  Check P1 buffer address for write access
                     0B11  1987
                     0B11  1988 ;++
                     0B11  1989 ; CHECK_P1 - Check P1 buffer address for write access
                     0B11  1990 ;
                     0B11  1991 ; Functional description:
                     0B11  1992 ;
                     0B11  1993 ;  This routine checks the P1 buffer and if okay, the buffer address
                     0B11  1994 ;  is saved in IRP$L_MEDIA of the IRP.
                     0B11  1995 ;
                     0B11  1996 ;       INPUTS:
                     0B11  1997 ;               R1 = Size of buffer for write access
                     0B11  1998 ;               R3 = IRP address
                     0B11  1999 ;               R4 = PCB address
                     0B11  2000 ;               R5 = UCB address
                     0B11  2001 ;               R7 = Function code
                     0B11  2002 ;               R9 = CDB address
                     0B11  2003 ;
                     0B11  2004 ;       OUTPUTS:
                     0B11  2005 ;               R0 is destroyed.
                     0B11  2006 ;
                     0B11  2007 ;       No RETURN on NO ACCESS.
                     0B11  2008 ;
                     0B11  2009 ;       IMPLICIT OUTPUTS:
                     0B11  2010 ;
                     0B11  2011 ;               IRP$L_MEDIA(R3) = User P1 buffer address.
                     0B11  2012 ;               IRP$V_FUNC bit set in IRP$W_STS by EXE$READCHK subroutine.
                     0B11  2013 ;
                     0B11  2014 ;--
                     0B11  2015 CHECK_P1:
        38 A3   D4   0B11  2016          CLRL    IRP$L_MEDIA(R3)                 ; Assume no P1 buffer
     50    6C   D0   0B14  2017          MOVL    P1(AP),R0                       ; Get address of user buffer
           0A   13   0B17  2018          BEQL    10$                             ; Br if none
00000000'GF    16   0B19  2019          JSB     G^EXE$READCHK                   ; Check access to buffer
                    0B1F  2020                                                  ; (No return - no access)
 38 A3    50   D0   0B1F  2021          MOVL    R0,IRP$L_MEDIA(R3)              ; Save P1 buffer address in IRP
                    05   0B23  2022 10$:     RSB                             ; Return to caller
```

```
                     0B24  2024          .SBTTL  ALLOC_P2BUF,  Allocate a P2 buffer and charge user's quota
                     0B24  2025
                     0B24  2026 ;++
                     0B24  2027 ; ALLOC_P2BUF - Allocate a P2 buffer and charge user's quota
                     0B24  2028 ;
                     0B24  2029 ; Functional description:
                     0B24  2030 ;
                     0B24  2031 ; This routine allocates a system buffer and returns the address in the IRP at
                     0B24  2032 ; IRP$L_SVAPTE.  The size of the allocation, including buffer header must
                     0B24  2033 ; be at least 24 bytes in length.
                     0B24  2034 ;
                     0B24  2035 ;       INPUTS:
                     0B24  2036 ;               R1 = Size of allocation desired
                     0B24  2037 ;               R3 = IRP address
                     0B24  2038 ;
                     0B24  2039 ;       OUTPUTS:
                     0B24  2040 ;               R0 = status of request
                     0B24  2041 ;               R1-R5 are preserved.
                     0B24  2042 ;
                     0B24  2043 ;       IMPLICIT OUTPUTS:
                     0B24  2044 ;               IRP$L_SVAPTE(R3) = address of system buffer
                     0B24  2045 ;               IRP$W_BOFF(R3) = byte count charged to user's process
                     0B24  2046 ;               IRP$W_BCNT(R3) = original byte count requested
                     0B24  2047 ;
                     0B24  2048 ;       All parts of the P2 buffer header are initialized, except for the
                     0B24  2049 ;       user's P2 buffer address.
                     0B24  2050 ;
                     0B24  2051 ;--
                     0B24  2052 ALLOC_P2BUF:                                    ; Allocate a non-paged buffer
          51    D5   0B24  2053          TSTL    R1                           ; Zero length buffer?
          50    13   0B26  2054          BEQL    30$                          ; Br if yes
          0E    BB   0B28  2055          PUSHR   #^M<R1,R2,R3>                ; Save registers
    32 A3     51    B0   0B2A  2056          MOVW    R1,IRP$W_BCNT(R3)            ; Save original byte count
       0C     51    D1   0B2E  2057          CMPL    R1,S^#24+P2B_C_LENGTH        ; Is buffer big enough?
          03    1A   0B31  2058          BGTRU   5$                           ; Br if yes
       51    0C    D0   0B33  2059          MOVL    S^#24-P2B_C_LENGTH,R1        ; Else, set size to minimum
       51    0C    C0   0B36  2060 5$:      ADDL2   S^#P2B_C_LENGTH,R1           ; Add in size of header
 00000000'GF     16   0B39  2061          JSB     G^EXE$BUFQUOPRC              ; Check for buffered quota
          0D 50    E9   0B3F  2062          BLBC    R0,10$                       ; Branch if quota bad
                        0B42  2063
                        0B42  2064 ; Quota OKAY, allocate buffer and copy info.
                        0B42  2065
          51    DD   0B42  2066          PUSHL   R1                           ; Save size to charge user
 00000000'GF     16   0B44  2067          JSB     G^EXE$ALONONPAGED            ; Go allocate a buffer
          05 50    E8   0B4A  2068          BLBS    R0,20$                       ; Br if success
          8E    D5   0B4D  2069          TSTL    (SP)+                        ; Pop saved size
          0E    BA   0B4F  2070 10$:     POPR    #^M<R1,R2,R3>                ; Restore registers
                05   0B51  2071          RSB                                  ; Return with error code in R0
                        0B52  2072
                        0B52  2073 ; System buffer allocated decrement user's quota
                        0B52  2074
          53 8ED0   0B52  2075 20$:     POPL    R3                           ; Restore user quota charge
    62    0C A2     9E   0B55  2076          MOVAB   P2B_T_DATA(R2),P2B_L_POINTER(R2) ; Set address to start of data
    08 A2     53    B0   0B59  2077          MOVW    R3,P2B_W_SIZE(R2)            ; Save buffer size in buffer
    0A A2     13    90   0B5D  2078          MOVB    S^#DYN$C_BUFIO,P2B_B_TYPE(R2) ; Set structure type
       50     52    D0   0B61  2079          MOVL    R2,R0                        ; Save P2 char buf addr
 52    0080 C4     D0   0B64  2080          MOVL    PCB$L_JIB(R4),R2             ; Get JIB address
```

```
                20 A2    53   C2  0B69  2081          SUBL    R3,JIB$L_BYTCNT(R2)             ; Decrement user's quota
                   0E    BA   0B6D  2082          POPR    #^M<R1,R2,R3>                   ; Restore registers
             2C A3    50   D0  0B6F  2083          MOVL    R0,IRP$L_SVAPTE(R3)             ; Save P2 buffer address in IRP
          30 A3    08 A0   B0  0B73  2084          MOVW    P2B_W_SIZE(R0),IRP$W_BOFF(R3)   ; Return buffer size in IRP
                50    01   3C  0B78  2085 30$:      MOVZWL  S^#SS$_NORMAL,R0                ; Set success
                           05  0B7B  2086          RSB                                     ; Return to caller
```

```
                        0B7C  2088          .SBTTL  CHG_UCB_NOB,  Change UCB and the NOB parameter values
                        0B7C  2089
                        0B7C  2090 ;++
                        0B7C  2091 ; CHG_UCB_NOB  -  Change UCB and the NOB parameter values
                        0B7C  2092 ;
                        0B7C  2093 ; Functional description:
                        0B7C  2094 ;
                        0B7C  2095 ; This routine is called to initialize the NOB with new P1 and P2 buffer
                        0B7C  2096 ; characteristics. It is assumed here that the parameters have already
                        0B7C  2097 ; been validated.
                        0B7C  2098 ;
                        0B7C  2099 ;       INPUTS:
                        0B7C  2100 ;
                        0B7C  2101 ;               R3 = IRP address
                        0B7C  2102 ;               R5 = UCB address
                        0B7C  2103 ;               R6 = Protocol mode in which to run the driver
                        0B7C  2104 ;
                        0B7C  2105 ;       IPL = FIPL
                        0B7C  2106 ;
                        0B7C  2107 ;       OUTPUTS:
                        0B7C  2108 ;               R0-R2 = destroyed.
                        0B7C  2109 ;
                        0B7C  2110 ;--
                        0B7C  2111 CHG_UCB_NOB:
              54   DD   0B7C  2112          PUSHL   R4                              ; Save R4
54   00A8 C5   D0   0B7E  2113          MOVL    UCB$L_NO_BUFFER(R5),R4          ; Get NOB address
    33 38 A3   E9   0B83  2114          BLBC    IRP$L_MEDIA(R3),10$             ; Br if no P1 buffer
                        0B87  2115
                        0B87  2116 ; Set new P1 buffer characteristics
                        0B87  2117
           06   E0   0B87  2118          BBS     #NO_DS_V_INITED,-               ; Br if device inited
        0A 68 A5        0B89  2119                  UCB$W_DEVSTS(R5),5$
                        0B8C  2120
                        0B8C  2121 ; Please note that setting this of this parameter on device startup is
                        0B8C  2122 ; usually accomplished via the NOB$W_DEVBUFSIZ field in the NOB buffer. This
                        0B8C  2123 ; was done to make the validating and the setting of this field simpler.
                        0B8C  2124
     42 A5   3A A3   B0 0B8C  2125          MOVW    IRP$L_MEDIA+2(R3),UCB$W_DEVBUFSIZ(R5) ; Set new buffer size
     74 A4   42 A5   B0 0B91  2126          MOVW    UCB$W_DEVBUFSIZ(R5),NOB$W_DEVBUFSIZ(R4) ; Make sure both are updated
                        0B96  2127
44 A5   18   08   00   F0 0B96  2128 5$:       INSV    #0,#8,#24,UCB$L_DEVDEPEND(R5)   ; Reset all Read/Write flags
           3C A3   C8 0B9C  2129          BISL    IRP$L_MEDIA+4(R3),-             ; Set new characteristics
           44 A5        0B9F  2130                  UCB$L_DEVDEPEND(R5)
                        0BA1  2131
                        0BA1  2132 ; Now update UCB based on P1 buffer
                        0BA1  2133
                        0BA1  2134          ASSUME  NMA$C_LINPR_POI EQ 0
                        0BA1  2135
           70 A4   94   0BA1  2136          CLRB    NOB$B_PRO(R4)                   ; Assume point to point mode
                        0BA4  2137
                        0BA4  2138          ASSUME  NMA$C_LINCN_NOR EQ 0
                        0BA4  2139
           72 A4   94   0BA4  2140 6$:     CLRB    NOB$B_CON(R4)                   ; Assume normal mode
              01   E1   0BA7  2141          BBC     #XM$V_CHR_LOOPB,-               ; Br if not loopback mode
        03 44 A5        0BA9  2142                  UCB$L_DEVDEPEND(R5),7$
                        0BAC  2143          ASSUME  NMA$C_LINCN_LOO EQ 1
           72 A4   96   0BAC  2144          INCB    NOB$B_CON(R4)                   ; Must be loopback mode
```

```
                        0BAF  2145 7$:     ASSUME  NMA$C_DPX_FUL EQ 0
          71 A4   94    0BAF  2146         CLRB    NOB$B_DUP(R4)                   ; Assume full duplex
             02   E1    0BB2  2147         BBC     #XM$V_CHR_HDPLX,-               ; Br if not half duplex
       03 44 A5         0BB4  2148                 UCB$L_DEVDEPEND(R5),10$
                        0BB7  2149         ASSUME  NMA$C_DPX_HAL EQ 1
          71 A4   96    0BB7  2150         INCB    NOB$B_DUP(R4)                   ; Must be half duplex
                        0BBA  2151
                        0BBA  2152
                        0BBA  2153 ; Set new P2 buffer characteristics
                        0BBA  2154
    51    F4E2 CF   9E  0BBA  2155 10$:     MOVAB   LINE_PARAM,R1                   ; Get address of verification table
             0F0C   30  0BBF  2156         BSBW    UPDATE_P2                       ; Update the UCB
                        0BC2  2157
                        0BC2  2158 ; Please note that setting this of this parameter on device startup is
                        0BC2  2159 ; usually accomplished via the NOB$W_DEVBUFSIZ field in the NOB buffer. This
                        0BC2  2160 ; was done to make the validating and the setting of this field simpler.
                        0BC2  2161
 42 A5   74 A4   B0     0BC2  2162         MOVW    NOB$W_DEVBUFSIZ(R4),UCB$W_DEVBUFSIZ(R5) ; Set def buffer size
                        0BC7  2163
                        0BC7  2164 ; Set device characteristics and device mode definition
                        0BC7  2165
          44 A5   94    0BC7  2166         CLRB    UCB$L_DEVDEPEND(R5)             ; Reset all characteristics
                        0BCA  2167 20$:     SETBIT  #XM$V_CHR_HDPLX,UCB$L_DEVDEPEND(R5) ; Assume half duplex mode
                        0BCF  2168         ASSUME  NMA$C_DPX_FUL EQ 0
          71 A4   95    0BCF  2169         TSTB    NOB$B_DUP(R4)                   ; Full duplex mode?
             05   12    0BD2  2170         BNEQ    30$                             ; Br if no - okay
                        0BD4  2171         CLRBIT  #XM$V_CHR_HDPLX,UCB$L_DEVDEPEND(R5) ; Set to full duplex mode
                        0BD9  2172 30$:     ASSUME  NMA$C_LINCN_NOR EQ 0
          72 A4   95    0BD9  2173         TSTB    NOB$B_CON(R4)                   ; Is line in normal mode?
             05   13    0BDC  2174         BEQL    40$                             ; Br if yes
                        0BDE  2175         SETBIT  #XM$V_CHR_LOOPB,UCB$L_DEVDEPEND(R5) ; Set loopback mode
             54 8ED0    0BE3  2176 40$:     POPL    R4                              ; Restore R4
                   05   0BE6  2177         RSB
                        0BE7  2178
```

```
                        0BE7  2180
                        0BE7  2181          .SBTTL  CHG_TRIB,  Change trib parameter values
                        0BE7  2182
                        0BE7  2183 ;++
                        0BE7  2184 ; CHG_TRIB - Change trib parameter values
                        0BE7  2185 ;
                        0BE7  2186 ; Functional description:
                        0BE7  2187 ;
                        0BE7  2188 ; This routine is called to initialize the trib parameter block with
                        0BE7  2189 ; new P1 and P2 buffer characteristics. It is assumed here that the
                        0BE7  2190 ; parameters have already been validated.
                        0BE7  2191 ;
                        0BE7  2192 ;       INPUTS:
                        0BE7  2193 ;               R3 = IRP address
                        0BE7  2194 ;               R5 = UCB address
                        0BE7  2195 ;
                        0BE7  2196 ;       IPL = FIPL
                        0BE7  2197 ;
                        0BE7  2198 ;       OUTPUTS:
                        0BE7  2199 ;               R1,R2 = destroyed.
                        0BE7  2200 ;
                        0BE7  2201 ;--
                        0BE7  2202 CHG_TRIB:                                    ; Validate P2 buffer
              54    DD  0BE7  2203          PUSHL   R4                        ; Save R4
54    00AB C5    D0  0BE9  2204          MOVL    UCB$L_NO_BUFFER(R5),R4    ; Get NOB address
51    F4E2 CF    9E  0BEE  2205          MOVAB   TRIB_PARAM,R1             ; Get address of verify table
  54    60 A4    9E  0BF3  2206          MOVAB   NOB$Z_DDCMP(R4),R4        ; Get the addr of param block
    10 38 A3    E9  0BF7  2207          BLBC    IRP$L_MEDIA(R3),10$       ; Br if no P1 buffer
                        0BFB  2208
                        0BFB  2209 ; Set new P1 buffer characteristics
                        0BFB  2210
44 A5    3C A3    C8  0BFB  2211          BISL    IRP$L_MEDIA+4(R3),UCB$L_DEVDEPEND(R5) ; Set new char
                        0C00  2212
                        0C00  2213          ASSUME  NMA$C_STATE_ON EQ 0
                        0C00  2214          ASSUME  NMA$C_STATE_OFF EQ 1
                        0C00  2215
          08 A4    94  0C00  2216          CLRB    DLK$B_MAINT(R4)           ; Assume MOP
03 44 A5    00    E0  0C03  2217          BBS     #XM$V_CHR_MOP,UCB$L_DEVDEPEND(R5),10$ ; Branch BS if true
          08 A4    96  0C08  2218          INCB    DLK$B_MAINT(R4)           ; Set NORMAL mode
                        0C0B  2219
                        0C0B  2220 ; Set new P2 buffer characteristics
                        0C0B  2221
          0EC0    30  0C0B  2222 10$:     BSBW    UPDATE_P2                 ; Update trib parameters
                        0C0E  2223          CLRBIT  #XM$V_CHR_MOP,UCB$L_DEVDEPEND(R5) ; Assume NORMAL mode
       05 08 A4    E8  0C13  2224          BLBS    DLK$B_MAINT(R4),20$       ; Branch LBS if true
                        0C17  2225          SETBIT  #XM$V_CHR_MOP,UCB$L_DEVDEPEND(R5) ; Set MOP mode
             54 8ED0  0C1C  2226 20$:     POPL    R4                        ; Restore R4
                05    0C1F  2227          RSB                               ; Return to caller
                        0C20  2228
```

```
                        0C20  2230          .SBTTL  NO$STARTIO - Start setmode I/O operation
                        0C20  2231 ;++
                        0C20  2232 ; NO$STARTIO - Start setmode operation
                        0C20  2233 ;
                        0C20  2234 ; FUNCTIONAL DESCRIPTION:
                        0C20  2235 ;
                        0C20  2236 ; This routine is entered to process a setmode request.  All setmode requests
                        0C20  2237 ; are queued through the UCB to single-stream them.
                        0C20  2238 ;
                        0C20  2239 ; SETMODE FUNCTIONS --
                        0C20  2240 ;
                        0C20  2241 ;       For all functions a change in the characteristics is done.
                        0C20  2242 ;
                        0C20  2243 ;       For control startup, the UCB is initialized, and the DDCMP timer is
                        0C20  2244 ;       started.
                        0C20  2245 ;       For trib startup, the protocol is set up with charateristics and
                        0C20  2246 ;       started, and transmits and receives are started on the board.
                        0C20  2247 ;
                        0C20  2248 ;       For control shutdown, the DDCMP timer is stopped,
                        0C20  2249 ;       all quotas are returned; and a call is made to trib shutdown to shut
                        0C20  2250 ;       the trib.
                        0C20  2251 ;       For trib shutdown, all buffers and IRP's are returned,
                        0C20  2252 ;       and the protocol is halted.
                        0C20  2253 ;
                        0C20  2254 ;       INPUTS:
                        0C20  2255 ;               R3 = I/O packet address
                        0C20  2256 ;               R5 = UCB address
                        0C20  2257 ;
                        0C20  2258 ;       OUTPUTS:
                        0C20  2259 ;               R3 and R5 preserved.
                        0C20  2260 ;
                        0C20  2261 ;--
                        0C20  2262          .ENABL  LSB
                        0C20  2263 NO$STARTIO:                                  ; Start I/O routine
51    21 A3    9A      0C20  2264          MOVZBL  IRP$B_NOFUNC(R3),R1         ; Get the function code
      54    DD          0C24  2265          PUSHL   R4
      0D    10          0C26  2266          BSBB    START_DISP                  ; Branch to case
      54 8ED0           0C28  2267          POPL    R4
51    44 A5    D0      0C2B  2268          MOVL    UCB$L_DEVDEPEND(R5),R1      ; Set device characteristics
                        0C2F  2269          REQCOM                              ; Complete the request
                        0C35  2270
                        0C35  2271 START_DISP:
                        0C35  2272          $DISPATCH       R1,TYPE=B,-
                        0C35  2273          <-      ; Function              Action
                        0C35  2274                  <NO_FC_V_STRT_CIR       START_CIRCUIT>,-
                        0C35  2275                  <NO_FC_V_STRT_LIN       START_LINE>,-
                        0C35  2276                  <NO_FC_V_STOP_CIR       SHUTDOWN_CIRCUIT>,-
                        0C35  2277                  <NO_FC_V_STOP_LIN       SHUTDOWN_LINE>,-
                        0C35  2278                  <NO_FC_V_READ_MODEM     READ_MODEM>,-
                        0C35  2279          >
                        0C43  2280
                        0C43  2281 5$:      BUG_CHECK NOBUFPCKT,FATAL           ; Anything else is fatal
                        0C47  2282
                        0C47  2283          .DSABL  LSB
                        0C47  2284
```

```
                        0C47  2286          .SBTTL  START_TRANSMIT - Start transmit I/O
                        0C47  2287 ;++
                        0C47  2288 ; START_TRANSMIT - Start transmit I/O
                        0C47  2289 ;
                        0C47  2290 ; FUNCTIONAL DESCRIPTION:
                        0C47  2291 ;
                        0C47  2292 ; This routine is entered to start tranmitting over the async port driver. In
                        0C47  2293 ; order to do this the transmit is split into several pieces. The header, the
                        0C47  2294 ; header CRC, data, data CRC, and the PADS. It works by checking to see if
                        0C47  2295 ; the transmiter is busy. The transmitter is busy if the UCB$V_INT bit is set in
                        0C47  2296 ; UCB$W_STS. If the port is not busy then a transmit is started by calling
                        0C47  2297 ; the PORT_STARTIO routine and the UCB$V_INT is set in UCB#W_STS. The transmit
                        0C47  2298 ; finsihes in GETNXT routine after all the parts of the message has been
                        0C47  2299 ; transmitted. If the device is running half duplex then transmits will only be
                        0C47  2300 ; sent until the select flag is sent.
                        0C47  2301 ;
                        0C47  2302 ;       INPUTS:
                        0C47  2303 ;               R1 = Status information from routine calling START_TRANSMIT
                        0C47  2304 ;                    More specifically it is an interface between the protocol
                        0C47  2305 ;                    timer routine and this routine to tell start transmit that
                        0C47  2306 ;                    the protocol timer has expired therefore send the message.
                        0C47  2307 ;               R4 = NOB address
                        0C47  2308 ;               R5 = UCB address
                        0C47  2309 ;
                        0C47  2310 ;       OUTPUTS:
                        0C47  2311 ;               NONE
                        0C47  2312 ;
                        0C47  2313 ;--
                        0C47  2314 START_TRANSMIT:
                        0C47  2315
                        0C47  2316 ; If the DDCMP timer has expired a message may need to be sent
                        0C47  2317
      05 51    0F   E0  0C47  2318          BBS     #DLK$V_TMREXPD,R1,20$
               04   E0  0C4B  2319 10$:     BBS     #NO_DS_V_XMTING,-                       ; If BS then the xmt'r is off
         45 68 A5        0C4D  2320                  UCB$W_DEVSTS(R5),25$
                         0C50  2321 20$:     DSBINT  UCB$B_DIPL(R5)
    3D 64 A5    01   E2  0C57  2322          BBSS    #UCB$V_INT,UCB$W_STS(R5),30$    ; If BS then port is busy
            42 A4    95  0C5C  2323          TSTB    NOB$B_XSTATE(R4)                ; If NEQ then the transmitter is bus
               38    12  0C5F  2324          BNEQ    30$
             007B    30  0C61  2325          BSBW    GET_XMT                         ; Get next message to send
                         0C64  2326          ENBINT
            3B 50    E9  0C67  2327          BLBC    R0,40$                          ; If LBC no message to send
  52    0118 C5    D0  0C6A  2328          MOVL    UCB$L_TT_PORT(R5),R2            ; Get port vector table
011C C5    24 A8    DE  0C6F  2329          MOVAL   XMTQ$B_MSGHDR(R8),UCB$L_TT_OUTADR(R5) ; Set address of block to send
  0120 C5    06    9B  0C75  2330          MOVZBW  #NO$C_HEADER_LEN,UCB$W_TT_OUTLEN(R5) ; Set size of block to send
               54    DD  0C7A  2331          PUSHL   R4
                         0C7C  2332          DSBINT  UCB$B_DIPL(R5)
    42 A4    01    90  0C83  2333          MOVB    #NO$C_HEADER,NOB$B_XSTATE(R4)   ; Set transmitter state
                         0C87  2334
                         0C87  2335 ; The following instruction must be the last instruction executed before
                         0C87  2336 ; the return to the port output routine. The reason for this is that the
                         0C87  2337 ; condition codes must correctly so that the port does the right transfer
                         0C87  2338
  010B C5    01    8E  0C87  2339          MNEGB   #1,UCB$B_TT_OUTYPE(R5)          ; Set block transfer to port
            00 B2    16  0C8C  2340          JSB     @PORT_STARTIO(R2)               ; Call at port startio routine
                         0C8F  2341          ENBINT
               54 8ED0  0C92  2342          POPL    R4
```

```
                50    01    3C   0C95  2343 25$:    MOVZWL  S^#SS$_NORMAL,R0
                            05   0C98  2344         RSB
                                 0C99  2345
                                 0C99  2346 30$:    CLRBIT  #UCB$V_INT,UCB$W_STS(R5)        ; Reset port in use
                                 0C9E  2347         ENBINT
                50    01    3C   0CA1  2348         MOVZWL  S^#SS$_NORMAL,R0
                            05   0CA4  2349         RSB
                                 0CA5  2350
                                 0CA5  2351 40$:    DSBINT  UCB$B_DIPL(R5)
                                 0CAC  2352         CLRBIT  #UCB$V_INT,UCB$W_STS(R5)        ; Clear nothing given to port
                                 0CB1  2353         ENBINT
          04 68 A5    09    E5   0CB4  2354         BBCC    #NO_DS_V_MSG_SENT,UCB$W_DEVSTS(R5),50$ ; If BC do not try tosend an
                50    01    3C   0CB9  2355         MOVZWL  S^#SS$_NORMAL,R0
                            05   0CBC  2356         RSB
                                 0CBD  2357
                                 0CBD  2358
                      30    BB   0CBD  2359 50$:    PUSHR   #^M<R4,R5>
                56    02    9A   0CBF  2360         MOVZBL  #DLK$C_XMTMSG,R6                ; Inform DDCMP that the last
                57    02    9A   0CC2  2361         MOVZBL  #DLK$M_QEMPTY,R7                ; msg on queue was sent
             55    0C A4    D0   0CC5  2362         MOVL    NOB$A_PRO_BUFFER(R4),R5         ; Get addr of start of TFB
                   F334'    30   0CC9  2363         BSBW    DDCMP                           ; Branch to protocol
                      30    BA   0CCC  2364         POPR    #^M<R4,R5>                      ; Restore registers
                                 0CCE  2365
                                 0CCE  2366         ASSUME DLK$V_MSGSENT EQ 0
                56    02    91   0CCE  2367         CMPB    #DLK$C_XMTMSG,R6                ; If NEQ no ACK to be xmted
                      08    12   0CD1  2368         BNEQ    60$
                05 57       E9   0CD3  2369         BLBC    R7,60$                          ; If BC no  ACK need be sent
                      51    D4   0CD6  2370         CLRL    R1                              ; Set no status info
                   FF6C     31   0CD8  2371         BRW     START_TRANSMIT                  ; Else send the ACK
                50    01    3C   0CDB  2372 60$:    MOVZWL  S^#SS$_NORMAL,R0                ; Set status
                            05   0CDE  2373         RSB
                                 0CDF  2374
```

```
                                        0CDF  2376          .SBTTL  GET_XMT - Get next transmit ti send
                                        0CDF  2377 ;++
                                        0CDF  2378 ; GET_XMT - Get next transmit to send
                                        0CDF  2379 ;
                                        0CDF  2380 ; FUNCTIONAL DESCRIPTION:
                                        0CDF  2381 ;
                                        0CDF  2382 ; This routine checks the various tranmit queues and sets up the device  with
                                        0CDF  2383 ; the next message to transmit. This means that it increments the transmit
                                        0CDF  2384 ; count and decides if the transmitter must be shut down after this message
                                        0CDF  2385 ; is sent (for half duplex only). You will notice a hack pertaining to the
                                        0CDF  2386 ; control message buffer. The reason for this is that DDCMP makes the
                                        0CDF  2387 ; assumption that it can change the data inside the control messages at
                                        0CDF  2388 ; will. There is no problem with that for most devices, however for AYSNCH
                                        0CDF  2389 ; DDCMP we calculate our own CRC's and thus if data gets changed from under
                                        0CDF  2390 ; us we will lose the message (in a most inefficient way we should note).
                                        0CDF  2391 ;
                                        0CDF  2392 ;       INPUTS:
                                        0CDF  2393 ;               R4 = NOB address
                                        0CDF  2394 ;               R5 = UCB address
                                        0CDF  2395 ;
                                        0CDF  2396 ;       IPL = DIPL
                                        0CDF  2397 ;
                                        0CDF  2398 ;       OUTPUTS:
                                        0CDF  2399 ;               R0 = status LBS then message to send, LBC no message to send
                                        0CDF  2400 ;               R4,R5 are preserved
                                        0CDF  2401 ;               R8 = address of message to send
                                        0CDF  2402 ;
                                        0CDF  2403 ;--
                                        0CDF  2404 GET_XMT:
                50    D4                0CDF  2405          CLRL    R0                                      ; Assume no message to send
           56   0C A4    D0             0CE1  2406          MOVL    NOB$A_PRO_BUFFER(R4),R6                 ; Get protocol buffer address
           51   20 B6    0F             0CE5  2407          REMQUE  @TF$Q_CTLQ(R6),R1                       ; Check control queue
                   4C    1C             0CE9  2408          BVC     50$                                     ; If VC then messaeg to send
           58   30 B6    0F             0CEB  2409          REMQUE  @TF$Q_XMTQ(R6),R8                       ; Check data queue
                   45    1D             0CEF  2410          BVS     40$                                     ; If VS then no message to send
           0A A8    13    91             0CF1  2411          CMPB    S^#DYN$C_BUFIO,XMTQ$B_BUFTYP(R8) ; If NEQ then a problem with
                   5F    12             0CF5  2412          BNEQ    60$                                     ; queues
        03 1F A8    02    E0             0CF7  2413 10$:     BBS     #XMTQ$V_CONTROL,XMTQ$B_FLAG(R8),20$ ; If BS then cntrl msg don't inc
              10 A6    96                0CFC  2414          INCB    TF$B_XQCNT(R6)
        0A 1F A8    01    E5             0CFF  2415 20$:     BBCC    #XMTQ$V_SELECT,XMTQ$B_FLAG(R8),30$ ; If BC do not turn link
        05 44 A5    02    E1             0D04  2416          BBC     #XM$V_CAR_HDPLX,UCB$L_DEVDEPEND(R5),30$ ; If Not half duplex dont' t
                            0D09  2417          SETBIT  #NO_DS_V_XMTING,UCB$W_DEVSTS(R5) ; Set xmter off, but finish xmting
              34 A4    D5             0D0E  2418 30$:     TSTL    NOB$L_XMT_INPR(R4)
                   43    12             0D11  2419          BNEQ    60$
           34 A4    58    D0             0D13  2420          MOVL    R8,NOB$L_XMT_INPR(R4)                   ; Set up xmt inprogress
                            0D17  2421
                            0D17  2422 ; Note although this may look wrong the instruction is correct because
                            0D17  2423 ; teh terminal line speed is really a two byte field. The low byte is the
                            0D17  2424 ; transmit speed and the high byte is the receive speed. If the receive
                            0D17  2425 ; speed is zero then it is the same as the transmit speed.
     50    00F4 C5    9A             0D17  2426          MOVZBL  UCB$W_TT_SPEED(R5),R0
                            0D1C  2427
                            0D1C  2428 ; Set the due time on the transmit to be given to the port.
                            0D1C  2429
   50    F3E3 CF40    3C             0D1C  2430          MOVZWL  DUETIM_TABLE[R0],R0
51    00000000'GF    D0             0D22  2431          MOVL    G^EXE$GL_ABSTIM,R1
 3C A4    51    50    C1             0D29  2432          ADDL3   R0,R1,NOB$L_XMT_DUETIM(R4)
```

```
                                 0D2E  2433          SETBIT  #NO_DS_V_XMT_TIME,UCB$W_DEVSTS(R5)
        50    01    3C           0D33  2434          MOVZWL  S^#SS$_NORMAL,R0
                    05           0D36  2435 40$:      RSB
                                 0D37  2436
  58    0080  C4    DE           0D37  2437 50$:      MOVAL   NOB$Z_CTL_MSG(R4),R8            ; Get the address of buffer to use
1F A8     1F  A1    88           0D3C  2438          BISB    XMTQ$B_FLAG(R1),XMTQ$B_FLAG(R8) ; Copy control message flags
   1F A1      03    8A           0D41  2439          BICB    #XMTQ$M_SELECT!XMTQ$M_ONQUEUE,XMTQ$B_FLAG(R1) ; Clear flags
20 A8     20  A1    B0           0D45  2440          MOVW    XMTQ$W_RCRC(R1),XMTQ$W_HCRC(R8) ; Get LRC
24 A8     24  A1    D0           0D4A  2441          MOVL    XMTQ$B_MSGHDR(R1),XMTQ$B_MSGHDR(R8) ; Get first longword of hdr
28 A8     28  A1    B0           0D4F  2442          MOVW    XMTQ$B_MSGHDR+4(R1),XMTQ$B_MSGHDR+4(R8) ; Get second word
                    A1    11     0D54  2443          BRB     10$                             ; Cont processing
                                 0D56  2444
                                 0D56  2445 60$:      BUG_CHECK       NOBUFPCKT,FATAL
                                 0D5A  2446
```

```
                     0D5A  2448          .SBTTL  READ_MODEM - Read device modem register
                     0D5A  2449 ;++
                     0D5A  2450 ;READ_MODEM - Read device modem register
                     0D5A  2451 ;
                     0D5A  2452 ; This routine returns the device modem register in the buffer passed.
                     0D5A  2453 ; These are set according to the XMDEF's for modem bits.
                     0D5A  2454 ;
                     0D5A  2455 ;         INPUTS:
                     0D5A  2456 ;                 R3 = IRP address
                     0D5A  2457 ;                 R5 = UCB address
                     0D5A  2458 ;
                     0D5A  2459 ;--
                     0D5A  2460 READ_MODEM:
                     0D5A  2461
                     0D5A  2462 ;& WHAT DO WE DO HERE??????
                     0D5A  2463
06 68 A5    06   E0  0D5A  2464          BBS     #NO_DS_V_INITED,UCB$W_DEVSTS(R5),5$ ; If BS then read register
 50   20D4 8F   3C  0D5F  2465          MOVZWL  #SS$_DEVINACT,R0                    ; Else return device inactive
                05  0D64  2466          RSB
                     0D65  2467
      50   01   3C  0D65  2468 5$:      MOVZWL  S^#SS$_NORMAL,R0
                05  0D68  2469          RSB
```

```
                     0D69  2471          .SBTTL  FILL DUETIME_TABLE - Set up the due time table
                     0D69  2472 ;++
                     0D69  2473 ; FILL_DUETIME_TABLE
                     0D69  2474 ;
                     0D69  2475 ; Functional description:
                     0D69  2476 ;
                     0D69  2477 ; This routine sets up the default values in the table as well as sets up the
                     0D69  2478 ; known terminal line speeds based on the default buffer size due times.
                     0D69  2479 ; The due time is the max length of time we believe it should take to
                     0D69  2480 ; transmit of receive a given message based on the speed of the line
                     0D69  2481 ;
                     0D69  2482 ;       INPUTS:
                     0D69  2483 ;               R5 = UCB address
                     0D69  2484 ;
                     0D69  2485 ;       OUTPUTS:
                     0D69  2486 ;               The DUETIM_TABLE is filled.
                     0D69  2487 ;
                     0D69  2488 ;--
                     0D69  2489 FILL_DUETIM_TABLE:
              07 BB  0D69  2490          PUSHR   #^M<R0,R1,R2>
                     0D6B  2491
                     0D6B  2492 ; Set up table with defaults. This is in case a terminal line supports a
                     0D6B  2493 ; speed that we don't know about. We wish to make this reasonable so we will
                     0D6B  2494 ; wait at least one second.
                     0D6B  2495
        50 0100 8F   3C  0D6B  2496          MOVZWL  #NO$C_DUETIM_TABLE_SIZE,R0
              51 D4  0D70  2497          CLRL    R1
    F38C CF41 01 B0  0D72  2498 10$:     MOVW    #1,DUETIM_TABLE[R1]
              51 96  0D78  2499          INCB    R1
           F5 50 F5  0D7A  2500          SOBGTR  R0,10$
50    42 A5 0C A1  0D7D  2501          ADDW3   #12,UCB$W_DEVBUFSIZ(R5),R0              ; Get the max msg size
        50 08 A4  0D82  2502          MULW2   #8,R0                                   ; mult by 8 bits per byte
                  0D85  2503          STORE_DUETIM    50,R0,R1,R2,DUETIM_TABLE
                  0D95  2504          STORE_DUETIM    75,R0,R1,R2,DUETIM_TABLE
                  0DA7  2505          STORE_DUETIM    110,R0,R1,R2,DUETIM_TABLE
                  0DB9  2506          STORE_DUETIM    134,R0,R1,R2,DUETIM_TABLE
                  0DCB  2507          STORE_DUETIM    150,R0,R1,R2,DUETIM_TABLE
                  0DDD  2508          STORE_DUETIM    300,R0,R1,R2,DUETIM_TABLE
                  0DEF  2509          STORE_DUETIM    600,R0,R1,R2,DUETIM_TABLE
                  0E01  2510          STORE_DUETIM    1200,R0,R1,R2,DUETIM_TABLE
                  0E13  2511          STORE_DUETIM    1800,R0,R1,R2,DUETIM_TABLE
                  0E25  2512          STORE_DUETIM    2000,R0,R1,R2,DUETIM_TABLE
                  0E37  2513          STORE_DUETIM    2400,R0,R1,R2,DUETIM_TABLE
                  0E49  2514          STORE_DUETIM    3600,R0,R1,R2,DUETIM_TABLE
                  0E5B  2515          STORE_DUETIM    4800,R0,R1,R2,DUETIM_TABLE
                  0E6D  2516          STORE_DUETIM    7200,R0,R1,R2,DUETIM_TABLE
                  0E7F  2517          STORE_DUETIM    9600,R0,R1,R2,DUETIM_TABLE
                  0E91  2518          STORE_DUETIM    19200,R0,R1,R2,DUETIM_TABLE
              07 BA  0EA3  2519          POPR    #^M<R0,R1,R2>
                 05  0EA5  2520          RSB
```

```
                                0EA6  2522          .SBTTL  START - Start unit, device and/or protocol
                                0EA6  2523 ;++
                                0EA6  2524 ; START_LINE - Start unit
                                0EA6  2525 ;
                                0EA6  2526 ; Functional description:
                                0EA6  2527 ;
                                0EA6  2528 ; This routine initializes the UCB; allocates the map registers for
                                0EA6  2529 ; transmits and receives; and master clears the device.
                                0EA6  2530 ;
                                0EA6  2531 ; If a failure occurs the line shutdown sequence is entered.
                                0EA6  2532 ;
                                0EA6  2533 ;       INPUTS:
                                0EA6  2534 ;               R3 = I/O packet
                                0EA6  2535 ;               R5 = UCB address
                                0EA6  2536 ;
                                0EA6  2537 ;       IMPLICIT INPUTS:
                                0EA6  2538 ;
                                0EA6  2539 ;               IRP$L_MEDIA contains a copy of the mode buffer specified
                                0EA6  2540 ;                       by the user.
                                0EA6  2541 ;               IRP$W_QUOTA contains the quota taken from the user for the unit.
                                0EA6  2542 ;
                                0EA6  2543 ;       OUTPUTS:
                                0EA6  2544 ;               The request is completed.
                                0EA6  2545 ;
                                0EA6  2546 ;--
                                0EA6  2547 START_LINE:                                  ; Start protocol operation
                          06    E1  0EA6  2548          BBC     #NO_DS_V_INITED,-       ; If BC then device not inited
                    06 68 A5        0EA8  2549                  UCB$W_DEVSTS(R5),5$     ; Branch to set up idle UCB
                50   02C4 8F    3C  0EAB  2550          MOVZWL  #SS$_DEVACTIVE,R0       ; Set device active
                                05  0EB0  2551          RSB                             ; ... and return
                                    0EB1  2552
                54   00A8 C5    D0  0EB1  2553 5$:      MOVL    UCB$L_NO_BUFFER(R5),R4  ; Get NOB address
            44 A5    FFF3 300 8F    CA  0EB6  2554          BICL    #NO$C_LINE_PAR,UCB$L_DEVDEPEND(R5) ; Reset all Read/Write flags
                40 A4    40 A3    B0  0EBE  2555          MOVW    IRP$W_QUOTA(R3),NOB$W_QUOTA(R4) ; Set quota for RCV's
                         40 A3    B4  0EC3  2556          CLRW    IRP$W_QUOTA(R3)         ; Quota is return is handled by us
                                     0EC6  2557
                58 A4    0C A3    D0  0EC6  2558          MOVL    IRP$L_PID(R3),NOB$L_PID(R4) ; Save starter's PID
                5C A4    28 A3    B0  0ECB  2559          MOVW    IRP$W_CHAN(R3),NOB$W_CHANL(R4) ; Save channel number
                            FE96  30  0ED0  2560          BSBW    FILL_DUETIM_TABLE       ; Fill in xmt/rcv message due times
             68 A5    0040 8F    A8  0ED3  2561          BISW    #NO_DS_M_INITED,UCB$W_DEVSTS(R5) ; Indicate UCB initialized
                                     0ED9  2562
                                     0ED9  2563 ; Call the terminal port driver to set line speed and parity. This call
                                     0ED9  2564 ; may affect the UCB$L_DEVDEPEND field, thus we must save this field
                                     0ED9  2565 ; so that information does not get lost.
                                     0ED9  2566
                         44 A5    DD  0ED9  2567          PUSHL   UCB$L_DEVDEPEND(R5)
                52   0118 C5    D0  0EDC  2568          MOVL    UCB$L_TT_PORT(R5),R2    ; Get port vector table
                                     0EE1  2569          DSBINT  UCB$B_DIPL(R5)
                00F8 C5    18  90  0EE8  2570          MOVB    #NO$C_SET_LINE,UCB$B_TT_PARITY(R5) ; We must always send 8 bits of d
                                     0EED  2571
                                     0EED  2572 ; Make sure that the device does not do XON/XOFF. If it should be doing
                                     0EED  2573 ; this, you can notice this because transmits appear to just stop
                                     0EED  2574 ; on the line.
                                     0EED  2575
                                     0EED  2576          CLRBIT  #TTY$V_PC_XOFENA,UCB$W_TT_PRTCTL(R5)
                                     0EF3  2577
                                     0EF3  2578 ; Set the port up so that it takes advantage of the DMA capabilities
```

```
                                0EF3  2579 ; of the device. Devices like the DMF32 do DMA output.
                                0EF3  2580
                     0B   A8    0EF3  2581          BISW2   #TTY$M_PC_DMAENA!TTY$M_PC_PRMMAP!TTY$M_PC_NOTIME,-
                0122 C5         0EF5  2582                  UCB$W_TT_PRTCTL(R5)
                     54   DD    0EF8  2583          PUSHL   R4
                  08 B2   16    0EFA  2584          JSB     @PORT_SET_LINE(R2)              ; Jump to tell the port
                                0EFD  2585
                                0EFD  2586 ; Now call the port to clear the line.
                                0EFD  2587
                  20 B2   16    0EFD  2588          JSB     @PORT_ABORT(R2)
                  24 B2   16    0F00  2589          JSB     @PORT_RESUME(R2)
                     54 8ED0    0F03  2590          POPL    R4
                                0F06  2591          ENBINT
                  44 A5 8ED0    0F09  2592          POPL    UCB$L_DEVDEPEND(R5)             ; Restore the devdepend field
                                0F0D  2593
                                0F0D  2594
                                0F0D  2595
                                0F0D  2596 ; Set up DDCMP defaults and other information necessary for the protocol
                                0F0D  2597
                03C0 8F   BB    0F0D  2598          PUSHR   #^M<R6,R7,R8,R9>
           50    0C A4   D0    0F11  2599          MOVL    NOB$A_PRO_BUFFER(R4),R0         ; Get protocol buffer
01D1 C0    0B A5   90    0F15  2600          MOVB    UCB$B_FIPL(R5),GF$B_FIPL+TF$K_LENGTH(R0) ; Set fork IPL for DDCMP
01D2 C0    5E A5   90    0F1B  2601          MOVB    UCB$B_DIPL(R5),GF$B_DIPL+TF$K_LENGTH(R0) ; Set device IPL for DDCMP
           01C8 C0   94    0F21  2602          CLRB    GF$B_STATE+TF$K_LENGTH(R0)      ; Make sure state is halted
4C A0      01C8 C0   DE    0F25  2603          MOVAL   TF$K_LENGTH(R0),TF$A_GFB(R0)    ; Set address of global field
                38   BB    0F2B  2604          PUSHR   #^M<R3,R4,R5>
           58   60 A4   9E    0F2D  2605          MOVAB   NOB$Z_DDCMP(R4),R8              ; Set addr of buf with param
           59   78 A4   9E    0F31  2606          MOVAB   NOB$Z_DLA_ADDR(R4),R9           ; Set addr of buffer for addresses
           51   34 A4   9E    0F35  2607          MOVAB   NOB$L_XMT_INPR(R4),R1           ; Set up addresses
                69   51   D0    0F39  2608          MOVL    R1,DLA$A_XMT_INPR(R9)
           51   28 A4   9E    0F3C  2609          MOVAB   NOB$Q_POST(R4),R1
            04 A9   51   D0    0F40  2610          MOVL    R1,DLA$A_POSTQ(R9)
                55   50   D0    0F44  2611          MOVL    R0,R5                           ; Get adar of start of TFB
                56   05   9A    0F47  2612          MOVZBL  #DLK$C_CHAR,R6                  ; Set DDCMP default char
                     57   D4    0F4A  2613          CLRL    R7                              ; Zero the register
      57    09F0 8F   A8    0F4C  2614          BISW    #<DLK$M_MSGCNT!-                ; indicate chara to set
                                0F51  2615                  DLK$M_SELTIM!-
                                0F51  2616                  DLK$M_SELWAIT!-
                                0F51  2617                  DLK$M_REPTIM!-
                                0F51  2618                  DLK$M_REPWAIT!-
                                0F51  2619                  DLK$M_SETDEF>,R7
                   FOAC'  30    0F51  2620          BSBW    DDCMP
                     38   BA    0F54  2621          POPR    #^M<R3,R4,R5>
                                0F56  2622
                                0F56  2623 ; Before starting up the DDCMP timer make sure that the driver can not
                                0F56  2624 ; be reloaded or unloaded while the timer is running.
                                0F56  2625
0000000D'EF         04   88    0F56  2626          BISB    #DPT$M_NOUNLOAD,DPT$TAB+DPT$B_FLAGS
                                0F5D  2627
                                0F5D  2628 ; Start up the DDCMP timer. First set up registers with params for call
                                0F5D  2629 ; then make the call to the protocol.
                                0F5D  2630
                     38   BB    0F5D  2631          PUSHR   #^M<R3,R4,R5>
                56   06   9A    0F5F  2632          MOVZBL  #DLK$C_START_TIMER,R6           ; Set up R6 with the DDCMP comm
      57    164C'CF   DE    0F62  2633          MOVAL   W^DEVTIMER,R7                   ; R7 must have device timer addr
                58   55   D0    0F67  2634          MOVL    R5,R8                           ; R8 must have UCB addr
                     59   D4    0F6A  2635          CLRL    R9                              ; Clear R9
```

```
                                                                I  1
55   0C A4    DO  0F6C  2636          MOVL    NOB$A_PRO_BUFFER(R4),R5         ; Get addr of start of TFB
      F08D'   30  0F70  2637          BSBW    DDCMP                           ; Call protocol
        38    BA  0F73  2638          POPR    #^M<R3,R4,R5>
   03C0 8F    BA  0F75  2639          POPR    #^M<R6,R7,R8,R9>
   50   01    3C  0F79  2640          MOVZWL  S^#SS$_NORMAL,R0                ; Set success
             05  0F7C  2641          RSB
                 0F7D  2642
                 0F7D  2643 START_CTRL_ERROR:
        50   DD  0F7D  2644          PUSHL   R0                              ; Save abort reason
      08CC   30  0F7F  2645          BSBW    SHUTDOWN_LINE                   ; Shut down the device
        50 8ED0  0F82  2646          POPL    R0                              ; Restore abort reason
             05  0F85  2647          RSB
                 0F86  2648
```

```
                        0F86  2650
                        0F86  2651 ;++
                        0F86  2652 ; START_CIRCUIT- Start device and protocol
                        0F86  2653 ;
                        0F86  2654 ; Functional description:
                        0F86  2655 ;
                        0F86  2656 ; This routine sets up the protocol and device charateristics; sets up receive
                        0F86  2657 ; buffers for the device; starts the protocol; and finally gives the first
                        0F86  2658 ; transmit and receives to the board.
                        0F86  2659 ;
                        0F86  2660 ; If a failure occurs the circuit shutdown sequence is entered.
                        0F86  2661 ;
                        0F86  2662 ;       INPUTS:
                        0F86  2663 ;               R3 = I/O packet
                        0F86  2664 ;               R5 = UCB address
                        0F86  2665 ;
                        0F86  2666 ;       IMPLICIT INPUTS:
                        0F86  2667 ;
                        0F86  2668 ;               IRP$L_MEDIA contains a copy of the mode buffer specified
                        0F86  2669 ;                       by the user.
                        0F86  2670 ;
                        0F86  2671 ;        OUTPUTS:
                        0F86  2672 ;               The request is completed.
                        0F86  2673 ;
                        0F86  2674 ;--
                        0F86  2675 START_CIRCUIT:                                       ; Start the circuit
          0B    E1      0F86  2676         BBC     #XM$V_STS_ACTIVE,-                   ; If BC then device and
 06 44 A5               0F88  2677                 UCB$L_DEVDEPEND(R5),5$               ; protocol are not active
50  02C4 8F    3C      0F8B  2678         MOVZWL  #SS$_DEVACTIVE,R0                    ; Return error
               05      0F90  2679         RSB
                       0F91  2680
     54  00A8 C5   D0  0F91  2681 5$:     MOVL    UCB$L_NO_BUFFER(R5),R4               ; Get NOB address
               06      E0  0F96  2682         BBS     #NO_DS_V_INITED,-                    ; If BS then line started
         06 68 A5              0F98  2683                 UCB$W_DEVSTS(R5),8$
     50  20D4 8F   3C  0F9B  2684         MOVZWL  #SS$_DEVINACT,R0                     ; Else return device inactive
               05      0FA0  2685         RSB
                       0FA1  2686
                       0FA1  2687 ; Reset bit fields
                       0FA1  2688
          03C0 8F    BB  0FA1  2689 8$:     PUSHR   #^M<R6,R7,R8,R9>
   68 A5    FFBF 8F    AA  0FA5  2690         BICW    #^C<NO_DS_M_INITED>,UCB$W_DEVSTS(R5)
44 A5   FFFFF700 8F    CA  0FAB  2691         BICL    #NO$C_CIR_PAR,UCB$L_DEVDEPEND(R5) ; Reset all Read/Write flags
             0075      30  0FB3  2692         BSBW    SET_CHAR                             ; Set protocol char
                       0FB6  2693         DSBINT  #IPL$_POWER                          ; Interlock powerfail
    05 64 A5    05    E1  0FBC  2694         BBC     #UCB$V_POWER,UCB$W_STS(R5),10$       ; BR if no power failed
                       0FC1  2695         ENBINT                                       ; Enable intterrupts at FIPL
             4B    11  0FC4  2696         BRB     START_POWERFAIL                      ; Branch to report powerfail
          0800 8F    A8  0FC6  2697 10$:    BISW    #XM$M_STS_ACTIVE,-
            44 A5          0FCA  2698                 UCB$L_DEVDEPEND(R5)                  ; Set active state
                       0FCC  2699         ENBINT                                       ; Return to fork IPL
          56    04    9A  0FCF  2700         MOVZBL  #DLK$C_USRINT,R6                     ; Set to start the protocol
                00    E1  0FD2  2701         BBC     #XM$V_CHR_MOP,-                      ; If BC then don't enter
         06 44 A5          0FD4  2702                 UCB$L_DEVDEPEND(R5),15$              ; maint mode
          57    04    9A  0FD7  2703         MOVZBL  #DLK$M_MAINT,R7                      ; Set to start maint mode
             0003      31  0FDA  2704         BRW     20$
          57    01    9A  0FDD  2705 15$:    MOVZBL  #DLK$M_START,R7                      ; Else simply start the prot
                38    BB  0FE0  2706 20$:    PUSHR   #^M<R3,R4,R5>                        ; Save registers
```

```
                55   0C A4    D0  0FE2  2707          MOVL    NOB$A_PRO_BUFFER(R4),R5         ; Get addr of start of TFB
                       58     7C  0FE6  2708          CLRQ    R8
                     F015'    30  0FE8  2709          BSBW    DDCMP                           ; Jump to the protocol
                       38     BA  0FEB  2710          POPR    #^M<R3,R4,R5>                   ; Restore registers
                   56   09     91  0FED  2711          CMPB    #DLK$C_ACTNOTCOM,R6             ; If EQL then protocol not
                       13     13  0FF0  2712          BEQL    25$                             ; in halted state don't reinit
                    007B      30  0FF2  2713          BSBW    FILLFREELIST                    ; Fill rcv free buffer Q
                  1E 50       E9  0FF5  2714          BLBC    R0,START_CIRCUIT_ERROR          ; Branch to shutdown the device
                       51     D4  0FF8  2715          CLRL    R1                              ; Set no status for start transmit
                     FC4A     30  0FFA  2716          BSBW    START_TRANSMIT                  ; Branch to send STRT message
                  16 50       E9  0FFD  2717          BLBC    R0,START_CIRCUIT_ERROR          ; Branch to shutdown the device
                   50   01     3C  1000  2718          MOVZWL  S^#SS$_NORMAL,R0                ; Set normal return
                       19     11  1003  2719          BRB     START_CIRCUIT_COMP              ; Comp the request
                              1005  2720
            50   02C4 8F     3C  1005  2721 25$:     MOVZWL  #SS$_DEVACTIVE,R0               ; Set reason for abort
                              100A  2722              SETBIT  #XMS$V_ERR_TRIB,-               ; Set to restart ciruit
                              100A  2725                      UCB$L_DEVDEPEND(R5)
                       05     11  100F  2724          BRB     START_CIRCUIT_ERROR
                              1011  2725 START_POWERFAIL:
            50   0364 8F     3C  1011  2726          MOVZWL  #SS$_POWERFAIL,R0               ; Set reason to abort
                              1016  2727 START_CIRCUIT_ERROR:
                       50     DD  1016  2728          PUSHL   R0                              ; Save reason to abort
                     08C5     30  1018  2729          BSBW    SHUTDOWN_CIRCUIT                ; Shutdown the circuit
                       50 8ED0   101B  2730          POPL    R0                              ; Restore reason for abort
                              101E  2731 START_CIRCUIT_COMP:
                53   58 A5     D0  101E  2732          MOVL    UCB$L_IRP(R5),R3                ; Retreive packet address
                7C A5    50     B0  1022  2733          MOVW    R0,UCB$W_BOFF(R5)               ; Set success
                    03C0 8F     BA  1026  2734          POPR    #^M<R6,R7,R8,R9>
                              05  102A  2735          RSB                                     ; Jump to complete the request
                              102B  2736
                              102B  2737
```

```
                                102B  2739          .SBTTL  Set protocol characteristic
                                102B  2740 ;++
                                102B  2741 ; SET_CHAR - Set characteristics of protocol
                                102B  2742 ;
                                102B  2743 ; FUNCTIONAL DESCRIPTION:
                                102B  2744 ;
                                102B  2745 ; This routine is entered to set characteristic on a halted protocol.
                                102B  2746 ;
                                102B  2747 ;       INPUTS:
                                102B  2748 ;               R4 = NOB address
                                102B  2749 ;               R5 = UCB address
                                102B  2750 ;
                                102B  2751 ;
                                102B  2752 ;       OUTPUTS:
                                102B  2753 ;               New protocol characteristics given to protocol.
                                102B  2754 ;
                                102B  2755 ;--
                                102B  2756 SET_CHAR:
                          57    D4  102B  2757          CLRL    R7                              ; Set no charac to change
                    58    60 A4   9E  102D  2758          MOVAB   NOB$Z_DDCMP(R4),R8              ; Get trib parameter block
                       01 A8    90  1031  2759          MOVB    DLK$B_MSGCNT(R8),-              ; Use this field for RTO's
                       02 A8        1034  2760                  DLK$B_MAXREP(R8)
                       01 A8    90  1036  2761          MOVB    DLK$B_MSGCNT(R8),-              ; And for number of Select
                       03 A8        1039  2762                  DLK$B_MAXSEL(R8)                ; intervals
                       04 A8    B0  103B  2763          MOVW    DLK$W_REPWAIT(R8),-             ; Use repwait for selwait
                       06 A8        103E  2764                  DLK$W_SELWAIT(R8)
                57  01F8 8F    A8  1040  2765          BISW    #<DLK$M_MSGCNT!-                ; Set to reset these param
                                1045  2766                  DLK$M_SELTIM!-
                                1045  2767                  DLK$M_SELWAIT!-
                                1045  2768                  DLK$M_REPTIM!-
                                1045  2769                  DLK$M_REPWAIT!-
                                1045  2770                  DLK$M_LNTYP>,R7
                                1045  2771
                                1045  2772          ASSUME  NMA$C_LINPR_POI EQ 0
                                1045  2773          SETBIT  #DLK$V_SETDEF,R7                ; Set default char
                       70 A4    95  1049  2774          TSTB    NOB$B_PRO(R4)                   ; If EQL then point to point
                          08    13  104C  2775          BEQL    10$
                                104E  2776          CLRBIT  #DLK$V_SETDEF,R7                ; Use user given char
                                1052  2777          SETBIT  #DLK$V_STATYP,R7                ; Else set trib
                                1056  2778 10$:      ASSUME  NMA$C_DPX_FUL EQ 0
                       71 A4    95  1056  2779          TSTB    NOB$B_DUP(R4)                   ; If EQL the full duplex
                          04    13  1059  2780          BEQL    20$
                                105B  2781          SETBIT  #DLK$V_DUPLEX,R7                ; Else set half duplex
                          38    BB  105F  2782 20$:     PUSHR   #^M<R3,R4,R5>                   ; Save registers
                55    0C A4    D0  1061  2783          MOVL    NOB$A_PRO_BUFFER(R4),R5         ; Get addr of start of TFB
                   56     05    9A  1065  2784          MOVZBL  #DLK$C_CHAR,R6                  ; Set charac in DDCMP
                          59    D4  1068  2785          CLRL    R9
                        EF93'   30  106A  2786          BSBW    DDCMP
                          38    BA  106D  2787          POPR    #^M<R3,R4,R5>                   ; Restore the registers
                          05    05  106F  2788 30$:     RSB
                                1070  2789
                                1070  2790
```

```
                     1070  2792          .SBTTL  FILLFREELIST - FILL MESSAGE FREE LIST
                     1070  2793 ;++
                     1070  2794 ; FILLFREELIST - Fill message block free list
                     1070  2795 ; ADDFREELIST - Add a buffer to receive list
                     1070  2796 ;
                     1070  2797 ; Functional description:
                     1070  2798 ;
                     1070  2799 ; This routine fills the receive buffer free list up to the quota specified
                     1070  2800 ; at device startup.
                     1070  2801 ;
                     1070  2802 ;       Inputs:
                     1070  2803 ;
                     1070  2804 ;               R2 = Buffer address (ADDFREELIST only)
                     1070  2805 ;               R4 = NOB address
                     1070  2806 ;               R5 = UCB address
                     1070  2807 ;
                     1070  2808 ;       IPL = FIPL
                     1070  2809 ;
                     1070  2810 ;       Outputs:
                     1070  2811 ;               R3,R4,R5 are preserved
                     1070  2812 ;               R1,R2 destroyed.
                     1070  2813 ;--
                     1070  2814 FILLFREELIST:                                    ; Fill free list
           52   D4  1070  2815          CLRL    R2                              ; Clear buffer address
           0B   E0  1072  2816          BBS     #XM$V_STS_ACTIVE,-              ; Continue if device active
 01 44 A5           1074  2817                  UCB$L_DEVDEPEND(R5),ADDFREELIST
           05       1077  2818          RSB
                    1078  2819 ADDFREELIST:                                    ; Add to receive buffer list
           53   DD  1078  2820          PUSHL   R3                              ; Save registers
       42  A5   B1  107A  2821 5$:      CMPW    UCB$W_DEVBUFSIZ(R5),-           ; Can new block be allocated ?
       40  A4       107D  2822                  NOB$W_QUOTA(R4)
           35   1A  107F  2823          BGTRU   20$                             ; Br if no - list filled
           51   D4  1081  2824          CLRL    R1                              ; Zero size
     004C  8F   A1  1083  2825          ADDW3   #RCV_T_DATA+CXB$C_TRAILER,-     ; Compute needed block size
 51    42  A5       1087  2826                  UCB$W_DEVBUFSIZ(R5),R1
           52   D5  108A  2827          TSTL    R2                              ; Buffer allocated already?
           09   12  108C  2828          BNEQ    7$                              ; Br if yes
00000000'GF   16  108E  2829          JSB     G^EXE$ALUNONPAGED               ; Allocate nonpaged memory
       18  50   E9  1094  2830          BLBC    R0,10$                          ; Br if failure
 08 A2     51   B0  1097  2831 7$:      MOVW    R1,RCV_W_BLKSIZE(R2)            ; Insert block size
 0A A2     17   90  109B  2832          MOVB    S^#DYN$C_NET,RCV_B_BLKTYPE(R2)  ; Insert block type
       10  A2   B4  109F  2833          CLRW    RCV_W_ERROR(R2)                 ; Clear error status
 20 A4     62   0E  10A2  2834          INSQUE  (R2),NOB$Q_FREE(R4)             ; Insert block on list
       42  A5   A2  10A6  2835          SUBW    UCB$W_DEVBUFSIZ(R5),-           ; Decrement quota
       40  A4       10A9  2836                  NOB$W_QUOTA(R4)
           52   D4  10AB  2837          CLRL    R2                              ; Clear buffer pointer
           CB   11  10AD  2838          BRB     5$
                    10AF  2839
                    10AF  2840 10$:     SETBIT  #XM$V_STS_BUFFAIL,-             ; Set buffer alloc failure
                    10AF  2841                  UCB$L_DEVDEPEND(R5)
           10   11  10B4  2842          BRB     30$
                    10C36 2843
                    10B6  2844 20$:     CLRBIT  #XM$V_STS_BUFFAIL,-             ; Clear buffer alloc failure
                    10B6  2845                  UCB$L_DEVDEPEND(R5)
     50    52   D0  10BB  2846          MOVL    R2,R0                           ; Set address of buffer
           06   13  10BE  2847          BEQL    30$                             ; Br if none
00000000'GF   16  10C0  2848          JSB     G^COM$DRVDEALMEM                ; Deallocate it
```

```
                50    01    3C  10C6  2849 30$:    MOVZWL  S^#SS$_NORMAL,R0            ; Set for normal return
                      53 8ED0  10C9  2850         POPL    R3                          ; Restore registers
                         05  10CC  2851         RSB
                             10CD  2852
```

```
                     10CD  2854          .SBTTL  START_RECEIVE and GET RECEIVE - Start receive and Get a receive buff
                     10CD  2855 ;++
                     10CD  2856 ; START_RECEIVE - Start receives
                     10CD  2857 ;
                     10CD  2858 ; FUNCTIONAL DESCRIPTION:
                     10CD  2859 ;
                     10CD  2860 ; This routine sets up the driver to start receive the header. The header is
                     10CD  2861 ; receive into some fields of the NOB so that even if no buffer has been
                     10CD  2862 ; allocated or can be allocated we can still get the header and the byte
                     10CD  2863 ; count of the message.
                     10CD  2864 ;
                     10CD  2865 ;       INPUTS:
                     10CD  2866 ;               R4 = NOB address
                     10CD  2867 ;               R5 = UCB address
                     10CD  2868 ;               IPL = Device IPL
                     10CD  2869 ;
                     10CD  2870 ;       OUTPUTS:
                     10CD  2871 ;               R4, R5 are preserved
                     10CD  2872 ;
                     10CD  2873 ;--
                     10CD  2874 START_RECEIVE:
          43 A4  94  10CD  2875          CLRB    NOB$B_RSTATE(R4)                ; Peset receiver state
                     10D0  2876
                     10D0  2877          ASSUME  NOB$W_FLAGS EQ NOB$W_ERROR+2
                     10D0  2878
          52 A4  B4  10D0  2879          CLRW    NOB$W_INDEX(R4)                 ; Clear index into rcv buffer
          54 A4  D4  10D3  2880          CLRL    NOB$W_ERROR(R4)                 ; Reset flag and error fields
                     10D6  2881
                     10D6  2882 ; Set up message count with size of header and crc
                     10D6  2883
    50 A4    08  9B  10D6  2884          MOVZBW  #NO$C_HEADER_HCRC,NOB$W_MSGSIZ(R4)
                 05  10DA  2885          RSB
                     10DB  2886
                     10DB  2887
                     10DB  2888 ;++
                     10DB  2889 ; GET_RECEIVE - Get a receive buffer if one is available
                     10DB  2890 ;
                     10DB  2891 ; FUNCTIONAL DESCRIPTION:
                     10DB  2892 ;
                     10DB  2893 ; This routine sets up the driver to receive the data. It gets the next
                     10DB  2894 ; free buffer from the queue. And sets up the fields in the
                     10DB  2895 ; buffer to start receive the data. This buffer is posted
                     10DB  2896 ; when the driver has finished receiveing the data for it.
                     10DB  2897 ;
                     10DB  2898 ;       INPUTS:
                     10DB  2899 ;               R4 = NOB address
                     10DB  2900 ;               R5 = UCB address
                     10DB  2901 ;               IPL = Device IPL
                     10DB  2902 ;
                     10DB  2903 ;       OUTPUTS:
                     10DB  2904 ;               R0 = LBC no buffer available
                     10DB  2905 ;                    LBS buffer is available and returned in R2
                     10DB  2906 ;               R2 = Address of buffer to receive into if a buffer was available
                     10DB  2907 ;               R4, R5 are preserved
                     10DB  2908 ;
                     10DB  2909 ;--
                     10DB  2910 GET_RECEIVE:
```

```
                50    D4  10DB  2911          CLRL    R0                                  ; Assume no buffer
                                  10DD  2912
                                  10DD  2913 ; If line is in process of being shutdown do not try to get a receive buffer
                                  10DD  2914
          17 68 A5    0A    E0  10DD  2915          BBS     #NO_DEV_V_SHUTDOWN,UCB$W_DEVSTS(R5),10$
             52    20 B4    0F  10E2  2916          REMQUE  @NOB$Q_FREE(R4),R2                  ; Get next available buffer
                      11    1D  10E6  2917          BVS     10$                                 ; If VS none available
             30 A4    52    D0  10E8  2918          MOVL    R2,NOB$L_RCV_INPR(R4)               ; Set receive buffer address
                                  10EC  2919
                                  10EC  2920          ASSUME  NOB$W_INDEX EQ NOB$W_MSGSIZ+2
                                  10EC  2921          ASSUME  NOB$W_ERROR EQ NOB$W_INDEX+2
                                  10EC  2922          ASSUME  NOB$W_FLAGS EQ NOB$W_ERROR+2
                                  10EC  2923
                                  10EC  2924          ASSUME  RCV_W_INDEX EQ RCV_W_MSGSIZ+2
                                  10EC  2925          ASSUME  RCV_W_ERROR EQ RCV_W_INDEX+2
                                  10EC  2926          ASSUME  RCV_W_FLAGS EQ RCV_W_ERROR+2
                                  10EC  2927
          0C A2    50 A4    7D  10EC  2928          MOVQ    NOB$W_MSGSIZ(R4),RCV_W_MSGSIZ(R2)
                                  10F1  2929
                                  10F1  2930          ASSUME NO$C_HEADER_HCRC EQ 8
                                  10F1  2931
          40 A2    48 A4    7D  10F1  2932          MOVQ    NOB$Z_HEADER(R4),RCV_Z_HEADER(R2)
                50    01    3C  10F6  2933          MOVZWL  S^#SS$_NORMAL,R0                    ; Set normal return
                            05  10F9  2934 10$:      RSB
                                  10FA  2935
```

```
                     10FA  2937          .SBTTL  NO$GETNXT - Class driver GETNXT routine
                     10FA  2938 ;++
                     10FA  2939 ;  NO$GETNXT - Class driver GETNXT routine entry point for PORT driver
                     10FA  2940 ;
                     10FA  2941 ; FUNCTIONAL DESCRIPTION
                     10FA  2942 ;
                     10FA  2943 ; This is the ASYNCH DDCMP transmit "interrupt" service routine. This routine
                     10FA  2944 ; will be called whenever the terminal port driver has completed the current
                     10FA  2945 ; character or burst of characters given to it by the ASYNCH DDCMP class driver.
                     10FA  2946 ; When called this routine will calculate the next character or burst of
                     10FA  2947 ; characters to transmit over the port driver. If data is returned by
                     10FA  2948 ; GETNXT then a timer is set in the port driver and the bit UCB$V_INT
                     10FA  2949 ; must be set in UCB$W_STS to say that the port is still busy. Remeber that
                     10FA  2950 ; messages are tranmitted over the port driver in chunks of data (See
                     10FA  2951 ; START_TRANSMIT to get break down).
                     10FA  2952 ;
                     10FA  2953 ; Note that UCB$B_TT_OUTYPE is always set to negative (unless there is
                     10FA  2954 ; no data to send), becasue we always send data in bursts.
                     10FA  2955 ;
                     10FA  2956 ;       INPUTS:
                     10FA  2957 ;               R5 = UCB address
                     10FA  2958 ;
                     10FA  2959 ;       IPL = DIPL
                     10FA  2960 ;
                     10FA  2961 ;       OUTPUTS:
                     10FA  2962 ;               R3 = character to output
                     10FA  2963 ;               R5 = UCB address
                     10FA  2964 ;
                     10FA  2965 ;       IMPLICIT OUTPUTS:
                     10FA  2966 ;               UCB$B_TT_OUTYPE = Zero - no data to output
                     10FA  2967 ;                                 One - single character in R3
                     10FA  2968 ;                                 Negative - output a burst of data
                     10FA  2969 ;               Burst mode only:
                     10FA  2970 ;               UCB$L_TT_OUTADR = Address of the first character of the burst
                     10FA  2971 ;               UCB$W_TT_OUTLEN = Length of the burst
                     10FA  2972 ;
                     10FA  2973 ;               R0 preserved, All others destroyed
                     10FA  2974 ;
                     10FA  2975 ;--
                     10FA  2976 NO$GETNXT:
         03D1 8F     BB  10FA  2977          PUSHR   #^M<R0,R4,R6,R7,R8,R9>
54       00A8 C5     D0  10FE  2978          MOVL    UCB$L_NO_BUFFER(R5),R4          ; Get NOB address
                57   13  1103  2979          BEQL    20$
  58       34 A4     D0  1105  2980          MOVL    NOB$L_XMT_INPR(R4),R8           ; Get transmit address
                63   13  1109  2981          BEQL    30$                             ; If EQL no buffer to transmit
                     110B  2982          SETBIT  #UCB$V_INT,UCB$W_STS(R5)        ; Assume we always have data to send
                     1110  2983
                     1110  2984 ; Increment the transmiter state to say that we should send the next block
                     1110  2985 ; of the transmit. If we increment past the sending of pads then that
                     1110  2986 ; means we are thru with this data block and we should post this
                     1110  2987 ; data message for complete and start the next message transmitting.
                     1110  2988          ASSUME NO$C_HEADER EQ NO$C_EMPTY+1
                     1110  2989          ASSUME NO$C_HCRC EQ NO$C_HEADER+1
                     1110  2990          ASSUME NO$C_DATA EQ NO$C_HCRC+1
                     1110  2991          ASSUME NO$C_DCRC EQ NO$C_DATA+1
                     1110  2992          ASSUME NO$C_PADS EQ NO$C_DCRC+1
                     1110  2993
```

```
                42 A4    0A   91  1110  2994 10$:    CMPB    #NO$C_XMTERR,NOB$B_XSTATE(R4)   ; If NEQ normal xmt states
                         03   12  1114  2995         BNEQ    12$
                      013C    31  1116  2996         BRW     SEND_XMTERR                     ; Else handle the xmt error
                   42 A4      96  1119  2997 12$:    INCB    NOB$B_XSTATE(R4)
                50 42 A4      9A  111C  2998         MOVZBL  NOB$B_XSTATE(R4),R0             ; Get state for dispatch
                              1120  2999         $DISPATCH R0,TYPE=B,-
                              1120  3000         <-      ; Functon               ; Action
                              1120  3001                 <NO$C_EMPTY             25$>,-
                              1120  3002                 <NO$C_HEADER            SEND_HEADER>,-
                              1120  3003                 <NO$C_HCRC              SEND_HCRC>,-
                              1120  3004                 <NO$C_DATA              SEND_DATA>,-
                              1120  3005                 <NO$C_DCRC              SEND_DCRC>,-
                              1120  3006                 <NO$C_PADS              SEND_PADS>,-
                              1120  3007         >
                              1130  3008
                              1130  3009 ; Fall thru to complete the buffer and start up the next transfer if possible
                              1130  3010
                              1130  3011 14$:    CLRBIT  #NO_DS_V_XMT_TIME,UCB$W_DEVSTS(R5) ; Clear due time not valid
                   34 A4      D4  1135  3012         CLRL    NOB$L_XMT_INPR(R4)              ; Clear addr from inprogess
             0C 1F A8         02  E0  1138  3013         BBS     #XMTQ$V_CONTROL,XMTQ$B_FLAG(R8),15$ ; If BS cntrl msg don't post
                2C B4         68  0E  113D  3014         INSQUE  (R8),@NOB$Q_POST+4(R4)          ; Post the buffer for completion
                              54  DD  1141  3015         PUSHL   R4
                            0272  30  1143  3016         BSBW    SCHED_FORK_IO                   ; Schedule the fork
                              54  8ED0  1146  3017         POPL    R4
                   42 A4      94  1149  3018 15$:    CLRB    NOB$B_XSTATE(R4)                ; Reset the transmiter state
             0B 68 A5         04  E0  114C  3019         BBS     #NO_DS_V_XMTING,UCB$W_DEVSTS(R5),20$ ; If BS then turn link don't xm
                              1151  3020
                              1151  3021 ; If the following bit is set then the circuit is going thru shutdown,
                              1151  3022 ; therefore do not start up any other transmits over the port.
                              1151  3023
             06 68 A5         0A  E0  1151  3024         BBS     #NO_DS_V_SHUTDOWN,UCB$W_DEVSTS(R5),20$
                            FB86  30  1156  3025         BSBW    GET_XMT                         ; Else get next transmit to send
                         B4 50    E8  1159  3026         BLBS    R0,TO$                          ; If LBS then data to send
                              115C  3027 20$:    CLRBIT  #UCB$V_INT,UCB$W_STS(R5)        ; Set port not busy
                      03D1 8F     BA  1161  3028         POPR    #^M<R0,R4,R6,R7,R8,R9>          ; Restore registers
                              1165  3029
                              1165  3030 ; The following instruction must be the last instruction executed before
                              1165  3031 ; the return to the port input routine. The reason for this is that the
                              1165  3032 ; condition codes must correctly so that the port does the right transfer
                              1165  3033
                      010B C5     94  1165  3034         CLRB    UCB$B_TT_OUTYPE(R5)             ; Else set no data to send
                              05  1169  3035         RSB
                              116A  3036
                              116A  3037 25$:    BUG_CHECK       NOBUFPCKT,FATAL
                              116E  3038
                              116E  3039 ; This code is here because DDCMP retransmission of messages must take the
                              116E  3040 ; message in porgress away from the device. This is to ensure that
                              116E  3041 ; messages are not transmitted out of order. Since we do not want to
                              116E  3042 ; cut the message short we will send pads until the message byte count
                              116E  3043 ; has run out.
                              116E  3044
                   42 A4      95  116E  3045 30$:    TSTB    NOB$B_XSTATE(R4)                ; If EQL not sending anything
                      E9      13  1171  3046         BEQL    20$
                50 42 A4      90  1173  3047         MOVB    NOB$B_XSTATE(R4),R0             ; Get state
                         50   96  1177  3048         INCB    R0                              ; Set to next state
                42 A4    0A   90  1179  3049         MOVB    #NO$C_XMTERR,NOB$B_XSTATE(R4)   ; Set new state for xmter
                              117D  3050         $DISPATCH R0,TYPE=B,-
```

```
                                11D  3051          <-      ;Function                       ; Action
                                11D  3052                  <NO$C_EMPTY                  25$>,-
                                11D  3053                  <NO$C_HEADER                 40$>,-
                                11D  3054                  <NO$C_HCRC                   45$>,-
                                11D  3055                  <NO$C_DATA                   50$>,-
                                11D  3056                  <NO$C_DCRC                   55$>,-
                                11D  3057                  <NO$C_PADS                   60$>,-
                                11D  3058          >
      50    46 A4    B0   118D  3059          MOVW    NOB$W_XMTERR_SIZE(R4),R0
                     1F   1191  3060          BRB     65$
                          1193  3061
50    42 A5    0A   A1   1193  3062 40$:     ADDW3   #10,UCB$W_DEVBUFSIZ(R5),R0      ; Set the bytes to xmt given state
                     18   1198  3063          BRB     65$
50    42 A5    04   A1   119A  3064 45$:     ADDW3   #4,UCB$W_DEVBUFSIZ(R5),R0
                     11   119F  3065          BRB     65$
         50    04   9B   11A1  3066 50$:     MOVZBW  #4,R0
                     0C   11A4  3067          BRB     65$
         50    02   9B   11A6  3068 55$:     MOVZBW  #2,R0
                     07   11A9  3069          BRB     65$
         50    02   9B   11AB  3070 60$:     MOVZBW  #2,R0
      42 A5    05   90   11AE  3071          MOVB    #NO$C_PADS,NOB$B_XSTATE(R5)
34 A4    0080 C4   DE   11B2  3072 65$:     MOVAL   NOB$Z_CTL_MSG(R4),NOB$L_XMT_INPR(R4) ; Set to fake out xmter
      46 A4    50   B0   11B8  3073          MOVW    R0,NOB$W_XMTERR_SIZE(R4)
011C C5    44 A4    DE   11BC  3074          MOVAL   NOB$W_PADS(R4),UCB$L_TT_OUTADR(R5)
   0120 C5    02   9B   11C2  3075          MOVZBW  #2,UCB$W_TT_OUTLEN(R5)
   010B C5    01   8E   11C7  3076          MNEGB   #1,UCB$B_TT_OUTYPE(R5)
         03D1 8F   BA   11CC  3077          POPR    #^M<R0,R4,R6,R7,R8,R9>
                     05   11D0  3078          RSB
                          11D1  3079
                          11D1  3080 ; Set up the port to transmit the header portion of the message
                          11D1  3081
                          11D1  3082 SEND_HEADER:
      42 A4    01   90   11D1  3083          MOVB    #NO$C_HEADER,NOB$B_XSTATE(R4)   ; Set sending header state
011C C5    24 A8    DE   11D5  3084          MOVAL   XMTQ$B_MSGHDR(R8),UCB$L_TT_OUTADR(R5) ; Set address of header
   0120 C5    06   9B   11DB  3085          MOVZBW  #NO$C_HEADER_LEN,UCB$W_TT_OUTLEN(R5) ; Set length of header
         03D1 8F   BA   11E0  3086          POPR    #^M<R0,R4,R6,R7,R8,R9>
                          11E4  3087
                          11E4  3088 ; The following instruction must be the last instruction executed before
                          11E4  3089 ; the return to the port output routine. The reason for this is that the
                          11E4  3090 ; condition codes must correctly so that the port does the right transfer
                          11E4  3091
   010B C5    01   8E   11E4  3092          MNEGB   #1,UCB$B_TT_OUTYPE(R5)          ; Set block transfer to port
                     05   11E9  3093          RSB
                          11EA  3094
                          11EA  3095 ; Set up the port to transmit the header CRC portion of the message
                          11EA  3096
                          11EA  3097 SEND_HCRC:
      42 A4    02   90   11EA  3098          MOVB    #NO$C_HCRC,NOB$B_XSTATE(R4)     ; Set sending HCRC state
011C C5    20 A8    DE   11EE  3099          MOVAL   XMTQ$B_HCRC(R8),UCB$L_TT_OUTADR(R5) ; Set address of HCRC
   0120 C5    02   9B   11F4  3100          MOVZBW  #2,UCB$W_TT_OUTLEN(R5)          ; Set length of HCRC
         03D1 8F   BA   11F9  3101          POPR    #^M<R0,R4,R6,R7,R8,R9>
                          11FD  3102
                          11FD  3103 ; The following instruction must be the last instruction executed before
                          11FD  3104 ; the return to the port output routine. The reason for this is that the
                          11FD  3105 ; condition codes must correctly so that the port does the right transfer
                          11FD  3106
   010B C5    01   8E   11FD  3107          MNEGB   #1,UCB$B_TT_OUTYPE(R5)          ; Set block transfer to port
```

```
                     05  1202  3108          RSB
                         1203  3109
                         1203  3110 ; Set up the port to transmit the data portion of the message
                         1203  3111
                         1203  3112 SEND_DATA:
                         1203  3113 ; If the control bit is set then this is a control message and there is no
                         1203  3114 ; data to send with this message. Thus we want to send the pads out
                         1203  3115 ; now.
                         1203  3116
          34 1F A8   02  E0  1203  3117          BBS     #XMTQ$V_CONTROL,XMTQ$B_FLAG(R8),SEND_PADS
             42 A4   03  90  1208  3118          MOVB    #NO$C_DATA,NOB$B_XSTATE(R4)     ; Set sending data state
       011C C5   2A A8  DE  120C  3119          MOVAL   XMTQ$R_LENGTH(R8),UCB$L_TT_OUTADR(R5) ; Set address of data
                         1212  3120
                         1212  3121 ; Set length of the data message. Remember that DDCMP adds the header size
                         1212  3122 ; into the message size, so we should subtract it before giving the size
                         1212  3123 ; to the port.
                         1212  3124
  0120 C5   1A A8   06   A3  1212  3125          SUBW3   #NO$C_HEADER_LEN,XMTQ$W_MSGSIZE(R8),UCB$W_TT_OUTLEN(R5)
                03D1 8F   BA  1219  3126          POPR    #^M<R0,R4,R6,R7,R8,R9>
                         121D  3127
                         121D  3128 ; The following instruction must be the last instruction executed before
                         121D  3129 ; the return to the port output routine. The reason for this is that the
                         121D  3130 ; condition codes must correctly so that the port does the right transfer
                         121D  3131
          010B C5   01   8E  121D  3132          MNEGB   #1,UCB$B_TT_OUTYPE(R5)          ; Set block transfer to port
                     05  1222  3133          RSB
                         1223  3134
                         1223  3135 ; Set up the port to transmit the data CRC portion of the message
                         1223  3136
                         1223  3137 SEND_DCRC:
             42 A4   04  90  1223  3138          MOVB    #NO$C_DCRC,NOB$B_XSTATE(R4)     ; Set sending data crc state
       011C C5   22 A8  DE  1227  3139          MOVAL   XMTQ$Q_DCRC(R8),UCB$L_TT_OUTADR(R5) ; Set address of data crc
          0120 C5   02  9B  122D  3140          MOVZBW  #2,UCB$W_TT_OUTLEN(R5)          ; Set length of data CRC
                03D1 8F   BA  1232  3141          POPR    #^M<R0,R4,R6,R7,R8,R9>
                         1236  3142
                         1236  3143 ; The following instruction must be the last instruction executed before
                         1236  3144 ; the return to the port output routine. The reason for this is that the
                         1236  3145 ; condition codes must correctly so that the port does the right transfer
                         1236  3146
          010B C5   01   8E  1236  3147          MNEGB   #1,UCB$B_TT_OUTYPE(R5)          ; Set block transfer to port
                     05  123B  3148          RSB
                         123C  3149
                         123C  3150 ; Set up the port to transmit the pads for the message
                         123C  3151
                         123C  3152 SEND_PADS:
             42 A4   05  90  123C  3153          MOVB    #NO$C_PADS,NOB$B_XSTATE(R4)     ; Set sending pads state
       011C C5   44 A4   DE  1240  3154          MOVAL   NOB$W_PADS(R4),UCB$L_TT_OUTADR(R5) ; Set address of pads
          0120 C5   02  9B  1246  3155          MOVZBW  #2,UCB$W_TT_OUTLEN(R5)          ; Set length of pads
                03D1 8F   BA  124B  3156          POPR    #^M<R0,R4,R6,R7,R8,R9>
                         124F  3157
                         124F  3158 ; The following instruction must be the last instruction executed before
                         124F  3159 ; the return to the port output routine. The reason for this is that the
                         124F  3160 ; condition codes must correctly so that the port does the right transfer
                         124F  3161
          010B C5   01   8E  124F  3162          MNEGB   #1,UCB$B_TT_OUTYPE(R5)          ; Set block transfer to port
                     05  1254  3163          RSB
                         1255  3164
```

```
                        1255  3165
                        1255  3166 ; There was an error on the line and this mechanism is used to ensure
                        1255  3167 ; that the other side gets enough data to cause a CRC error. Please
                        1255  3168 ; note that that the last time thru we set the PAD state in order
                        1255  3169 ; to start up the next XMT
                        1255  3170
                        1255  3171 SEND_XMTERR:
      46 A4    02   A2   1255  3172          SUBW2   #2,NOB$W_XMTERR_SIZE(R4)        ; Decr number of pads to send
               04   14   1259  3173          BGTR    10$                             ; If gtr continue
      42 A4    05   90   125B  3174          MOVB    #NO$C_PADS,NOB$B_XSTATE(R4)     ; Else reset state to start transmit
011C C5   44 A4    DE   125F  3175 10$:      MOVAL   NOB$W_PADS(R4),UCB$L_TT_OUTADR(R5) ; Set address of pads
  0120 C5    02    9B   1265  3176          MOVZBW  #2,UCB$W_TT_OUTLEN(R5)          ; Set length of pads
       03D1  8F    BA   126A  3177          POPR    #^M<R0,R4,R6,R7,R8,R9>
                        126E  3178
                        126E  3179 ; The following instruction must be the last instruction executed before
                        126E  3180 ; the return to the port output routine. The reason for this is that the
                        126E  3181 ; condition codes must correctly so that the port does the right transfer
                        126E  3182
  010B C5    01    8E   126E  3183          MNEGB   #1,UCB$B_TT_OUTYPE(R5)          ; Set block transfer to port
                   05   1273  3184          RSB
```

```
                                1274  3186          .SBTTL  NO$PUTNXT - Asynch class driver receive character routine
                                1274  3187 ;++
                                1274  3188 ; NO$PUTNXT - Asynch class driver receive character routine
                                1274  3189 ;
                                1274  3190 ; FUNCTIONAL DESCRIPTION:
                                1274  3191 ;
                                1274  3192 ; This routine is called whenever the terminal port driver has received a
                                1274  3193 ; character. Because of the nature of ASYNCH DDCMP this routine will
                                1274  3194 ; not return data to transmit over the port.
                                1274  3195 ;
                                1274  3196 ;       INPUTS:
                                1274  3197 ;               R3 = Input character
                                1274  3198 ;               R5 = UCB address
                                1274  3199 ;
                                1274  3200 ;       OUTPUTS:
                                1274  3201 ;               R5 = UCB address
                                1274  3202 ;               R0 is preserved ALL other registers are destroyed
                                1274  3203 ;
                                1274  3204 ;       IMPLICT OUTPUTS:
                                1274  3205 ;               UCB$B_TT_OUTYPE = Zero no data to send
                                1274  3206 ;
                                1274  3207 ;--
                                1274  3208 NO$PUTNXT:
     03D1 8F    BB              1274  3209          PUSHR   #^M<R0,R4,R6,R7,R8,R9>
54   00A8 C5    D0              1278  3210          MOVL    UCB$L_NO_BUFFER(R5),R4          ; Get address of NOB
          16    13              127D  3211          BEQL    30$                             ; If EQL device not started
  50   43 A4    9A              127F  3212 10$:     MOVZBL  NOB$B_RSTATE(R4),R0             ; Get receiver state
                                1283  3213          $DISPATCH R0,TYPE=B,-
                                1283  3214          <-      ; Function              ; Action
                                1283  3215                  <NO$C_SEAR_MSG  SEARCH_MESSAGE>,-
                                1283  3216                  <NO$C_HEADER    RCV_HEADER>,-
                                1283  3217                  <NO$C_DATA      RCV_DATA>,-
                                1283  3218                  <NO$C_NOBUFFER  RCV_NOBUFFER>,-
                                1283  3219          >
                                1291  3220          BUG_CHECK       NOBUFPCKT,FATAL
                                1295  3221
     03D1 8F    BA              1295  3222 30$:     POPR    #^M<R0,R4,R6,R7,R8,R9>          ; Else ignore the char and return
                                1299  3223
                                1299  3224 ; The following instruction must be the last instruction executed before
                                1299  3225 ; the return to the port input routine. The reason for this is that the
                                1299  3226 ; condition codes must correctly so that the port does the right transfer
                                1299  3227
     010B C5    94              1299  3228          CLRB    UCB$B_TT_OUTYPE(R5)             ; Else set no data to send
                05              129D  3229          RSB
                                129E  3230
                                129E  3231 ; This searches for the start of a DDCMP message. All characters are ignored
                                129E  3232 ; until a start of message is received.
                                129E  3233
                                129E  3234 SEARCH_MESSAGE:
81 8F    53    91              129E  3235          CMPB    R3,#NO$C_SOH                    ; Is this a start of a data message
          10    13              12A2  3236          BEQL    10$                             ; If EQL then it is
90 8F    53    91              12A4  3237          CMPB    R3,#NO$C_DLE                    ; Start of a maintenance msg?
          0A    13              12A8  3238          BEQL    10$                             ; If EQL then yes
   05    53    91              12AA  3239          CMPB    R3,#NO$C_ENQ                    ; Start of control message?
          35    12              12AD  3240          BNEQ    20$                             ; If NEQ then no
                                12AF  3241          SETBIT  #RCV_V_CNTL,NOB$W_FLAGS(R4)     ; Set this is a control msg
                                12B4  3242
```

```
                                12B4  3243          ASSUME NO$C_HEADER EQ NO$C_SEAR_MSG+1
                                12B4  3244
                                12B4  3245 ; The terminal line receive speed is store in the UCB$W_TT_SPEED+1 unless
                                12B4  3246 ; the field is zero in which case it is the transmit speed
     50    00F5 C5    9A      12B4  3247 10$:     MOVZBL  UCB$W_TT_SPEED+1(R5),R0
                 05    12      12B9  3248          BNEQ    15$
     50    00F4 C5    9A      12BB  3249          MOVZBL  UCB$W_TT_SPEED(R5),R0
   50    EE3F CF40    3C      12C0  3250 15$:     MOVZWL  DUETIM_TABLE[R0],R0
 51    00000000'GF    D0      12C6  3251          MOVL    G^EXE$GL_ABSTIM,R1
  38 A4    51    50    C1      12CD  3252          ADDL3   R0,R1,NOB$L_RCV_DUETIM(R4)
                                12D2  3253          SETBIT  #NO_DS_V_RCV_TIME,UCB$W_DEVSTS(R5)
            43 A4    96      12D7  3254          INCB    NOB$B_RSTATE(R4)                ; Move to next state rcving hdr
         48 A4    53    90      12DA  3255          MOVB    R3,NOB$Z_HEADER(R4)             ; Save character rcvd in hdr area
            52 A4    B6      12DE  3256          INCW    NOB$W_INDEX(R4)                 ; Incr the index into buffer
            50 A4    B7      12E1  3257          DECW    NOB$W_MSGSIZ(R4)                ; Decr the number of bytes to rcv
         03D1 8F     BA      12E4  3258 20$:     POPR    #^M<R0,R4,R6,R7,R8,R9>
                                12E8  3259
                                12E8  3260 ; The following instruction must be the last instruction executed before
                                12E8  3261 ; the return to the port input routine. The reason for this is that the
                                12E8  3262 ; condition codes must correctly so that the port does the right transfer
                                12E8  3263
            010B C5    94      12E8  3264          CLRB    UCB$B_TT_OUTYPE(R5)             ; Else set no data to send
                 05    12EC  3265          RSB
                                12ED  3266
                                12ED  3267 ; This routine inputs each character into the header of the rcv buffer unitl
                                12ED  3268 ; both the header and header CRC are input.
                                12ED  3269
                                12ED  3270 RCV_HEADER:
     50    52 A4    3C      12ED  3271          MOVZWL  NOB$W_INDEX(R4),R0              ; Get index into rcv buffer
     51    48 A4    DE      12F1  3272          MOVAL   NOB$Z_HEADER(R4),R1             ; Get pointer to start of rcv buffer
      6140    53    90      12F5  3273          MOVB    R3,(R1)[R0]                     ; Move char into buffer
         52 A4    B6      12F9  3274          INCW    NOB$W_INDEX(R4)                 ; Incr the index
         50 A4    B7      12FC  3275          DECW    NOB$W_MSGSIZ(R4)                ; Decr the bytes to rcv
            26    14      12FF  3276          BGTR    10$                             ; If GTR still more bytes to rcv
                                1301  3277
                                1301  3278 ; After the entire header and crc's are received copy them into a buffer.
                                1301  3279 ; Set up the unit to make it start receiving the data into the message
                                1301  3280
                                1301  3281 ;       BBS     #NO_DS_V_SHUTDOWN,UCB$W_DEVSTS(R5),30$ ; If device is shut dont star
          FDD7    30      1301  3282          BSBW    GET_RECEIVE                     ; Start receive
         29 50    E9      1304  3283          BLBC    R0,20$                          ; If LBS then buffer to rcv into
                                1307  3284
                                1307  3285          ASSUME  RCV_V_CNTL EQ 0
                                1307  3286
      67 12 A2    E8      1307  3287          BLBS    RCV_W_FLAGS(R2),COMP_RCV        ; If LBS complete the control msg
   50    01 A1    B0      130B  3288          MOVW    MFD$W_CNTFLG(R1),R0             ; Else get size of data portion of m
50    C000 8F     AA      130F  3289          BICW2   #<MFD$M_SELECT!MFD$M_QSYNC>,R0  ; Clear the flags from the msg size
                                1314  3290
                                1314  3291 ; The following code is here so that we can catch the fact that systems run
                                1314  3292 ; with different buffer sizes. We don't want to corrupt pool if
                                1314  3293 ; we receive a message that is too large for our receive buffer. Incidently,
                                1314  3294 ; the CRC checking should catch this as an error and drop the message on the
                                1314  3295 ; floor.
                                1314  3296
     50    42 A5    B1      1314  3297          CMPW    UCB$W_DEVBUFSIZ(R5),R0          ; If GEQ then not larger than our bu
            04    18      1318  3298          BGEQ    5$
     50    42 A5    B0      131A  3299          MOVW    UCB$W_DEVBUFSIZ(R5),R0          ; Use the smaller of the two
```

```
                OC A2    50    02    A1   131E  3300 5$:     ADDW3   #2,R0,RCV_W_MSGSIZ(R2)          ; Set size of msg plus 2 for CRC
                    43   A4    03    90   1323  3301         MOVB    #NO$C_DATA,NOB$B_RSTATE(R4)     ; Set receive data state
                         03D1  8F    BA   1327  3302 10$:    POPR    #^M<R0,R4,R6,R7,R8,R9>
                                           132B  3303
                                           132B  3304 ; The following instruction must be the last instruction executed before
                                           132B  3305 ; the return to the port input routine. The reason for this is that the
                                           132B  3306 ; condition codes must correctly so that the port does the right transfer
                                           132B  3307
                               010B C5     94   132B  3308         CLRB    UCB$B_TT_OUTYPE(R5)             ; Else set no data to send
                                           05   132F  3309         RSB
                                           1330  3310
                                           1330  3311 ; Come here when we did not have a buffer in which to receive the data
                                           1330  3312
                                 4D 56 A4  E8   1330  3313 20$:    BLBS    NOB$W_FLAGS(R4),COMP_RCV_NOBUFFER ; If LBS complete the control msg
                              50    01 A1  B0   1334  3314         MOVW    MFD$W_CNTFLG(R1),R0             ; Else get size of data portion of m
                           50    C000 8F  AA   1338  3315         BICW2   #<MFD$M_SELECT!MFD$M_QSYNC>,R0  ; Clear the flags from the msg size
                        50 A4    50    02   A1   133D  3316         ADDW3   #2,R0,NOB$W_MSGSIZ(R4)          ; Set size of msg plus 2 for CRC
                            43   A4    04   90   1342  3317         MOVB    #NO$C_NOBUFFER,NOB$B_RSTATE(R4) ; Set nobuffer receive data state
                                 03D1  8F   BA   1346  3318         POPR    #^M<R0,R4,R6,R7,R8,R9>
                                           134A  3319
                                           134A  3320 ; The following instruction must be the last instruction executed before
                                           134A  3321 ; the return to the port input routine. The reason for this is that the
                                           134A  3322 ; condition codes must correctly so that the port does the right transfer
                                           134A  3323
                               010B C5     94   134A  3324         CLRB    UCB$B_TT_OUTYPE(R5)             ; Else set no data to send
                                           05   134E  3325         RSB
                                           134F  3326
                                           134F  3327 ; This routine inputs each character into the data part of the rcv buffer unitl
                                           134F  3328 ; both the data and data CRC are input.
                                           134F  3329
                                           134F  3330 RCV_DATA:
                        52    30 A4    D0   134F  3331         MOVL    NOB$L_RCV_INPR(R4),R2           ; Get buffer
                                 14    13   1353  3332         BEQL    10$                             ; Branch to return if not there
                           50    0E A2  3C   1355  3333         MOVZWL  RCV_W_INDEX(R2),R0              ; Get index into rcv buffer
                           51    40 A2  DE   1359  3334         MOVAL   RCV_Z_HEADER(R2),R1             ; Get pointer to start of rcv buffer
                            6140    53  90   135D  3335         MOVB    R3,(R1)[R0]                     ; Move char into buffer
                                 0E A2  B6   1361  3336         INCW    RCV_W_INDEX(R2)                 ; Incr the index
                                 0C A2  B7   1364  3337         DECW    RCV_W_MSGSIZ(R2)                ; Decr the bytes to rcv
                                    09  15   1367  3338         BLEQ    COMP_RCV                        ; If LEQ then rcv is complete
                            03D1 8F    BA   1369  3339 10$:    POPR    #^M<R0,R4,R6,R7,R8,R9>
                                           136D  3340
                                           136D  3341 ; The following instruction must be the last instruction executed before
                                           136D  3342 ; the return to the port input routine. The reason for this is that the
                                           136D  3343 ; condition codes must correctly so that the port does the right transfer
                                           136D  3344
                            010B C5     94   136D  3345         CLRB    UCB$B_TT_OUTYPE(R5)             ; Else set no data to send
                                        05   1371  3346         RSB
                                           1372  3347
                                           1372  3348 ; This routine posts the current receive for IO completion then sets up
                                           1372  3349 ; up the next buffer into which to receive.
                                           1372  3350
                                           1372  3351 COMP_RCV:
                                 30 A4    D4   1372  3352         CLRL    NOB$L_RCV_INPR(R4)              ; Clear no rcv buffer inprg
                        2C B4      62    0E   1375  3353         INSQUE  (R2),@NOB$Q_POST+4(R4)          ; Send buffer to post
                                    54    DD   1379  3354         PUSHL   R4
                                  003A    30   137B  3355         BSBW    SCHED_FORK_IO                   ; Set up fork
                                    54 8ED0   137E  3356         POPL    R4
```

```
                     1381  3357 COMP_RCV_NOBUFFER:
                     1381  3358         CLRBIT  #NO_DS_V_RCV_TIME,UCB$W_DEVSTS(R5) ; Set due time not valid
   FD44    30        1386  3359         BSBW    START_RECEIVE                     ; Set up next receive
03D1 8F    BA        1389  3360         POPR    #^M<R0,R4,R6,R7,R8,R9>
                     138D  3361
                     138D  3362 ; The following instruction must be the last instruction executed before
                     138D  3363 ; the return to the port input routine. The reason for this is that the
                     138D  3364 ; condition codes must correctly so that the port does the right transfer
                     138D  3365
010B C5    94        138D  3366         CLRB    UCB$B_TT_OUTYPE(R5)               ; Else set no data to send
           05        1391  3367         RSB
                     1392  3368
                     1392  3369 RCV_NOBUFFER:
  50 A4    B7        1392  3370         DECW    NOB$W_MSGSIZ(R4)
     EA    15        1395  3371         BLEQ    COMP_RCV_NOBUFFER
03D1 8F    BA        1397  3372         POPR    #^M<R0,R4,R6,R7,R8,R9>
                     139B  3373
                     139B  3374 ; The following instruction must be the last instruction executed before
                     139B  3375 ; the return to the port input routine. The reason for this is that the
                     139B  3376 ; condition codes must correctly so that the port does the right transfer
                     139B  3377
010B C5    94        139B  3378         CLRB    UCB$B_TT_OUTYPE(R5)               ; Else set no data to send
           05        139F  3379         RSB
                     13A0  3380
                     13A0  3381
                     13A0  3382
```

```
                        13A0  3384          .SBTTL  NO$CLASS_PORTFORK - Asycnh DDCMP fork routines
                        13A0  3385 ;++
                        13A0  3386 ; NO$CLASS_PORTFORK - Asynch DDCMP port fork routine
                        13A0  3387 ;
                        13A0  3388 ; FUNCTIONAL DESCRIPTION:
                        13A0  3389 ;
                        13A0  3390 ; This routine is called by the port driver when the port driver needs to start
                        13A0  3391 ; up a fork process. This routine only schedules a fork if no fork is already
                        13A0  3392 ; scheduled. If a fork is scheduled then a bit is set in UCB$L_FR3 to indicate
                        13A0  3393 ; that the port driver must be called at its FORK entry point.
                        13A0  3394 ;
                        13A0  3395 ;       INPUTS:
                        13A0  3396 ;               R5 = UCB address
                        13A0  3397 ;
                        13A0  3398 ;       OUTPUTS:
                        13A0  3399 ;               FORK may be scheduled
                        13A0  3400 ;
                        13A0  3401 ;--
                        13A0  3402 NO$CLASS_PORTFORK:
0D 68 A5   07   E2      13A0  3403          BBSS    #NO_DS_V_FORK_PEND,UCB$W_DEVSTS(R5),10$ ; If BS then fork pending
      1451'CF   3F      13A5  3404          PUSHAW  W^FORK_DONE                             ; Set address of fork routine
      53   01   9A      13A9  3405          MOVZBL  #NO_FS_M_PORTFORK,R3                    ; Set type of fork
00000000'GF    17      13AC  3406          JMP     G^EXE$FORK
                        13B2  3407
                        13B2  3408 10$:     SETBIT  #NO_FS_V_PORTFORK,UCB$L_FR3(R5) ; Set port fork pending
                  05    13B7  3409          RSB
                        13B8  3410
                        13B8  3411 ;++
                        13B8  3412 ; SCHED_FORK_IO - Schedules forks for asynch ddcmp class driver
                        13B8  3413 ;
                        13B8  3414 ; FUNCTIONAL DESCRIPTION:
                        13B8  3415 ;
                        13B8  3416 ; This routine schedules the IO completion forks for the asynch class
                        13B8  3417 ; driver. If there is a fork pending on the device then no fork
                        13B8  3418 ; is scheduled but a bit is set in UCB$L_FR3 indicating that there is
                        13B8  3419 ; some IO to complete when the fork is processed.
                        13B8  3420 ;
                        13B8  3421 ;       INPUTS:
                        13B8  3422 ;               R5 = UCB address
                        13B8  3423 ;
                        13B8  3424 ;       OUTPUTS:
                        13B8  3425 ;               FORK may be scheduled
                        13B8  3426 ;
                        13B8  3427 ;--
                        13B8  3428 SCHED_FORK_IO:
0D 68 A5   07   E2      13B8  3429          BBSS    #NO_DS_V_FORK_PEND,UCB$W_DEVSTS(R5),10$ ; If BS then fork pending
      1451'CF   3F      13BD  3430          PUSHAW  W^FORK_DONE                             ; Set address of fork routine
      53   02   9A      13C1  3431          MOVZBL  #NO_FS_M_IOFORK,R3                      ; Set IO fork
00000000'GF    17      13C4  3432          JMP     G^EXE$FORK
                        13CA  3433
                        13CA  3434 10$:     SETBIT  #NO_FS_V_IOFORK,UCB$L_FR3(R5)   ; Set port fork pending
                  05    13CF  3435          RSB
                        13D0  3436
                        13D0  3437 SCHED_FORK_POWERFAIL:
0D 68 A5   07   E2      13D0  3438          BBSS    #NO_DS_V_FORK_PEND,UCB$W_DEVSTS(R5),10$ ; If BS then fork pending
      1451'CF   3F      13D5  3439          PUSHAW  W^FORK_DONE                             ; Set address of fork routine
      53   04   9A      13D9  3440          MOVZBL  #NO_FS_M_POWERFAIL,R3                   ; Set Powerfail fork
```

```
                00000000'GF   17  13DC  3441          JMP     G^EXE$FORK
                                  13E2  3442
                                  13E2  3443 10$:      SETBIT  #DDB$V_FS_V_POWERFAIL,UCB$L_FR3(R5) ; Set port fork pending
                              05  13E7  3444          RSB
                                  13E8  3445
```

```
                        .SBTTL  NO$SETUP_UCB - Asynch DDCMP set up UCB
                ;++
                ; NO$SETUP_UCB - Asynch DDCMP set up UCB
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ; This does nothing
                ;
                ;       INPUTS:
                ;               R5 = UCB address
                ;
                ;       OUTPUTS:
                ;               NONE
                ;
                ;--
                NO$SETUP_UCB:
05              RSB
```

```
                13E9  3466          .SBTTL  NO$PORT_TRANSITION - Asynch DDCMP port transition routine
                13E9  3467 ;++
                13E9  3468 ; NO$PORT_TRANSITION - Asynch DDCMP port transition routine
                13E9  3469 ;
                13E9  3470 ; FUNCTIONAL DESCRIPTION:
                13E9  3471 ;
                13E9  3472 ; This does nothing
                13E9  3473 ;
                13E9  3474 ;       INPUTS:
                13E9  3475 ;               R1 = Transition type (one of the following)
                13E9  3476 ;                    MODEM$C_INIT - initialize the modem control
                13E9  3477 ;                    MODEM$C_SHUTDWN - Shut down the line and protocol
                13E9  3478 ;                    MODEM$C_DATASET - Data set signal changes
                13E9  3479 ;               R2 = Type specific argument
                13E9  3480 ;                    MODEM$C_DATASET - new receive modem mask
                13E9  3481 ;               R5 = UCB address
                13E9  3482 ;
                13E9  3483 ;       OUTPUTS:
                13E9  3484 ;               NONE
                13E9  3485 ;
                13E9  3486 ;--
                13E9  3487 NO$PORT_TRANSITION:
05              13E9  3488          RSB
                13EA  3489
```

```
                                13EA  3491
                                13EA  3492
                                13EA  3493          .SBTTL  NO$READERROR - Asynch DDCMP read error routine
                                13EA  3494 ;++
                                13EA  3495 ; NO$READERROR - Asynch DDCMP read error routine
                                13EA  3496 ;
                                13EA  3497 ; FUNCTIONAL DESCRIPTION:
                                13EA  3498 ;
                                13EA  3499 ; This routine deciphers the error sent by the port driver into an
                                13EA  3500 ; error code to give to the protocol.
                                13EA  3501 ;
                                13EA  3502 ;         INPUTS:
                                13EA  3503 ;                 R3 = Character and flags:
                                13EA  3504 ;                      Flags:
                                13EA  3505 ;                           Bit 12 - parity error on the given character
                                13EA  3506 ;                           Bit 13 - framing error on the given character
                                13EA  3507 ;                           Bit 14 - data overrun
                                13EA  3508 ;                 R5 = UCB address
                                13EA  3509 ;
                                13EA  3510 ;         OUTPUTS:
                                13EA  3511 ;                 NONE
                                13EA  3512 ;
                                13EA  3513 ;--
                                13EA  3514 NO$READERROR:
      01D5 8F    BB             13EA  3515          PUSHR   #^M<R0,R2,R4,R6,R7,R8>
54    00A8 C5    D0             13EE  3516          MOVL    UCB$L_NO_BUFFER(R5),R4          ; Get NOB address
           16    13             13F3  3517          BEQL    10$
  50     43 A4    9A            13F5  3518          MOVZBL  NOB$B_RSTATE(R4),R0             ; Get receiver state
                                13F9  3519          $DISPATCH R0,TYPE=B,-
                                13F9  3520          <-      ; Function                  ; Action
                                13F9  3521                  <NO$C_SEAR_MSG              NO_ERROR>,-
                                13F9  3522                  <NO$C_HEADER                HEADER_ERROR>,-
                                13F9  3523                  <NO$C_DATA                  DATA_ERROR>,-
                                13F9  3524                  <NO$C_NOBUFFER              NOBUFFER_ERROR>,-
                                13F9  3525          >
                                1407  3526          BUG_CHECK       NOBUFPCKT,FATAL
                                140B  3527 10$:
                                140B  3528 NO_ERROR:
      01D5 8F    BA             140B  3529          POPR    #^M<R0,R2,R4,R6,R7,R8>
           05    05             140F  3530          RSB
                                1410  3531
                                1410  3532 ; If the device is rcving the header and it did not get an overrun error
                                1410  3533 ; then set that this is a header crc error.
                                1410  3534
                                1410  3535 HEADER_ERROR:
                                1410  3536
                                1410  3537 ; Get a buffer from the queue of free buffers.  If there
                                1410  3538 ; is no buffer available then the error cna not be reported.
                                1410  3539
        FCC8    30              1410  3540          BSBW    GET_RECEIVE
      2B 50     E9              1413  3541          BLBC    R0,NOBUFFER_ERROR
06 53    0E     E0              1416  3542          BBS     #^XE,R3,OVERRUN_ERROR
10 A2    02     B0              141A  3543          MOVW    #DLK$M_HDRCRC,RCV_W_ERROR(R2)
           12    11             141E  3544          BRB     COMP_RCV_ERROR
                                1420  3545
                                1420  3546 ; Set that the device got a receive overrun error
                                1420  3547
```

```
                                1420  3548 OVERRUN_ERROR:
10 A2    20    B0               1420  3549         MOVW    #DLK$M_RCVOVR,RCV_W_ERROR(R2)
         0C    11               1424  3550         BRB     COMP_RCV_ERROR
                                1426  3551
                                1426  3552 ; If the device was receiving data then set that this is a data crc error.
                                1426  3553
                                1426  3554 DATA_ERROR:
52    30 A4    D0               1426  3555         MOVL    NOB$L_RCV_INPR(R4),R2           ; Get receive buffer
F2 53    0E    E0               142A  3556         BBS     #^XE,R3,OVERRUN_ERROR
10 A2    04    B0               142E  3557         MOVW    #DLK$M_DATACRC,RCV_W_ERROR(R2)
                                1432  3558         ;BRB    COMP_RCV_ERROR
                                1432  3559
                                1432  3560 COMP_RCV_ERROR:
      30 A4    D4               1432  3561         CLRL    NOB$L_RCV_INPR(R4)              ; Clear rcv inpr
2C B4    62    0E               1435  3562         INSQUE  (R2),@NOB$Q_POST+4(R4)          ; Put buffer on queue to comp
         54    DD               1439  3563         PUSHL   R4
      FF7A    30               143B  3564         BSBW    SCHED_FORK_IO                   ; Sched a fork to comp buffer
         54 8ED0               143E  3565         POPL    R4
                                1441  3566 NOBUFFER_ERROR:
      FC89    30               1441  3567         BSBW    START_RECEIVE                   ; Start up next receive
   01D5 8F    BA               1444  3568         POPR    #^M<R0,R2,R4,R6,R7,R8>
              05               1448  3569         RSB
                               1449  3570
```

```
                        .SBTTL  NO$CLASS_DISCONNECT - Asynch DDCMP class disconnect
                ;++
                ; NO$CLASS_DISCONNECT - Asynch DDCMP class disconnect
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ; This does nothing
                ;
                ;       INPUTS:
                ;               R5 = UCB address
                ;
                ;       OUTPUTS:
                ;               NONE
                ;
                ;--
                NO$CLASS_DISCONNECT:
05              RSB
```

```
                     144A  3593          .SBTTL  NO$CLASS_POWERACTION - Asynch DDCMP power action
                     144A  3594 ;++
                     144A  3595 ; NO$CLASS_POWERACTION - Asynch DDCMP power action
                     144A  3596 ;
                     144A  3597 ; FUNCTIONAL DESCRIPTION:
                     144A  3598 ;
                     144A  3599 ; This routine forks to shutdown the device on a power fail.
                     144A  3600 ;
                     144A  3601 ;       INPUTS:
                     144A  3602 ;               R5 = UCB address
                     144A  3603 ;
                     144A  3604 ;       OUTPUTS:
                     144A  3605 ;               NONE
                     144A  3606 ;
                     144A  3607 ;--
                     144A  3608 NO$CLASS_POWERACTION:
FF83   30            144A  3609          BSBW    SCHED_FORK_POWERFAIL
       05            144D  3610          RSB
```

```
                        .SBTTL  NO$NULL - Asynch DDCMP null routine
                ;++
                ; NO$NULL - Asynch DDCMP null routine
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ; This does nothing
                ;
                ;       INPUTS:
                ;               R5 = UCB address
                ;
                ;       OUTPUTS:
                ;               NONE
                ;
                ;--
                NO$NULL:
05              RSB

                .END
```

```
                         144F
                         144F         .SBTTL  FORKDONE - Fork process
                         144F  ;++
                         144F  ; FORKDONE - Fork process
                         144F  ;
                         144F  ; FUNCTIONAL DESCRIPTION:
                         144F  ;
                         144F  ; This routine is entered for two reasons either to complete some IO, or
                         144F  ; because the port has some work that must be done at FORK level.
                         144F  ;
                         144F  ;      INPUTS:
                         144F  ;
                         144F  ;              R4 = NOB address
                         144F  ;              R5 = UCB address
                         144F  ;
                         144F  ;      IMPLICIT INPUTS:
                         144F  ;              UCB$L_FR3 = Type of fork
                         144F  ;
                         144F  ;      IPL = FIPL
                         144F  ;
                         144F  ;      OUTPUTS:
                         144F  ;              R5 = UCB address
                         144F  ;              R0,R6 - R9 are preserved.
                         144F  ;
                         144F  ;      If a receive I/O request is pending, the receive is completed.
                         144F  ;      Otherwise, queue the message for a future I/O.
                         144F  ;
                         144F  ;--
                  02C2'  144F         .WORD   TIMEOUT-.
                         1451  FORK_DONE:
                         1451         DSBINT  UCB$B_DIPL(R5)                  ; Disable device interrupts
                         1458         CLRBIT  #NO_DS_V_FORK_PEND,UCB$W_DEVSTS(R5) ; Clear fork pending
   0D 10 A5    00    E5  145D         BBCC    #NO_FS_V_PORTFORK,UCB$L_FR3(R5),5$ ; If BC don't call portfork
                         1462         ENBINT
  50   0118 C5    D0     1465         MOVL    UCB$L_TT_PORT(R5),R0            ; Get port vector table
          34 B0    16    146A         JSB     @PORT_FORKRET(R0)               ; Call port fork routine
             03    11    146D         BRB     10$
                         146F  5$:    ENBINT
                         1472  10$:   DSBINT  UCB$B_DIPL(R5)                  ; Disable device interrupts
                         1479         CLRBIT  #NO_FS_V_IOFORK,UCB$L_FR3(R5)
                         147E
                         147E  ; Check for a powerfail and shut down the device if one has happened
                         147E
   42 10 A5    02    E4  147E         BBSC    #NO_FS_V_POWERFAIL,UCB$L_FR3(R5),45$
                         1483         ENBINT
  54   00A8 C5    D0     1486         MOVL    UCB$L_NO_BUFFER(R5),R4          ; Get NOB address
       03C0 8F    BB     148B         PUSHR   #^M<R6,R7,R8,R9>                ; Save registers
  52     28 B4    0F     148F  15$:   REMQUE  @NOB$Q_POST(R4),R2              ; Get next buffer to complete
             1C    1D    1493         BVS     30$                             ; If VS then no buffer found
  13     0A A2    91     1495         CMPB    IRP$B_TYPE(R2),S^#DYN$C_BUFIO   ; If NEQ then not XMT buffer
             08    12    1499         BNEQ    20$
           0151    30    149B         BSBW    TRANSMIT_DONE                   ; Complete the transmit
          18 50    E9    149E         BLBC    R0,40$                          ; If BC then error on buffer
             EC    11    14A1         BRB     15$                             ; Get next entry to post
  17     0A A2    91     14A3  20$:   CMPB    IRP$B_TYPE(R2),S^#DYN$C_NET     ; If EQL then RCV buffer
             2E    12    14A7         BNEQ    60$
           002F    30    14A9         BSBW    RECEIVE_DONE                    ; Complete the Receive
```

```
      0A 50   E9  14AC  3688            BLBC    R0,40$                          ; If BC then error on buffer
          DE  11  14AF  3689            BRB     15$                             ; Get next entry to post
   50     01  3C  14B1  3690 30$:       MOVZWL  S^#SS$_NORMAL,R0                ; Set status
   03C0   8F  BA  14B4  3691            POPR    #^M<R6,R7,R8,R9>                ; Restore registers
              05  14B8  3692            RSB                                     ; Return
                  14B9  3693
   03C0   8F  BA  14B9  3694 40$:       POPR    #^M<R6,R7,R8,R9>                ; Restore registers
          16  E1  14BD  3695            BBC     #XM$V_ERR_TRIB,-                ; Assume trib shutdown
OD 44 A5          14BF  3696                    UCB$L_DEVDEPEND(R5),50$
        041B  31  14C2  3697            BRW     SHUTDOWN_CIRCUIT
                  14C5  3698
                  14C5  3699 45$:       ENBINT
   0082 C5  B6  14C8  3700            INCW    UCB$W_ERRCNT(R5)
        0212  30  14CC  3701            BSBW    POKE_OSER
                  14CF  3702 50$:       SETBIT  #XM$V_ERR_FATAL,UCB$L_DEVDEPEND(R5) ; Set fatal error
        0377  31  14D4  3703            BRW     SHUTDOWN_LINE                   ; Br to shut down the device
                  14D7  3704
                  14D7  3705 60$:       BUG_CHECK NOBUFPCKT,FATAL               ; Else fatal, error
                  14DB  3706
```

```
                        14DB  3708          .SBTTL  RECEIVE_DONE - Complete a receive buffer
                        14DB  3709
                        14DB  3710 ;++
                        14DB  3711 ;RECEIVE_DONE
                        14DB  3712 ;
                        14DB  3713 ; FUNCTIONAL DESCRIPTION:
                        14DB  3714 ;
                        14DB  3715 ; When the class driver finishes a receive buffer this routine is called to
                        14DB  3716 ; post the receive. First
                        14DB  3717 ; a call is made to the protocol to strip off the header and check the CRC's
                        14DB  3718 ; to ensure the data made it over the wire ok. If an error is detected then
                        14DB  3719 ; the error is recorded in the protocol and the buffer is returned to the
                        14DB  3720 ; free buffer pool, and NOT given to the user. If no error occurs and the
                        14DB  3721 ; buffer is a protocol control message then the buffer is returned to the free
                        14DB  3722 ; buffer pool. If this is a data message and a IRP is free the buffer is
                        14DB  3723 ; completed with the IRP, else it is put on a queue to await I/O completion.
                        14DB  3724 ;
                        14DB  3725 ;       INPUTS:
                        14DB  3726 ;               R2 = Message buffer address
                        14DB  3727 ;               R4 = NOB address
                        14DB  3728 ;               R5 = UCB address
                        14DB  3729 ;
                        14DB  3730 ;       OUTPUTS:
                        14DB  3731 ;               R0 = Status
                        14DB  3732 ;               R2,R3,R4, and R5 are preserved
                        14DB  3733 ;--
                        14DB  3734 RECEIVE_DONE:                                        ; Receive done
            3C   BB     14DB  3735          PUSHR   #^M<R2,R3,R4,R5>                    ; Save registers
       56   01   9A     14DD  3736          MOVZBL  #DLK$C_RCVMSG,R6                    ; Set that this is a RCV
   57   10 A2   B0     14E0  3737          MOVW    RCV_W_ERROR(R2),R7                  ; Set error bits
   5?   40 A2   DE     14E4  3738          MOVAL   RCV_Z_HEADER(R2),R8                 ; Get buffer address
            59   D4     14E8  3739          CLRL    R9                                  ; Clear reg
   55   0C A4   D0     14EA  3740          MOVL    NOB$A_PRO_BUFFER(R4),R5             ; Set protocol buffer address
          EB0F'  30     14EE  3741          BSBW    DDCMP
            3C   BA     14F1  3742          POPR    #^M<R2,R3,R4,R5>                    ; Restore registers
   0C A2   59   B0     14F3  3743          MOVW    R9,RCV_W_MSGSIZ(R2)                 ; Set transfer size
       56   09   91     14F7  3744          CMPB    #DLK$C_ACTNOTCOM,R6                 ; If EQL then protl not active
            4E   13     14FA  3745          BEQL    40$
       57   0A   B3     14FC  3746          BITW    #<DLK$M_PRSTERR!-                   ; If NEQ then fatal protocol
                        14FF  3747                  DLK$M_HDRERR>,R7                    ; error go to shutdown the
            51   12     14FF  3748          BNEQ    55$                                 ; circuit
   57   00C0 8F   B3     1501  3749          BITW    #<DLK$M_STRTRCV!-                   ; If NEQ then trib error
                        1506  3750                  DLK$M_MNTRCV>,R7                    ; go to shutdown the circuit
            51   12     1506  3751          BNEQ    60$
   3E 57    10   E0     1508  3752          BBS     #DLK$V_CRC,R7,40$                   ; If CRC failed don't complete buff
   14 57    0E   E1     150C  3753          BBC     #DLK$V_TRNLK,R7,20$                 ; If BC don't turn link
            10   AA     1510  3754          BICW    #NO_DS_M_XMTING,-                   ; Start transmitter
          68 A5         1512  3755                  UCB$W_DEVSTS(R5)
       0084 8F   BB     1514  3756          PUSHR   #^M<R2,R7>                          ; Save the register
            51   D4     1518  3757          CLRL    R1                                  ; Set no status for start xmt
          F72A   30     151A  3758          BSBW    START_TRANSMIT                      ; Start the tranmsit
       0084 8F   BA     151D  3759          POPR    #^M<R2,R7>                          ; Restore the register
         50 50   E9     1521  3760          BLBC    R0,70$                              ; Branch on error
   08 57    0A   E1     1524  3761 20$:     BBC     #DLK$V_XMTCMP,R7,23$                ; If BC no transmit to complete
            52   DD     1528  3762          PUSHL   R2                                  ; Save receive buffer
          00E2   30     152A  3763          BSBW    FINISH_XMT_IO                       ; Else branch to post the I/O
            52 8ED0     152D  3764          POPL    R2                                  ; Restore receive buffer
```

```
                16 57    08    E1  1530  3765 23$:    BBC     #DLK$V_RCVACK,R7,40$             ; If BC then not a data message
                                   1534  3766
                53   18 B4    0F  1534  3767         REMQUE  @NOB$Q_RCVS(R4),R3               ; Remov waiting rcv I/O request
                           02    1D  1538  3768         BVS     25$                              ; VS then no packet to complete
                           3D    11  153A  3769         BRB     FINISH_RCV_IO                    ; If found then finish the I/O
                                   153C  3770
                14 B4      62    0E  153C  3771 25$:    INSQUE  (R2),@NOB$Q_ATTN+4(R4)           ; Else, queue message buffer
                           54    DD  1540  3772 30$:    PUSHL   R4
                         019C    30  1542  3773         BSBW    POKE_USER                        ; Poke the user
                           54  8ED0  1545  3774         POPL    R4
                           04    11  1548  3775         BRB     45$
                                   154A  3776
                24 B4      62    0E  154A  3777 40$:    INSQUE  (R2),@NOB$Q_FREE+4(R4)           ; Return the buff to free list
                   50      01    3C  154E  3778 45$:    MOVZWL  S^#SS$_NORMAL,R0                 ; Set normal return
                                   05  1551  3779         RSB
                                       1552  3780
                                       1552  3781 55$:    SETBIT  #XM$V_ERR_FATAL,UCB$L_DEVDEPEND(R5) ; Set that a fatal error occurd
                                       1557  3782
                                       1557  3783
                                       1557  3784         .IF     DF ERR$$$
                                       1557  3785         incw    ucb$w_xg_prst(r5)                ; inc persitant error count
                                       1557  3786         .ENDC   ;DF ERR$$$
                                       1557  3787
                           16    11  1557  3788 58$:    BRB     65$
                                   C8  1559  3789 60$:    BISL    #<XM$M_ERR_START!-               ; Assume the trib error occurrd
                                       155A  3790                 XM$M_ERR_TRIB>,-              ; because a STRT was received
        44 A5    00C00000    8F         155A  3791                 UCB$L_DEVDEPEND(R5)
                    0A 57    07    E1  1561  3792         BBC     #DLK$V_MNTRCV,R7,65$             ; If BC then true
                                       1565  3793         CLRBIT  #XM$V_ERR_START,UCB$L_DEVDEPEND(R5)
                                       156A  3794         SETBIT  #XM$V_ERR_MAINT,UCB$L_DEVDEPEND(R5) ; Else set maint msg rcv'd
                50      20D4    8F  3C  156F  3795 65$:    MOVZWL  #SS$_DEVINACT,R0                 ; Set protocol inactive
                  24 B4     62    0E  1574  3796 70$:    INSQUE  (R2),@NOB$Q_FREE+4(R4)           ; Return the buff to free list
                                   05  1578  3797         RSB
                                       1579  3798
```

```
                                1579  3800          .SBTTL  FINISH_RCV_IO - Finish receive I/O processing
                                1579  3801 ;++
                                1579  3802 ; FINISH_RCV_IO - Finish receive I/O processing
                                1579  3803 ;
                                1579  3804 ; FUNCTIONAL DESCRIPTION:
                                1579  3805 ;
                                1579  3806 ; This routine completes a receive operation that has been matched with a
                                1579  3807 ; message block. After the receive has been completed the message free list is
                                1579  3808 ; filled and a receive is started if needed.
                                1579  3809 ;
                                1579  3810 ;       INPUTS:
                                1579  3811 ;               R2 = message buffer address
                                1579  3812 ;               R3 = I/O packet address
                                1579  3813 ;               R4 = NOB address
                                1579  3814 ;               R5 = UCB address
                                1579  3815 ;
                                1579  3816 ;       IPL = FIPL
                                1579  3817 ;
                                1579  3818 ;       OUTPUTS:
                                1579  3819 ;               R5 = UCB address
                                1579  3820 ;
                                1579  3821 ;       The request is completed via I/O post.
                                1579  3822 ;--
                                1579  3823 FINISH_RCV_IO:                                    ; Finish receive I/O request
    62    48 A2    DE          1579  3824          MOVAL   RCV_T_DATA(R2),(R2)               ; Insert address of the data
    50    0C A2    3C          157D  3825          MOVZWL  RCV_W_MSGSIZ(R2),R0               ; Get size of transfer
             51    D4          1581  3826          CLRL    R1                                ; Assume error
    2C A3    52    D0          1583  3827          MOVL    R2,IRP$L_SVAPTE(R3)               ; Save block address
 04 A2    38 A3    D0          1587  3828          MOVL    IRP$L_MEDIA(R3),4(R2)             ; Insert saved user VA
          42 A5    A0          158C  3829          ADDW    UCB$W_DEVBUFSIZ(R5),-             ; Adjust unit quota
          40 A4                158F  3830                  NOB$W_QUOTA(R4)
    32 A3    50    B1          1591  3831          CMPW    R0,IRP$W_BCNT(R3)                 ; Request larger than actual?
             04    1B          1595  3832          BLEQU   20$                               ; Br if no
    50    32 A3    3C          1597  3833          MOVZWL  IRP$W_BCNT(R3),R0                 ; Set size to min. of two sizes
    32 A3    50    B0          159B  3834 20$:     MOVW    R0,IRP$W_BCNT(R3)                 ; Set size to transfer
 50    50    10    78          159F  3835          ASHL    #16,R0,R0                         ; Set up status
             07    12          15A3  3836          BNEQ    25$                               ; Br if success
 50    0054 8F    B0          15A5  3837          MOVW    #SS$_CTRLERR,R0                   ; Set data path error
             03    11          15AA  3838          BRB     30$
       50    01    B0          15AC  3839 25$:     MOVW    S^#SS$_NORMAL,R0                  ; Set success
    38 A3    50    D0          15AF  3840 30$:     MOVL    R0,IRP$L_MEDIA(R3)                ; Set status and size
                               15B3  3841
    0A A2    13    90          15B3  3842          MOVB    #DYN$C_BUFIO,IRP$B_TYPE(R2)       ; PSI expects this in all buffs
             12    10          15B7  3843          BSBB    IO_DONE                           ; Post the I/O request
          FAB4    30          15B9  3844          BSBW    FICLFREELIST                      ; Return rcv buffer to free q
             05    05          15BC  3845          RSB                                       ; and start next RCV
                               15BD  3846
                               15BD  3847
                               15BD  3848 ; Complete a transfer I/O operation
                               15BD  3849
                               15BD  3850 TRANSMIT_IO_DONE:
             07    E1          15BD  3851          BBC     #XMTQ$V_INTERNAL,-                ; If BC then not an "Interal"
       09 1F A1                15BF  3852                  XMTQ$B_FLAG(R1),IO_DONE           ; IRP, else must dealloc the
                               15C2  3853                                                    ; buffer used to transmit
       50    51    D0          15C2  3854          MOVL    R1,R0                             ; Set to dealloc the buffer
 00000000'GF    16          15C5  3855          JSB     G^COM$DRVDEALMEM                  ; Deall the buffer
                               15CB  3856 IO_DONE:                                         ; Comp a transfer I/O operation
```

```
                44 A5   D0  15CB  3857          MOVL    UCB$L_DEVDEPEND(R5),-           ; Set other info
                3C A3       15CE  3858                  IRP$L_MEDIA+4(R3)
                  07    E1  15D0  3859          BBC     #IRP$V_DIAGBUF,-                ; Br if no diagnostic buffer
             14 2A A3         15D2  3860                  IRP$W_STS(R3),20$
     50   4C B3    08    C1  15D5  3861          ADDL3   #8,@IRP$L_DIAGBUF(R3),R0        ; Addr buffer past start time
   80   00000000'GF    7D  15DA  3862          MOVQ    G^EXE$GQ_SYSTIME,(R0)+          ; Insert stop time
      80    0082 C5    3C  15E1  3863          MOVZWL  UCB$W_ERRCNT(R5),(R0)+          ; Insert error counter
              00F2     30  15E6  3864          BSBW    NO$REGDUMP
        00000000'GF    17  15E9  3865 20$:     JMP     G^COM$POST                      ; Post the I/O
                           15EF  3866
```

```
                     15EF  3868
                     15EF  3869          .SBTTL  TRANSMIT_DONE - Transmit completion routine
                     15EF  3870 ;++
                     15EF  3871 ; TRANSMIT_DONE - Transmit completion routine
                     15EF  3872 ;
                     15EF  3873 ; FUNCTIONAL DESCRIPTION:
                     15EF  3874 ;
                     15EF  3875 ; This routine is called when a transmit buffer needs completion. If the
                     15EF  3876 ; transmit buffer completed was a protocol control buffer then nothing
                     15EF  3877 ; happens with the buffer. In fact it should be part of an the NOB
                     15EF  3878 ; structure . If the transmit was a data buffer, the routine
                     15EF  3879 ; calls the protocol to deal with the buffer. If the protocol notifies the
                     15EF  3880 ; driver that it has XMT's to send to I/O completion, the driver pulls these
                     15EF  3881 ; buffers off the complete queue and sends them off to complete via COM$POST.
                     15EF  3882 ;
                     15EF  3883 ;       INPUTS:
                     15EF  3884 ;               R2 = Address of buffer to complete
                     15EF  3885 ;               R3 = If error then contains the error from the device
                     15EF  3886 ;               R4 = NOB address
                     15EF  3887 ;               R5 = UCB address
                     15EF  3888 ;
                     15EF  3889 ;       IPL = FIPL
                     15EF  3890 ;
                     15EF  3891 ;       OUTPUTS:
                     15EF  3892 ;               R4,R5 are preserved
                     15EF  3893 ;
                     15EF  3894 ;--
                     15EF  3895 TRANSMIT_DONE:
        58   52   D0  15EF  3896          MOVL    R2,R8                           ; Put buffer addr into R8
             30   BB  15F2  3897          PUSHR   #^M<R4,R5>                      ; Save registers
    55    0C   A4   D0  15F4  3898          MOVL    NOB$A_PRO_BUFFER(R4),R5         ; Set protocol buffer address
        56   02   9A  15F8  3899          MOVZBL  #DLK$C_XMTMSG,R6                ; Set up to put on RTOQ if the
        57   01   9A  15FB  3900          MOVZBL  #DLK$M_MSGSENT,R7               ; msg needs to be timed out
             59   D4  15FE  3901          CLRL    R9
           E9FD'  30  1600  3902          BSBW    DDCMP                           ; Branch to protocol
             30   BA  1603  3903          POPR    #^M<R4,R5>                      ; Restore registers
    02 57    0A   E1  1605  3904          BBC     #DLK$V_XMTCMP,R7,30$            ; If BC then no XMT's to compl
             04   10  1609  3905          BSBB    FINISH_XMT_IO                   ; Else complete the XMT
        50   01   3C  160B  3906 30$:      MOVZWL  S^#SS$_NORMAL,R0                ; Set normal return
                  05  160F  3907          RSB
                      160F  3908
                      160F  3909 FINISH_XMT_IO:
          03C0   8F   BB  160F  3910          PUSHR   #^M<R6,R7,R8,R9>                ; Save registers
    56    0C   A4   D0  1613  3911          MOVL    NOB$A_PRO_BUFFER(R4),R6         ; Get protocol buffer address
    58    28   B6   0F  1617  3912 5$:       REMQUE  @TF$Q_CMPQ(R6),R8               ; Get the next entry on queue
             27   1D  161B  3913          BVS     20$                             ; If VS then branch to finish
    53    0C   A8   D0  161D  3914          MOVL    XMTQ$L_IRP(R8),R3               ; Get IRP associated with XMT
    50    1A   A8   3C  1621  3915          MOVZWL  XMTQ$W_MSGSIZE(R8),R0           ; Get transfer size
        50   06   A2  1625  3916          SUBW2   #NO$C_HEADER_LEN,R0             ; Subtract out the protocol hdr
50      50   10   78  1628  3917          ASHL    #16,R0,R0                       ; Set up status
             07   12  162C  3918          BNEQ    10$                             ; Br if data transmitted
50    0054   8F   B0  162E  3919          MOVW    #SS$_CTRLERR,R0                 ; Set device error
             03   11  1633  3920          BRB     15$
        50   01   B0  1635  3921 10$:      MOVW    #SS$_NORMAL,R0                  ; Assume success
    38  A3   50   D0  1638  3922 15$:      MOVL    R0,IRP$L_MEDIA(R3)
        51   58   D0  163C  3923          MOVL    R8,R1                           ; Set XMTQ buff address
           FF7B   30  163F  3924          BSBW    TRANSMIT_IO_DONE                ; Complete the IO
```

```
                D3   11  1642  3925          BRB     5$                              ; Get next to complete
50              01   3C  1644  3926 20$:     MOVZWL  S^#SS$_NORMAL,R0
03C0            8F   BA  1647  3927          POPR    #^M<R6,R7,R8,R9>                ; Restore registers
                     05  164B  3928          RSB
```

```
                     164C  3930
                     164C  3931 ;++
                     164C  3932 ; DEVTIMER - Device wakeup routine
                     164C  3933 ;
                     164C  3934 ; Functional Description:
                     164C  3935 ;
                     164C  3936 ; This routine is called when the protocol timer every second when the
                     164C  3937 ; protocol timer ticks to start any transmits the protocol may
                     164C  3938 ; have queued as a result of the timer going off.
                     164C  3939 ;
                     164C  3940 ;       INPUTS:
                     164C  3941 ;               R1 = Error status from protocol specific timer routine
                     164C  3942 ;               R5 = Address of the UCB
                     164C  3943 ;
                     164C  3944 ;
                     164C  3945 ;       OUTPUTS:
                     164C  3946 ;               R0 = Status of line is LBS then line is up
                     164C  3947 ;                                       LBC then line is down
                     164C  3948 ;               On error the line is shut down, however the UCB is not
                     164C  3949 ;               deallocated
                     164C  3950 ;
                     164C  3951 ;--
                     164C  3952          .ENABL  LSB
                     164C  3953 DEVTIMER:
      54   00A8 C5   D0  164C  3954          MOVL    UCB$L_NO_BUFFER(R5),R4          ; Get NOB address
   7A 44 A5    10   E0  1651  3955          BBS     #XM$V_ERR_FATAL,UCB$L_DEVDEPEND(R5),30$ ; If BS then fatal err
      76 51    01   E0  1656  3956          BBS     #DLK$V_PRSTERR,R1,30$           ; If BS then psrt err shut down
                     165A  3957                                                  ; the device
                     165A  3958
                     165A  3959 ; Before we do anything else we should check to see if the xmts and/or
                     165A  3960 ; receives in progress timed out.
                     165A  3961
                     165A  3962          DSBINT  UCB$B_DIPL(R5)
   52   00000000'GF   D0  1661  3963          MOVL    G^EXE$GL_ABSTIM,R2
      19 68 A5    0B   E1  1668  3964          BBC     #NO_DS_V_XMT_TIME,UCB$W_DEVSTS(R5),5$ ; If clear time not valid
         3C A4    52   D1  166D  3965          CMPL    R2,NOB$L_XMT_DUETIM(R4)         ; If LEQ time not up
                  13   15  1671  3966          BLEQ    5$
                009B   30  1673  3967          BSBW    TIMEOUT                         ; Else timeout message
      50    00F4 C5   9A  1676  3968          MOVZBL  UCB$W_TT_SPEED(R5),R0           ; Reset due time
   50    EA84 CF40   3C  167B  3969          MOVZWL  DUETIM_TABLE[R0],R0
   3C A4    52   50   A1  1681  3970          ADDW3   R0,R2,NOB$L_XMT_DUETIM(R4)
      20 68 A5    0C   E1  1686  3971 5$:     BBC     #NO_DS_V_RCV_TIME,UCB$W_DEVSTS(R5),8$ ; If clear time not valid
         38 A4    50   D1  168B  3972          CMPL    R0,NOB$L_RCV_DUETIM(R4)         ; If LEQ time not up
                  1A   15  168F  3973          BLEQ    8$
                007D   30  1691  3974          BSBW    TIMEOUT                         ; Else timeout message
      50    00F5 C5   9A  1694  3975          MOVZBL  UCB$W_TT_SPEED+1(R5),R0         ; Reset due time
                  05   12  1699  3976          BNEQ    6$
      50    00F4 C5   9A  169B  3977          MOVZBL  UCB$W_TT_SPEED(R5),R0
   50    EA5F CF40   3C  16A0  3978 6$:     MOVZWL  DUETIM_TABLE[R0],R0
   38 A4    52   50   A1  16A6  3979          ADDW3   R0,R2,NOB$L_RCV_DUETIM(R4)
                     16AB  3980 8$:     ENBINT
            50   01   3C  16AE  3981          MOVZWL  S^#SS$_NORMAL,R0                ; Assume everything ok
                     16B1  3982
                     16B1  3983 ; If the protocol timer has expired then clear the XMTING flag so that
                     16B1  3984 ; the control message which must be sent can be. This is true when the
                     16B1  3985 ; device is running half duplex. If the message with the select flag is dropped
                     16B1  3986 ; then we must be sure that we can send anouthe message with a select flag.
```

```
                        16B1  3987
                        16B1  3988          BBC     #DLK$V_TMREXPD,R1,10$
                        16B5  3989          CLRBIT  #NO_DS_V_XMTING,UCB$W_DEVSTS(R5)
                        16BA  3990 10$:     PUSHL   R1
                        16BC  3991          BSBW    START_TRANSMIT                  ; Else try to start a transmit
                        16BF  3992          POPL    R1
                        16C2  3993          BLBC    R0,30$                          ; If LBC then fatal error
                        16C5  3994          BICW    #DLK$M_TMREXPD,R1               ; Clear timer expired
                        16CA  3995 20$:     RSB
                        16CB  3996
                        16CB  3997 ;&&&&&&&&&&&&&& WHAT TO DO &&&&&&&&&&&&
                        16CB  3998
                        16CB  3999 25$:     SETBIT  #XM$V_STS_DISC,UCB$L_DEVDEPEND(R5)  ; Set modem disconnect sts
                        16D0  4000
                        16D0  4001 ; TFB/GFB blocks are deallocated on shutdown. Make sure R0 is
                        16D0  4002 ; clear so that the timer routine does not try to access them.
                        16D0  4003
                        16D0  4004 30$:     SETBIT  #XM$V_ERR_FATAL,UCB$L_DEVDEPEND(R5) ; Set fatal error
                        16D5  4005          BSBW    SHUTDOWN_LINE                   ; Shut down the device
                        16D8  4006          CLRL    R0
                        16DA  4007          RSB
                        16DB  4008          .DSABL  LSB
                        16DB  4009
```

```
                     16DB  4011          .SBTTL  NO$REGDUMP - Error log and diagnostics register dump
                     16DB  4012 ;++
                     16DB  4013 ; NO$REGDUMP -Diagnostics register dump routine
                     16DB  4014 ;
                     16DB  4015 ; Functional description:
                     16DB  4016 ;
                     16DB  4017 ; This routine is used to return ?????
                     16DB  4018 ;
                     16DB  4019 ;       INPUTS:
                     16DB  4020 ;               R0 = ADDRESS OF THE BUFFER
                     16DB  4021 ;               R5 = UCB ADDRESS OF THE UNIT
                     16DB  4022 ;
                     16DB  4023 ;       OUTPUTS:
                     16DB  4024 ;               R0,R1 ARE USED
                     16DB  4025 ;               R5 = UCB ADDRESS OF THE UNIT
                     16DB  4026 ;--
                     16DB  4027 NO$REGDUMP:
          0B   E1    16DB  4028          BBC     #XM$V_STS_ACTIVE,-      ; BR if not active
00 44 A5             16DD  4029                  UCB$L_DEVDEPEND(R5),20$
                05   16E0  4030 20$:     RSB
                     16E1  4031
```

```
                                16E1  4033          .SBTTL  Poke user process on attention condition
                                16E1  4034
                                16E1  4035 ;++
                                16E1  4036 ; POKE_USER - Poke user process on attention condition
                                16E1  4037 ;
                                16E1  4038 ; FUNCTIONAL DESCRIPTION:
                                16E1  4039 ;
                                16E1  4040 ; This routine is used when data is avaliable or the unit is shutdown.
                                16E1  4041 ; The action is to declare the ast's and send a message to the assoc. mailbox.
                                16E1  4042 ;
                                16E1  4043 ;       INPUTS:
                                16E1  4044 ;               R4 = Message type -- 0 if no message
                                16E1  4045 ;               R5 = Unit UCB address
                                16E1  4046 ;
                                16E1  4047 ;       OUTPUTS:
                                16E1  4048 ;               R0 = Low bit clear only if user is not notified
                                16E1  4049 ;               R5 = UCB ADDRESS
                                16E1  4050 ;--
                                16E1  4051 POKE_USER:                                           ; POKE USER
                                16E1  4052          DSBINT  UCB$B_FIPL(R5)                      ; Sync to Fork
                          7E    16E8  4053          CLRL    -(SP)                               ; Assume failure
 51    00AC                C5  16EA  4054          MOVAB   UCB$L_NO_AST(R5),R1                 ; Get AST listhead
                          61    16EF  4055          TSTL    (R1)                                ; Empty ?
                          17    16F1  4056          BEQL    17$                                 ; If so, branch
                          6E    16F3  4057          INCL    (SP)                                ; Indicate success
       54                 51    16F5  4058          MOVL    R1,R4                               ; Copy list head address
       51                 61    16F8  4059 10$:     MOVL    (R1),R1                             ; Address a block
                          07    16FB  4060          BEQL    15$                                 ; If EQL then done
 1C A1    44                A5  16FD  4061          MOVL    UCB$L_DEVDEPEND(R5),^X01C(R1)       ; Change param
                          F4    1702  4062          BRB     10$
  00000000'GF               16  1704  4063 15$:     JSB     G^COM$DELATTNAST                    ; Deliver AST'S
                       50 8ED0  170A  4064 17$:     POPL    R0                                  ; Set status
                                170D  4065          ENBINT                                      ; Enable interrupts
                             05 1710  4066          RSB
```

```
                     1711  4068          .SBTTL  TIMEOUT - TIMEOUT
                     1711  4069 ;++
                     1711  4070 ; TIMEOUT - Timeout
                     1711  4071 ;
                     1711  4072 ; FUNCTIONAL DESCRIPTION:
                     1711  4073 ;
                     1711  4074 ; This routine is entered on device timeout. The action is to shut
                     1711  4075 ; the unit down.
                     1711  4076 ;
                     1711  4077 ;       INPUTS:
                     1711  4078 ;               R5 = UCB address
                     1711  4079 ;
                     1711  4080 ;       OUTPUTS:
                     1711  4081 ;               NONE
                     1711  4082 ;
                     1711  4083 ;       IPL = Device IPL
                     1711  4084 ;--
                     1711  4085 TIMEOUT:                                       ; TIMEOUT OR POWERFAIL
          0B   E1    1711  4086          BBC     #XM$V_STS_ACTIVE,-            ; Br BC device is not active
08 44 A5             1713  4087                  UCB$L_DEVDEPEND(R5),20$
          05   E1    1716  4088 10$:     BBC     #UCB$V_POWER,-                ; Br unless powerfail
04 64 A5             1718  4089                  UCB$W_STS(R5),30$
      FCB2   30      171B  4090          BSBW    SCHED_FORK_POWERFAIL          ; Create the error fork process
             05      171E  4091 20$:     RSB
                     171F  4092
                     171F  4093 ; Because some devices are flaky when driven as hard as DECnet drives the line,
                     171F  4094 ; we have added code here to clear the device in case it gets stuck. A side
                     171F  4095 ; effect of clearing the device is that any transmit data it had outstanding
                     171F  4096 ; gets lost. In order to handle that condition we will transmit enough pads
                     171F  4097 ; to make up for the size of the message that we feel has been dropped. This
                     171F  4098 ; is decided by the state the device is in and the type of message is was
                     171F  4099 ; transmitting.
                     171F  4100 ;
                     171F  4101 ; There is another case we must worry about here and that is if the reply
                     171F  4102 ; timeout timer has gone off. In that case the protocol takes the xmt away
                     171F  4103 ; from the device, so that messages are not retransmitted out of order.
                     171F  4104 ; Thus we will check for this condition (the NOB$L_XMT_INPR field is clear)
                     171F  4105 ; and if it occurs then we will send out pads to fill the messaege length,
                     171F  4106 ; with our best guess. We also put the ctrl msg address in the inpr field
                     171F  4107 ; to fake the xmter into thinking that the message is still active.
                     171F  4108
      0082 C5    B6  171F  4109 30$:     INCW    UCB$W_ERRCNT(R5)              ; Incr the error counter
           3E    BB  1723  4110          PUSHR   #^M<R1,R2,R3,R4,R5>
52    0118 C5    D0  1725  4111          MOVL    UCB$L_TT_PORT(R5),R2          ; Get the port vector
        20 B2    16  172A  4112          JSB     @PORT_ABORT(R2)               ; Reset the device
        24 B2    16  172D  4113          JSB     @PORT_RESUME(R2)
54    00A8 C5    D0  1730  4114          MOVL    UCB$L_NO_BUFFER(R5),R4        ; Get the no buffer
           53    D4  1735  4115          CLRL    R3
  51    42 A4    9A  1737  4116          MOVZBL  NOB$B_XSTATE(R4),R1           ; Get old state
  42 A4    0A    90  173B  4117          MOVB    #NO$C_XMTERR,NOB$B_XSTATE(R4) ; Set new state
                     173F  4118          $DISPATCH R1,TYPE=B,-
                     173F  4119          <-      ;function               ;action
                     173F  4120                  <NO$C_EMPTY             40$>,-
                     173F  4121                  <NO$C_HEADER            TIMEOUT_HEADER>,-
                     173F  4122                  <NO$C_HCRC              TIMEOUT_HCRC>,-
                     173F  4123                  <NO$C_DATA              TIMEOUT_DATA>,-
                     173F  4124                  <NO$C_DCRC              TIMEOUT_DCRC>,-
```

```
                                173F  4125                    <NO$C_PADS              TIMEOUT_PADS>,-
                                173F  4126                    <NO$C_XMTERR            TIMEOUT_XMTERR>,-
                                173F  4127            >
                                1759  4128
                  42 A4   94    1759  4129 40$:       CLRB    NOB$B_XSTATE(R4)                ; Reset state
                     3E   BA    175C  4130            POPR    #^M<R1,R2,R3,R4,R5>
                     05   05    175E  4131            RSB
                                175F  4132
                                175F  4133 TIMEOUT_HEADER:
          53    42 A5   0A    A1    175F  4134            ADDW3   #10,UCB$W_DEVBUFSIZ(R5),R3       ; Assume xmt is gone
                51    34 A4   D0    1764  4135            MOVL    NOB$L_XMT_INPR(R4),R1
                     57   13    1768  4136            BEQL    TIMEOUT_END
             07 1F A1    02    E0    176A  4137            BBS     #XMTQ$V_CONTROL,XMTQ$B_FLAG(R1),10$
                                176F  4138
                                176F  4139 ; Set send field high enough to cover xmting of header, hcrc, data, and dcrc.
                                176F  4140
          53    1A A1    04    A1    176F  4141            ADDW3   #4,XMTQ$W_MSGSIZE(R1),R3
                     4B   11    1774  4142            BRB     TIMEOUT_END
                                1776  4143
                                1776  4144 ; Set send length field high enough to cover xmitng of header and hcrc
                                1776  4145
                  53   06    B0    1776  4146 10$:       MOVW    #NO$C_HEADER_LEN,R3
                     46   11    1779  4147            BRB     TIMEOUT_END
                                177B  4148
                                177B  4149 TIMEOUT_HCRC:
          53    42 A5   04    A1    177B  4150            ADDW3   #4,UCB$W_DEVBUFSIZ(R5),R3       ; Assume xmt is gone
                51    34 A4   D0    1780  4151            MOVL    NOB$L_XMT_INPR(R4),R1
                     3B   13    1784  4152            BEQL    TIMEOUT_END
             0A 1F A1    02    E0    1786  4153            BBS     #XMTQ$V_CONTROL,XMTQ$B_FLAG(R1),10$
                                178B  4154
                                178B  4155 ; Set send field high enough to cover xmting of hcrc, data, and dcrc.
                                178B  4156
          53    1A A1    04    A1    178B  4157            ADDW3   #4,XMTQ$W_MSGSIZE(R1),R3
                53   06    A2    1790  4158            SUBW2   #NO$C_HEADER_LEN,R3             ; XMTQ$W_MSGSIZ includes HEADER size
                     2C   11    1793  4159            BRB     TIMEOUT_END
                                1795  4160
                                1795  4161 ; Set send length field high enough to cover xmitng of hcrc
                                1795  4162
                  53   02    B0    1795  4163 10$:       MOVW    #2,R3
                     27   11    1798  4164            BRB     TIMEOUT_END
                                179A  4165
                                179A  4166 TIMEOUT_DATA:
          53    42 A5   02    A1    179A  4167            ADDW3   #2,UCB$W_DEVBUFSIZ(R5),R3       ; Assume xmt is gone
                51    34 A4   D0    179F  4168            MOVL    NOB$L_XMT_INPR(R4),R1
                     1C   13    17A3  4169            BEQL    TIMEOUT_END
                                17A5  4170
                                17A5  4171 ; Set send field high enough to cover xmting of data, and dcrc.
                                17A5  4172
          53    1A A1    02    A1    17A5  4173            ADDW3   #2,XMTQ$W_MSGSIZE(R1),R3
                53   06    A2    17AA  4174            SUBW2   #NO$C_HEADER_LEN,R3             ; XMTQ$W_MSGSIZ includes HEADER size
                     12   11    17AD  4175            BRB     TIMEOUT_END
                                17AF  4176
                                17AF  4177 TIMEOUT_DCRC:
                  53   02    B0    17AF  4178            MOVW    #2,R3                           ; Set the field to send DCRC
                     0D   11    17B2  4179            BRB     TIMEOUT_END
                                17B4  4180
                                17B4  4181 TIMEOUT_PADS:
```

```
                53   02    B0  17B4  4182          MOVW    #2,R3                                   ; Set the field to send DCRC
             42 A4   05    90  17B7  4183          MOVB    #NOS$C_PADS,NOB$B_XSTATE(R4)            ; Set to finish message
                     04    11  17BB  4184          BRB     TIMEOUT_END
                               17BD  4185
                               17BD  4186 TIMEOUT_XMTERR:
             53   46 A4    B0  17BD  4187          MOVW    NOB$W_XMTERR_SIZE(R4),R3
                               17C1  4188          ;BRB    TIMEOUT_END
                               17C1  4189
                               17C1  4190 TIMEOUT_END:
                               17C1  4191
                   34 A4    D5  17C1  4192          TSTL    NOB$L_XMT_INPR(R4)                      ; If NEQ then message avaialable
                      06    12  17C4  4193          BNEQ    10$
          34 A4    0080 C4    DE  17C6  4194          MOVAL   NOB$Z_CTL_MSG(R4),NOB$L_XMT_INPR(R4) ; Else set to fake xmter
               46 A4     53    B0  17CC  4195 10$:     MOVW    R3,NOB$W_XMTERR_SIZE(R4)                ; Set size to xmt
          011C C5   44 A4    DE  17D0  4196          MOVAL   NOB$W_PADS(R4),UCB$L_TT_OUTADR(R5) ; set to send pads
             0120 C5   02    9B  17D6  4197          MOVZBW  #2,UCB$W_TT_OUTLEN(R5)                  ; set length
             010B C5   01    8E  17DB  4198          MNEGB   #1,UCB$B_TT_OUTYPE(R5)                  ; Set to send a burst
                       3E    BA  17E0  4199          POPR    #^M<R1,R2,R3,R4,R5>
                       05    17E2  4200          RSB
                               17E3  4201
```

```
                                        .SBTTL  NO$CANCEL - Cancel I/O rotuine
                                ;++
                                ; NO$CANCEL - Cancels all I/O in progress
                                ;
                                ; If the NO buffer has not been alloacted that means the device is
                                ; not running, set normal return to decrement the reference count
                                ; and return. This conditior can happen when the line gets switched
                                ; from a static terminal port to a static ddcmp port.
                                ; By looking at the channel number saved at line startup time, cancel
                                ; determines which entity (line or circuit) to shut down. Only on the
                                ; last deassign to the channels does the buffer allocated for the
                                ; TT UCB's extension get deallocated.
                                ;
                                ;       INPUTS:
                                ;               R2 = channel number
                                ;               R3 = current IRP address
                                ;               R4 = PCB address
                                ;               R5 = UCB address
                                ;               R8 = Cancel reason code (zero vanilla flavored cancel)
                                ;
                                ;       IPL = FIPL
                                ;
                                ;       OUTPUTS:
                                ;               R0 - R3 are destroyed
                                ;
                                ;--
                                NO$CANCEL:
 50   00A8 C5    D0  17E3  4230          MOVL    UCB$L_NO_BUFFER(R5),R0          ; Get NOB addess
        23    13  17E8  4231          BEQL    30$                             ; If EQL device not active
60 A4   58 A0    D1  17EA  4232          CMPL    NOB$L_PID(R0),PCB$L_PID(R4)     ; If not starters PID then
        1C    12  17EF  4233          BNEQ    30$                             ; do not process cancel
   52   5C A0    B1  17F1  4234          CMPW    NOB$W_CHANL(R0),R2              ; If EQL match on line channel
        05    13  17F5  4235          BEQL    10$
      00E6    30  17F7  4236          BSBW    SHUTDOWN_CIRCUIT                ; Else shutdown the circuit
        03    11  17FA  4237          BRB     15$
      004F    30  17FC  4238 10$:     BSBW    SHUTDOWN_LINE
        50    DD  17FF  4239 15$:     PUSHL   R0
      5C A5    B5  1801  4240          TSTW    UCB$W_REFC(R5)                  ; If reference count is nonzero
        03    12  1804  4241          BNEQ    20$                             ; do not deallocate nob or disc port
      0008    30  1806  4242          BSBW    CLEAR_NO_BUFFER                 ; Else do
        50  8ED0  1809  4243 20$:     POPL    R0                              ; Set status from shutdown
            05  180C  4244          RSB
                180D  4245
   50   01    3C  180D  4246 30$:     MOVZWL  #SS$_NORMAL,R0
            05  1810  4247          RSB
```

```
                                1811  4249          .SBTTL  CLEAR_NO_BUFFER - Clear NO buffer and disconnect the port
                                1811  4250 ;++
                                1811  4251 ; CLEAR_NO_BUFFER - Clear NO buffer and disconnect the port
                                1811  4252 ;
                                1811  4253 ; This routine is called on the last deassign to the device. It deallocates
                                1811  4254 ; the NO buffer which was used as an extension to the devices UCB. And
                                1811  4255 ; it disconnects with out deallocating the UCB from the port.
                                1811  4256 ;
                                1811  4257 ;       INPUTS:
                                1811  4258 ;               R4 = PCB address
                                1811  4259 ;               R5 = UCB address
                                1811  4260 ;
                                1811  4261 ;       IPL = FIPL
                                1811  4262 ;
                                1811  4263 ;       OUTPUTS:
                                1811  4264 ;               All registers are preserved.
                                1811  4265 ;
                                1811  4266 ;--
                                1811  4267 CLEAR_NO_BUFFER:
     00FE 8F    BB              1811  4268          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7>
50   00A8 C5    D0              1815  4269          MOVL    UCB$L_NO_BUFFER(R5),R0          ; If EQL buffer no allocated
          2D    13              181A  4270          BEQL    10$
     00A8 C5    D4              181C  4271          CLRL    UCB$L_NO_BUFFER(R5)             ; Set buffer no longer available
  57   58 A0    D0              1820  4272          MOVL    NOB$L_PID(R0),R7                ; Save starters' PID
  52   08 A0    3C              1824  4273          MOVZWL  UCB$W_SIZE(R0),R2               ; get size of buffer
          34    BB              1828  4274          PUSHR   #^M<R2,R4,R5>
00000000'GF     16              182A  4275          JSB     G^COM$DRVDEALMEM                ; Deallocate the block
          34    BA              1830  4276          POPR    #^M<R2,R4,R5>
                                1832  4277
                                1832  4278 ; Call the port driver to disconnect the port. Do not delete the UCB.
                                1832  4279
51   0118 C5    D0              1832  4280          MOVL    UCB$L_TT_PORT(R5),R1
       04 B1    16              1837  4281          JSB     @PORT_DISCONNECT(R1)
  60 A4   57    D1              183A  4282          CMPL    R7,PCB$L_PID(R4)                ; Current PID same as starter's PID
          09    12              183E  4283          BNEQ    10$                             ; NEQ then don't restore quota
50   0080 C4    D0              1840  4284          MOVL    PCB$L_JIB(R4),R0                ; Get JIB
  20 A0   52    C0              1845  4285          ADDL2   R2,JIB$L_BYTCNT(R0)             ; Restore the byte count
                                1849  4286
     00FE 8F    BA              1849  4287 10$:     POPR    #^M<R1,R2,R3,R4,R5,R6,R7>
                05              184D  4288          RSB
                                184E  4289
```

```
                                184E  4291
                                184E  4292          .SBTTL SHUTDOWN - Shut down unit, device and/or protocol
                                184E  4293 ;++
                                184E  4294 ; SHUTDOWN_LINE - Shut down unit
                                184E  4295 ;
                                184E  4296 ; FUNCTIONAL DESCRIPTION:
                                184E  4297 ;
                                184E  4298 ; This routine is used to shut down the unit as a result of a
                                184E  4299 ; setmode shutdown on the controller or a fatal error. The mapping
                                184E  4300 ; registers are returned, a call is made to shutdown the circuit; and
                                184E  4301 ; all quotas are returned. If this was the last deassign of a channel
                                184E  4302 ; to the device then the NOB is deallocated.
                                184E  4303 ;
                                184E  4304 ;       INPUTS:
                                184E  4305 ;               R5 = UCB address
                                184E  4306 ;
                                184E  4307 ;       For SHUTDOWN_LINE_ALT
                                184E  4308 ;
                                184E  4309 ;               R1 = 0 if DTR can be dropped on shut down
                                184E  4310 ;                    1 if DTR can not be dropped on shut down
                                184E  4311 ;
                                184E  4312 ;       OUTPUTS:
                                184E  4313 ;               R5 = UCB address
                                184E  4314 ;               R6 - R9 are preserved
                                184E  4315 ;               R0-R3 are destroyed.
                                184E  4316 ;--
                                184E  4317 SHUTDOWN_LINE:                                       ; Shut down unit
                  51    01    D0  184E  4318          MOVL    #1,R1                             ; Do not dealloacte NOB
                                1851  4319 SHUTDOWN_LINE_ALT:
                        06    E1  1851  4320          BBC     #NO_DS_V_INITED,-
                05 68   A5        1853  4321                  UCB$W_DEVSTS(R5),5$                ; BR if not inited
                        04    E0  1856  4322          BBS     #UCB$V_ONLINE,-
                06 64   A5        1858  4323                  UCB$W_STS(R5),10$                  ; Br if online
              50  20D4  8F    3C  185B  4324 5$:      MOVZWL  #SS$_DEVINACT,R0                   ; Set status
                        05    00  1860  4325          RSB                                       ; ... and return
                                1861  4326
                03DC    8F    BB  1861  4327 10$:     PUSHR   #^M<R2,R3,R4,R6,R7,R8,R9>          ; Save the registers
                                1865  4328
              64 A5     21    AA  1865  4329          BICW    #UCB$M_POWER!-
                                1869  4330                  UCB$M_TIM,UCB$W_STS(R5)            ; Reset device status
                                1869  4331          BICW    #^C<NO_DS_M_XMTING!-               ; Clear all but xmt off
                                186A  4332                      NO_DS_M_RCVING!-               ; rcv off
                                186A  4333                      NO_DS_M_ILOOP_SUP>,-           ; internal loop
          68 A5     FECF    8F      186A  4334                  UCB$W_DEVSTS(R5)
                                186F  4335
                                186F  4336
                                186F  4337 ; Shut down the circuit
                                186F  4338
                        51    DD  186F  4339          PUSHL   R1                                 ; Save NOB deallocate status
                    006C    30  1871  4340          BSBW    SHUTDOWN_CIRCUIT                   ; Shutdown the circuit
              54    00A8    C5  1874  4341          MOVL    UCB$L_NO_BUFFER(R5),R4             ; Get NOB address
                                1879  4342
                                1879  4343 ; Stop the DDCMP timer. First set up registers with params for call
                                1879  4344 ; then make the call to the protocol.
                                1879  4345
                        30    BB  1879  4346          PUSHR   #^M<R4,R5>
                        56    07  9A  187B  4347          MOVZBL  #DLK$C_STOP_TIMER,R6               ; Set up R6 with the DDCMP comm
```

```
                     57   7C  187E  4348          CLRQ    R7                              ; CLR all registers
                     59   D4  1880  4349          CLRL    R9                              ; Clear R9
               55    0C A4   D0  1882  4350          MOVL    NOB$A_PRO_BUFFER(R4),R5         ; Set up R5 with TF block addr
                   E777'     30  1886  4351          BSBW    DDCMP                           ; Call protocol
                  50   55    D0  1889  4352          MOVL    R5,R0                           ; Need to deallocate protocol buffer
                       30    BA  188C  4353          POPR    #^M<R4,R5>
                                 188E  4354
                                 188E  4355 ; After the timer has been stopped it is ok to reload the driver.
                                 188E  4356
00000000D'EF    04    8A  188E  4357          BICB    #DPT$M_NOUNLOAD,DPT$TAB+DPT$B_FLAGS
                                 1895  4358
                                 1895  4359 ; Deallocate the protocol buffer
                                 1895  4360
                    0C A4   D4  1895  4361          CLRL    NOB$A_PRO_BUFFER(R4)            ; Clear all knowledge of protocol bu
                52  08 A0   3C  1898  4362          MOVZWL  UCB$W_SIZE(R0),R2               ; Get the size
                       34    BB  189C  4363          PUSHR   #^M<R2,R4,R5>
                00000000'GF  16  189E  4364          JSB     G^COM$DRVDEALMEM                ; Dealloc the buffer
                       34    BA  18A4  4365          POPR    #^M<R2,R4,R5>
                                 18A6  4366
                                 18A6  4367
                                 18A6  4368 ; Restore the buffered I/O quota and protocol buffer quota to the starter
                                 18A6  4369
                50   58 A4   3C  18A6  4370          MOVZWL  NOB$L_PID(R4),R0                ; Get pid of last starter
           51   00000000'GF     D0  18AA  4371          MOVL    G^SCH$GL_PCBVEC,R1              ; Address PCB vector
                 50   6140     D0  18B1  4372          MOVL    (R1)[R0],R0                     ; Get PCB of owner
                56   60 A0     D0  18B5  4373          MOVL    PCB$L_PID(R0),R6                ; Save pid of PCB
                57   58 A4     D0  18B9  4374          MOVL    NOB$L_PID(R4),R7                ; Save PID of starter
                51   40 A4     3C  18BD  4375          MOVZWL  NOB$W_QUOTA(R4),R1              ; Get quota
                    40 A4      B4  18C1  4376          CLRW    NOB$W_QUOTA(R4)                 ; Clear quota returned
                52   51        C0  18C4  4377          ADDL    R1,R2                           ; Get bytes quota to return
                     51   8ED0   D0  18C7  4378          POPL    R1                              ; Restore drop DTR status
                57   56        D1  18CA  4379          CMPL    R6,R7                           ; Do the PIDs match
                     09        12  18CD  4380          BNEQ    20$                             ; If NEQ no don't return quota
           50   0080 C0     D0  18CF  4381          MOVL    PCB$L_JIB(R0),R0                ; Get JIB
             20 A0   52     C0  18D4  4382          ADDL    R2,JIB$L_BYTCNT(R0)             ; Return byte count quota
                50   01     3C  18D8  4383 20$:     MOVZWL  S^#SS$_NORMAL,R0                ; Set status
                03DC 8F     BA  18DB  4384          POPR    #^M<R2,R3,R4,R6,R7,R8,R9>       ; Save the registers
                             05  18DF  4385          RSB
                                 18E0  4386
                                 18E0  4387
```

```
                     18E0  4389
                     18E0  4390 ;++
                     18E0  4391 ; SHUTDOWN_CIRCUIT - Shut down device and protocol
                     18E0  4392 ;
                     18E0  4393 ; FUNCTIONAL DESCRIPTION:
                     18E0  4394 ;
                     18E0  4395 ; This routine is used to shut down the circuit as a result of a setmode
                     18E0  4396 ; shutdown on the tributary or on the controller or by a fatal error on
                     18E0  4397 ; the device. The routines frees allocated blocks; completes IRP's with
                     18E0  4398 ; the active bit clear; and halts the protocol.
                     18E0  4399 ;
                     18E0  4400 ;       INPUTS:
                     18E0  4401 ;               R5 = UCB address
                     18E0  4402 ;
                     18E0  4403 ;       OUTPUTS:
                     18E0  4404 ;               R5 = UCB address
                     18E0  4405 ;               R6 - R9 are preserved
                     18E0  4406 ;               R0-R3 are destroyed.
                     18E0  4407 ;--
                     18E0  4408 SHUTDOWN_CIRCUIT:
 06 44 A5    0B   E4 18E0  4409         BBSC    #XM$V_STS_ACTIVE,UCB$L_DEVDEPEND(R5),10$ ; If BS then trib is active
  50   20D4 8F    3C 18E5  4410         MOVZWL  #SS$_DEVINACT,R0                 ; Else set status and return
                  05 18EA  4411         RSB
                     18EB  4412
       03DC 8F    BB 18EB  4413 10$:     PUSHR   #^M<R2,R3,R4,R6,R7,R8,R9>       ; Save the registers
68 A5  0430 8F    A8 18EF  4414         BISW    #<NO_DS_M_XMTING!-               ; Set the XMTer and
                     18F5  4415                   NO_DS_M_RCVING!-               ; RCVer off
                     18F5  4416                   NO_DS_M_SHUTDOWN>,UCB$W_DEVSTS(R5) ; and set device is shuting dow
                     18F5  4417
                     18F5  4418 ; What we are about to do is in direct violation of the DDCMP standard.
                     18F5  4419 ; But as it is we can not think of what else to do. If anyone out there
                     18F5  4420 ; has a better suggestion please feel free to send it to us. We are however
                     18F5  4421 ; going to abort the receiver and tranmitter in the next few instructions.
                     18F5  4422 ; The ddcmp standard says that the transmitter should not be aborted, but
                     18F5  4423 ; in order to make this true under all conditions we would have to
                     18F5  4424 ; wait at this spot for as much as 90 seconds.
                     18F5  4425
                     18F5  4426         DSBINT  UCB$B_DIPL(R5)
52   0118 C5    D0 18FC  4427         MOVL    UCB$L_TT_PORT(R5),R2             ; Get the port vector
       20 B2    16 1901  4428         JSB     @PORT_ABORT(R2)                  ; Reset the device
       24 B2    16 1904  4429         JSB     @PORT_RESUME(R2)
54   00A8 C5    D0 1907  4430         MOVL    UCB$L_NO_BUFFER(R5),R4           ; Get NOB address
       42 A4    94 190C  4431         CLRB    NOB$B_XSTATE(R4)                 ; Reset transmitter state
        F7BB    30 190F  4432         BSBW    START_RECEIVE                    ; Reset receiver state
                   1912  4433
                   1912  4434
                   1912  4435 ; Clear any receives or transmits currently inprogress
                   1912  4436
56     30 A4    D0 1912  4437         MOVL    NOB$L_RCV_INPR(R4),R6            ; If EQL then on receive pending
          07    13 1916  4438         BEQL    20$
       30 A4    D4 1918  4439         CLRL    NOB$L_RCV_INPR(R4)               ; Clear receive from inprogress
24 B4     66    0E 191B  4440         INSQUE  (R6),@NOB$Q_FREE+4(R4)           ; Queue to free queue
51     34 A4    D0 191F  4441 20$:     MOVL    NOB$L_XMT_INPR(R4),R1            ; Comp XMT's on inprogress Q
          0C    13 1923  4442         BEQL    30$                              ; If EQL none to complete
       34 A4    D4 1925  4443         CLRL    NOB$L_XMT_INPR(R4)               ; Clear transmit from inprogress
          02    E0 1928  4444         BBS     #XMTQ$V_CONTROL,-                ; If BS then control message
  04 1F A1          192A  4445                 XMTQ$B_FLAG(R1),30$              ; don't put on post queue
```

```
                2C B4   61    0E  192D  4446          INSQUE  (R1),@NOB$Q_POST+4(R4)          ; Insert to be completed
                                  1931  4447 30$:     ENBINT                                  ; Return to fork level
                                  1934  4448
                                  1934  4449 ; Deallocate all the attention AST control blocks
                                  1934  4450
                      54    DD    1934  4451          PUSHL   R4
                57    00AC C5     9E  1936  4452 40$:     MOVAB   UCB$L_NO_AST(R5),R7             ; Address list head for AST's
                      50    67    D0  193B  4453          MOVL    (R7),R0                         ; Anything in the list?
                            1B    13  193E  4454          BEQL    45$                             ; If EQL then empty
                      56    22 A0 3C  1940  4455          MOVZWL  ACB$L_KAST+10(R0),R6            ; Force channel
                      52    24 A0 3C  1944  4456          MOVZWL  ACB$L_KAST+12(R0),R2            ; Get PID index
             54   00000000'GF    D0  1948  4457          MOVL    G^SCH$GL_PCBVEC,R4
                      54  6442 D0    194F  4458          MOVL    (R4)[R2],R4                     ; Get PCB
                00000000'GF    16  1953  4459          JSB     G^COM$FLUSHATTNS                ; Flush the attention AST's
                            DB    11  1959  4460          BRB     40$                             ; Continue until done
                      54 8ED0     195B  4461 45$:     POPL    R4
                                  195E  4462
                                  195E  4463 ; Clear the post queue
                                  195E  4464
                      56    28 B4   0F  195E  4465 50$:     REMQUE  @NOB$Q_POST(R4),R6              ; Get next buffer to complete
                            24    1D  1962  4466          BVS     60$                             ; If VS then no buffer found
                13    0A A6   91  1964  4467          CMPB    IRP$B_TYPE(R6),S^#DYN$C_BUFIO   ; If NEQ then not XMT buffer
                            12    12  1968  4468          BNEQ    55$
                53    0C A6   D0  196A  4469          MOVL    XMTQ$L_IRP(R6),R3               ; If EQL then no IRP to comp
                            EE    13  196E  4470          BEQL    50$
                   51       56  D0  1970  4471          MOVL    R6,R1                           ; Set up R1 for branch
                38 A3       2C  3C  1973  4472          MOVZWL  #SS$_ABORT,IRP$L_MEDIA(R3)      ; Set abort status and compl
                      FC43      30  1977  4473          BSBW    TRANSMIT_IO_DONE                ; the request
                            E2    11  197A  4474          BRB     50$
                17    0A A6   91  197C  4475 55$:     CMPB    IRP$B_TYPE(R6),S^#DYN$C_NET     ; If EQL then RCV buffer
                            2D    12  1980  4476          BNEQ    85$
                24 B4       66  0E  1982  4477          INSQUE  (R6),@NOB$Q_FREE+4(R4)          ; Put on free Q to complete
                            D6    11  1986  4478          BRB     50$
                                  1988  4479
                                  1988  4480 ; Deallocate all receive buffers
                                  1988  4481
                56    20 A4   9E  1988  4482 60$:     MOVAB   NOB$Q_FREE(R4),R6               ; Get queue listhead
                            08    10  198C  4483          BSBB    70$                             ; Empty queue
                56    10 A4   9E  198E  4484          MOVAB   NOB$Q_ATTN(R4),R6               ; Get queue listhead
                            02    10  1992  4485          BSBB    70$                             ; Empty queue
                            1D    11  1994  4486          BRB     90$                             ; Continue
                                  1996  4487
                50    00 B6   0F  1996  4488 70$:     REMQUE  @(R6),R0                        ; Get buffer
                            12    1D  199A  4489          BVS     80$                             ; If VS then none
             40 A4      42 A5   A0  199C  4490          ADDW    UCB$W_DEVBUFSIZ(R5),NOB$W_QUOTA(R4) ; Restore quota
                            54    DD  19A1  4491          PUSHL   R4
                00000000'GF    16  19A3  4492          JSB     G^COM$DRVDEALMEM                ; Deallocate buffer
                      54 8ED0     19A9  4493          POPL    R4
                            E8    11  19AC  4494          BRB     70$                             ; Loop
                                  19AE  4495 80$:     RSB
                                  19AF  4496
                                  19AF  4497 85$:     BUG_CHECK NOBUFPCKT,FATAL
                                  19B3  4498
                                  19B3  4499 ; Complete all associated receive IO packets
                                  19B3  4500
                53    18 B4   0F  19B3  4501 90$:     REMQUE  @NOB$Q_RCVS(R4),R3              ; If VS then queue is empty
                            09    1D  19B7  4502          BVS     100$
```

```
                38 A3    2C    3C  19B9  4503          MOVZWL  #SS$_ABORT,IRP$L_MEDIA(R3)      ; Set status and size
                     FC0B    30  19BD  4504          BSBW    IO_DONE                          ; Complete the request
                       F1    11  19C0  4505          BRB     90$                              ; Branch to get next IRP
                             19C2  4506
                             19C2  4507 ; Complete all XMITS
                             19C2  4508
                             19C2  4509 100$:    DSBINT  UCB$B_DIPL(R5)                   ; Lock out interrupts
                64 A5    03    AA  19C9  4510          BICW    #UCB$M_TIM!UCB$M_INT,UCB$W_STS(R5)
                             19CD  4511          ENBINT                                   ; Reset interrupts
                       38    BB  19D0  4512          PUSHR   #^M<R3,R4,R5>                    ; Save registers
                   56    04    9A  19D2  4513          MOVZBL  #DLK$C_USRINT,R6                 ; Set up to halt the protocol
                   57    02    9A  19D5  4514          MOVZBL  #DLK$M_STOP,R7
                       58    7C  19D8  4515          CLRQ    R8
                55    0C A4    D0  19DA  4516          MOVL    NOB$A_PRO_BUFFER(R4),R5          ; Set protocol buffer
                     E61F'   30  19DE  4517          BSBW    DDCMP
                56    28 A5    9E  19E1  4518          MOVAB   TF$Q_CMPQ(R5),R6                 ; Get queue to complete from
                       38    BA  19E5  4519          POPR    #^M<R3,R4,R5>                    ; Restore the registers
                51    00 B6    0F  19E7  4520 110$:    REMQUE  @(R6),R1                         ; Complete all XMT's
                       13    1D  19EB  4521          BVS     120$                             ; If VS then no more to cmplt
                13    0A A1    91  19ED  4522          CMPB    XMTQ$B_BUFTYP(R1),S^#DYN$C_BUFIO ; Branch NEQ not a valid buff
                       BC    12  19F1  4523          BNEQ    85$
                53    0C A1    D0  19F3  4524          MOVL    XMTQ$L_IRP(R1),R3                ; Get associated IRP
                38 A3    2C    3C  19F7  4525          MOVZWL  #SS$_ABORT,IRP$L_MEDIA(R3)       ; Set aborted status
                     FBBF    30  19FB  4526          BSBW    TRANSMIT_IO_DONE                 ; Complete the I/O
                       E7    11  19FE  4527          BRB     110$                             ; Branch to get next
                   50    01    3C  1A00  4528 120$:    MOVZWL  S^#SS$_NORMAL,R0                 ; Assume shutdown
                   03DC 8F    BA  1A03  4529          POPR    #^M<R2,R3,R4,R6,R7,R8,R9>        ; Save the registers
                             05  1A07  4530          RSB
                                 1A08  4531
```

```
                     1A08  4533          .SBTTL  VALIDATE_P2, Validate P2 buffer parameters
                     1A08  4534          .SBTTL  VALIDATE_P2_TRIB, Validate P2 buffer with Trib param
                     1A08  4535          .SBTTL  VALIDATE_P2_UCB, Validate P2 buffer with UCB
                     1A08  4536
                     1A08  4537 ;++
                     1A08  4538 ; VALIDATE_P2 - Validate P2 buffer parameters
                     1A08  4539 ; VALIDATE_P2_TRIB - Validate P2 buffer with trib parameters
                     1A08  4540 ; VALIDATE_P2_UCB - Validate P2 buffer with UCB
                     1A08  4541 ;
                     1A08  4542 ; This routine is called tc validate the P2 buffer parameters. The parameters
                     1A08  4543 ; are checked against a parameter table which verifies that the minimum value
                     1A08  4544 ; and maximum value is not violated, and that invalid status flags are not set.
                     1A08  4545 ;
                     1A08  4546 ; Inputs:
                     1A08  4547 ;
                     1A08  4548 ;       R1 = Address of parameter verification table (VALIDATE_P2 entry only)
                     1A08  4549 ;       R2 = Status word from UCB
                     1A08  4550 ;       R3 = IRP address
                     1A08  4551 ;       R4 = NOB or trib parameter block address for value
                     1A08  4552 ;            checking (VALIDATE_P2 entry only)
                     1A08  4553 ;       R5 = UCB address
                     1A08  4554 ;
                     1A08  4555 ;       IPL = FIPL
                     1A08  4556 ;
                     1A08  4557 ; Outputs:
                     1A08  4558 ;
                     1A08  4559 ;       R0 = status return of parameters
                     1A08  4560 ;
                     1A08  4561 ; If no error:
                     1A08  4562 ;       R1 = Address of parameter verification table
                     1A08  4563 ; If error:
                     1A08  4564 ;       R1 = Bad parameter value
                     1A08  4565 ;
                     1A08  4566 ;       All other registers are preserved.
                     1A08  4567 ;
                     1A08  4568 ;--
                     1A08  4569          .ENABL  LSB
                     1A08  4570 VALIDATE_P2_TRIB:                            ; Validate P2 buffer with TFB
          54    DD   1A08  4571          PUSHL   R4                         ; Save R4
51   E6C6 CF    9E   1A0A  4572          MOVAB   TRIB_PARAM,R1              ; Get address of verification table
54   00A8 C5    D0   1A0F  4573          MOVL    UCB$L_NO_BUFFER(R5),R4     ; Get NOB address
  54   60 A4    9E   1A14  4574          MOVAB   NOB$Z_DDCMP(R4),R4         ; Get address of block to set
                     1A18  4575                                             ; DDCMP parameters
          0C    11   1A18  4576          BRB     10$
                     1A1A  4577 VALIDATE_P2_UCB:                            ; Validate P2 buffer with UCB
          54    DD   1A1A  4578          PUSHL   R4                         ; Save R4
51   E680 CF    9E   1A1C  4579          MOVAB   LINE_PARAM,R1              ; Get address of verification table
54   00A8 C5    D0   1A21  4580          MOVL    UCB$L_NO_BUFFER(R5),R4     ; Get NOB address
          04    10   1A26  4581 10$:     BSBB    VALIDATE_P2                ; Do the validation
          54 8ED0   1A28  4582          POPL    R4                         ; Restore R4
             05     1A2B  4583          RSB                                ; Return to caller
                    1A2C  4584          .DSABL  LSB
                    1A2C  4585
                    1A2C  4586 VALIDATE_P2:                                 ; Validate P2 buffer parameters
       03EE 8F    BB   1A2C  4587          PUSHR   #^M<R1,R2,R3,R5,R6,R7,R8,R9>; Save registers
                    1A30  4588                                             ; NB:R1 must be on top of stack
  56   2C A3    D0   1A30  4589          MOVL    IRP$L_SVAPTE(R3),R6        ; Get system P2 buffer address
```

```
                03   12   1A34  4590          BNEQ    10$                                     ; Br if a system buffer
              0085   31   1A36  4591          BRW     150$                                    ; Else, leave
        58    32 A3  3C   1A39  4592 10$:      MOVZWL  IRP$W_BCNT(R3),R8                       ; Get size of P2 buffer
           58   06   C6   1A3D  4593          DIVL    #6,R8                                   ; Calculate number of parameters
           56   66   D0   1A40  4594          MOVL    P2B_L_POINTER(R6),R6                    ; Point to start of P2 data
                         1A43  4595
                         1A43  4596 ; Loop to check next parameter in P2 buffer
                         1A43  4597
           50   86   3C   1A43  4598 30$:     MOVZWL  (R6)+,R0                                ; Get parameter type from P2
           55   86   D0   1A46  4599          MOVL    (R6)+,R5                                ; Get parameter value from P2
           57   6E   D0   1A49  4600          MOVL    (SP),R7                                 ; Get parameter table address
                         1A4C  4601
                         1A4C  4602 ; Loop to check P2 buffer parameter to Line parameter table
                         1A4C  4603
           59   87   B0   1A4C  4604 40$:     MOVW    (R7)+,R9                                ; Get parameter + flags
                72   13   1A4F  4605          BEQL    170$                                    ; Br if end of verify table
     50    59   0C   00   ED   1A51  4606          CMPZV   #PRM_V_TYPE,#PRM_S_TYPE,R9,R0   ; Parameters match?
                16   13   1A56  4607          BEQL    50$                                     ; Br if yes
                87   B5   1A58  4608          TSTW    (R7)+                                   ; Skip offset word
                         1A5A  4609          SKIP    PRM_V_MIN,R9,R7                         ; Skip minimum value
                         1A60  4610          SKIP    PRM_V_MAX,R9,R7                         ; Skip maximum value
                         1A66  4611          SKIP    PRM_V_INVALID,R9,R7                     ; Skip invalid flags
                DE   11   1A6C  4612          BRB     40$                                     ; Try next parameter
                         1A6E  4613
                         1A6E  4614 ; Match found - nullify if same value & check min,max,valid,invalid
                         1A6E  4615
           51   87   B0   1A6E  4616 50$:     MOVW    (R7)+,R1                                ; Get offset + width
                54   D5   1A71  4617          TSTL    R4                                      ; Is data structure present?
                2B   13   1A73  4618          BEQL    100$                                    ; Br if not - check values
     53    51   02   0E   EF   1A75  4619          EXTZV   #OFF_V_WIDTH,#OFF_S_WIDTH,R1,R3 ; Get width only
     51    51   0E   00   EF   1A7A  4620          EXTZV   #OFF_V_VALUE,#OFF_S_VALUE,R1,R1 ; Get offset only
           51   54   C0   1A7F  4621          ADDL    R4,R1                                   ; Calculate address of datum
                         1A82  4622          CASE    R3,TYPE=B,LIMIT=#1,<-                   ; Br to handler
                         1A82  4623                  60$,-                                   ;   Byte value
                         1A82  4624                  70$,-                                   ;   Word value
                         1A82  4625                  80$>                                    ;   Longword value
                         1A8C  4626
                         1A8C  4627 ; Byte value in structure
                         1A8C  4628
           61   55   91   1A8C  4629 60$:     CMPB    R5,(R1)                                 ; Is this the same?
                08   11   1A8F  4630          BRB     90$                                     ; Check result
                         1A91  4631
                         1A91  4632 ; Word value
                         1A91  4633
           61   55   B1   1A91  4634 70$:     CMPW    R5,(R1)                                 ; Is this the same?
                03   11   1A94  4635          BRB     90$                                     ; Check result
                         1A96  4636
           61   55   D1   1A96  4637 80$:     CMPL    R5,(R1)                                 ; Is this the same?
                05   12   1A99  4638 90$:     BNEQ    100$                                    ; Br if no - continue checks
              FA A6   B4   1A9B  4639          CLRW    -6(R6)                                  ; Nullify the parameter code
                1B   11   1A9E  4640          BRB     140$                                    ; Try next parameter - skip checks
                         1AA0  4641
     05    59   0C   E1   1AA0  4642 100$:     BBC     #PRM_V_MIN,R9,110$                      ; Br if no minimum value
           87   55   B1   1AA4  4643          CMPW    R5,(R7)+                                ; Is the value too small?
                1A   1F   1AA7  4644          BLSSU   170$                                    ; Br if yes - error
     05    59   0D   E1   1AA9  4645 110$:     BBC     #PRM_V_MAX,R9,130$                      ; Br if no maximum value
           87   55   B1   1AAD  4646          CMPW    R5,(R7)+                                ; Is the value too big?
```

```
                11   1A  1AB0  4647          BGTRU   170$                    ; Br if yes - error
05 59           0E   E1  1AB2  4648 130$:    BBC     #PRM_V_INVALID,R9,140$  ; Br if no invalid flags
   52           87   B3  1AB6  4649          BITW    (R7)+,R2                ; Check invalid bits
                08   12  1AB9  4650          BNEQ    170$                    ; Br if on - error
      85        58   F5  1ABB  4651 140$:    SOBGTR  R8,30$                  ; Loop if more parameters
                         1ABE  4652
   50           01   3C  1ABE  4653 150$:    MOVZWL  S^#SS$_NORMAL,R0        ; Set success return
                06   11  1AC1  4654          BRB     180$                    ; And return
                         1AC3  4655
   6E           50   3C  1AC3  4656 170$:    MOVZWL  R0,(SP)                 ; Return bad parameter code
                         1AC6  4657                                          ; * R1 Must be on top of stack
   50           14   3C  1AC6  4658          MOVZWL  #SS$_BADPARAM,R0        ; Set error return
   03EE         8F   BA  1AC9  4659 180$:    POPR    #^M<R1,R2,R3,R5,R6,R7,R8,R9>    ; Restore registers
                     05  1ACD  4660          RSB                             ; Return to caller
```

```
                     1ACE  4662          .SBTTL  UPDATE_P2,  Update UCB/TRIB based on P2 buffer parameters
                     1ACE  4663
                     1ACE  4664 ;++
                     1ACE  4665 ; UPDATE_P2 - Update UCB/TRIB with P2 buffer parameters
                     1ACE  4666 ;
                     1ACE  4667 ; This routine is called to update the UCB/TRIB with the P2 buffer parameters.
                     1ACE  4668 ; The parameters are stored in the appropriate cells of the UCB/TRIB.
                     1ACE  4669 ;
                     1ACE  4670 ; Inputs:
                     1ACE  4671 ;
                     1ACE  4672 ;       R1 = Address of parameter verification table
                     1ACE  4673 ;       R3 = IRP address
                     1ACE  4674 ;       R4 = NOB or ddcmp parameter block  address for storing
                     1ACE  4675 ;       R5 = UCB address
                     1ACE  4676 ;
                     1ACE  4677 ;       IPL = FIPL
                     1ACE  4678 ;
                     1ACE  4679 ; Outputs:
                     1ACE  4680 ;
                     1ACE  4681 ;       R0 = destroyed.
                     1ACE  4682 ;       All other registers are preserved.
                     1ACE  4683 ;
                     1ACE  4684 ;--
                     1ACE  4685
                     1ACE  4686 UPDATE_P2:                                          ; Update the UCB/DDCMP parametr
    03E6 8F    BB    1ACE  4687          PUSHR   #^M<R1,R2,R5,R6,R7,R8,R9>         ; Save registers
                     1AD2  4688                                                    ; NB:R1 must be on top of stack
56    2C A3    D0    1AD2  4689          MOVL    IRP$L_SVAPTE(R3),R6               ; Get system P2 buffer address
          62    13    1AD6  4690          BEQL    80$                               ; Br if no system buffer
58    32 A3    3C    1AD8  4691          MOVZWL  IRP$W_BCNT(R3),R8                 ; Get size of P2 buffer
   58    06    C6    1ADC  4692          DIVL    #6,R8                             ; Calculate number of parameters
   56    66    D0    1ADF  4693          MOVL    P2B_L_POINTER(R6),R6              ; Point to start of data
                     1AE2  4694
                     1AE2  4695 ; Loop to get next parameter from P2 buffer
                     1AE2  4696
          50    86    3C    1AE2  4697 10$:    MOVZWL  (R6)+,R0                          ; Get parameter type from P2
          55    86    D0    1AE5  4698          MOVL    (R6)+,R5                          ; Get parameter value from P2
          57    6E    D0    1AE8  4699          MOVL    (SP),R7                           ; Get parameter table address
                     1AEB  4700
                     1AEB  4701 ; Loop to store buffer parameter in UCB/DDCMP parameter block
                     1AEB  4702
          59    87    B0    1AEB  4703 20$:    MOVW    (R7)+,R9                          ; Get parameter + flags
                47    13    1AEE  4704          BEQL    70$                               ; Br if end of verify table
51  59    0C    00    EF    1AF0  4705          EXTZV   #PRM_V_TYPE,#PRM_S_TYPE,R9,R1     ; Get type field
          51    50    B1    1AF5  4706          CMPW    R0,R1                             ; Parameters match?
                16    13    1AF8  4707          BEQL    30$                               ; Br if yes
                87    B5    1AFA  4708          TSTW    (R7)+                             ; Skip offset word
                      1AFC  4709          SKIP    PRM_V_MIN,R9,R7                   ; Skip minimum value
                      1B02  4710          SKIP    PRM_V_MAX,R9,R7                   ; Skip maximum value
                      1B08  4711          SKIP    PRM_V_INVALID,R9,R7               ; Skip invalid flags
                DB    11    1B0E  4712          BRB     20$                               ; Try next parameter
                      1B10  4713
                      1B10  4714 ; Match found - nullify if same value & check min,max,valid,invalid
                      1B10  4715
          51    87    B0    1B10  4716 30$:    MOVW    (R7)+,R1                          ; Get offset + width
52  51    02    0E    EF    1B13  4717          EXTZV   #OFF_V_WIDTH,#OFF_S_WIDTH,R1,R2   ; Get width only
51  51    0E    00    EF    1B18  4718          EXTZV   #OFF_V_VALUE,#OFF_S_VALUE,R1,R1   ; Get offset only
```

```
                     1B1D  4719          ADDL    R4,R1                           ; Calculate address of datum
                     1B20  4720          CASE    R2,TYPE=B,LIMIT=#1,<-           ; Br to handler
                     1B20  4721                  40$,-                           ;   Byte value
                     1B20  4722                  50$,-                           ;   Word value
                     1B20  4723                  60$>                            ;   Longword value
                     1B2A  4724
                     1B2A  4725 ; Byte, word, longword value in structure
                     1B2A  4726
51   54   C0         1B2A  4727 40$:    MOVB    R5,(R1)                         ; Store byte value
     08   11         1B2D  4728         BRB     70$                             ;
61   55   B0         1B2F  4729 50$:    MOVW    R5,(R1)                         ; Store word value
     03   11         1B32  4730         BRB     70$
61   55   D0         1B34  4731 60$:    MOVL    R5,(R1)                         ; Store longword value
  A8 58   F5         1B37  4732 70$:    SOBGTR  R8,10$                          ; Br if more parameters
                     1B3A  4733
03E6 8F   BA         1B3A  4734 80$:    POPR    #^M<R1,R2,R5,R6,R7,R8,R9>       ; Restore registers
          05         1B3E  4735         RSB                                     ; Return to caller
```

```
                                1B3F  4737          .SBTTL  RETURN_P2,  Return UCB/DDCMP buffer parameters
                                1B3F  4738
                                1B3F  4739 ;++
                                1B3F  4740 ; RETURN_P2 - Return P2 buffer parameters
                                1B3F  4741 ;
                                1B3F  4742 ; This routine is called to return the UCB/DDCMP buffer parameters.
                                1B3F  4743 ;
                                1B3F  4744 ; Inputs:
                                1B3F  4745 ;
                                1B3F  4746 ;       R1 = Address of return table (same format as verification table)
                                1B3F  4747 ;       R2 = Address of user buffer in which to return the parameters
                                1B3F  4748 ;       R3 = IRP address
                                1B3F  4749 ;       R4 = NOB or DDCMP paramter block address for storing
                                1B3F  4750 ;       R5 = UCB address
                                1B3F  4751 ;       IRP$W_BCNT(R3) = Size of transfer
                                1B3F  4752 ;
                                1B3F  4753 ; Outputs:
                                1B3F  4754 ;
                                1B3F  4755 ;       R0 = destroyed.
                                1B3F  4756 ;       All other registers are preserved.
                                1B3F  4757 ;
                                1B3F  4758 ;--
                                1B3F  4759
                                1B3F  4760 RETURN_P2:                                   ; Return P2 buffer parameters
    01E6 8F    BB              1B3F  4761          PUSHR   #^M<R1,R2,R5,R6,R7,R8>      ; Save registers
    56   52    D0              1B43  4762          MOVL    R2,R6                       ; Get user buffer address
         50    13              1B46  4763          BEQL    60$                         ; Br if no system buffer
58     32 A3    3C             1B48  4764          MOVZWL  IRP$W_BCNT(R3),R8           ; Get size of buffer
    58   06    C6              1B4C  4765          DIVL    #6,R8                       ; Calculate the number of param
                                1B4F  4766
                                1B4F  4767 ; Loop to return next parameter
                                1B4F  4768
         55    81    B0        1B4F  4769 10$:     MOVW    (R1)+,R5                    ; Get parameter + flags
               44    13        1B52  4770          BEQL    60$                         ; Br if end of verify table
57   55    0C    00    EF      1B54  4771          EXTZV   #PRM_V_TYPE,#PRM_S_TYPE,R5,R7 ; Get type field
               86    57    B0  1B59  4772          MOVW    R7,(R6)+                    ; Return parameter
               57    81    B0  1B5C  4773          MOVW    (R1)+,R7                    ; Get offset + width
52   57    02    0E    EF      1B5F  4774          EXTZV   #OFF_V_WIDTH,#OFF_S_WIDTH,R7,R2 ; Get width only
57   57    0E    00    EF      1B64  4775          EXTZV   #OFF_V_VALUE,#OFF_S_VALUE,R7,R7 ; Get offset only
               57    54    C0  1B69  4776          ADDL    R4,R7                       ; Calculate address of datum
                                1B6C  4777          CASE    R2,TYPE=B,LIMIT=#1,<-       ; Br to handler
                                1B6C  4778                  20$,-                       ;   Byte value
                                1B6C  4779                  30$,-                       ;   Word value
                                1B6C  4780                  40$>                        ;   Longword value
                                1B76  4781
                                1B76  4782 ; Byte, word, longword value in structure
                                1B76  4783
               86    67    9A  1B76  4784 20$:     MOVZBL  (R7),(R6)+                  ; Store byte value
                     08    11  1B79  4785          BRB     50$                         ;
               86    67    3C  1B7B  4786 30$:     MOVZWL  (R7),(R6)+                  ; Store word value
                     03    11  1B7E  4787          BRB     50$
               86    67    D0  1B80  4788 40$:     MOVL    (R7),(R6)+                  ; Store longword value
                               1B83  4789 50$:     SKIP    PRM_V_MIN,R5,R1             ; Skip minimum value
                               1B89  4790          SKIP    PRM_V_MAX,R5,R1             ; Skip maximum value
                               1B8F  4791          SKIP    PRM_V_INVALID,R5,R1         ; Skip invalid flags
                  B7   58    F5  1B95  4792          SOBGTR  R8,10$                      ; Try for more parameters
                               1B98  4793
```

```
01E6 8F   BA  1B98  4794 60$:    POPR    #^M<R1,R2,R5,R6,R7,R8>          ; Restore registers
          05  1B9C  4795         RSB                                     ; Return to caller
```

```
                     1B9D  4797          .SBTTL  UNPACK_P2_BUF,  Unpack a P2 parameter from P2 buffer
                     1B9D  4798
                     1B9D  4799 ;++
                     1B9D  4800 ; UNPACK_P2_BUF - Unpack a P2 parameter from P2 buffer
                     1B9D  4801 ;
                     1B9D  4802 ; Functional description:
                     1B9D  4803 ;
                     1B9D  4804 ; This routine is called to get a P2 parameter from the P2 buffer.
                     1B9D  4805 ;
                     1B9D  4806 ; Inputs:
                     1B9D  4807 ;
                     1B9D  4808 ;       R1 = Parameter type code
                     1B9D  4809 ;       R3 = IRP address
                     1B9D  4810 ;       R5 = UCB address
                     1B9D  4811 ;
                     1B9D  4812 ; Outputs:
                     1B9D  4813 ;
                     1B9D  4814 ;       R0 = Low bit set if specified Parameter type code is found in P2
                     1B9D  4815 ;       R2 = Parameter value if success else destroyed
                     1B9D  4816 ;
                     1B9D  4817 ;       All other registers are preserved.
                     1B9D  4818 ;
                     1B9D  4819 ;--
                     1B9D  4820
                     1B9D  4821 UNPACK_P2_BUF:                                  ; Unpack P2 buffer
   00E0 8F    BB     1B9D  4822          PUSHR   #^M<R5,R6,R7>                  ; Save registers
56   2C A3    D0     1BA1  4823          MOVL    IRP$L_SVAPTE(R3),R6            ; Get system P2 buffer address
        1C    13     1BA5  4824          BEQL    20$                            ; Br if none
57   32 A3    3C     1BA7  4825          MOVZWL  IRP$W_BCNT(R3),R7              ; Get size of P2 buffer
   57   06    C6     1BAB  4826          DIVL    #6,R7                          ; Caculate number of parameters
56   0C A6    9E     1BAE  4827          MOVAB   P2B_T_DATA(R6),R6              ; Point to start of data
   50   01    3C     1BB2  4828          MOVZWL  S^#SS$_NORMAL,R0               ; Assume success
                     1BB5  4829
                     1BB5  4830 ; Loop to check next parameter in P2 buffer
                     1BB5  4831
   55   86    3C     1BB5  4832 10$:     MOVZWL  (R6)+,R5                       ; Get parameter type from P2
   52   86    D0     1BB8  4833          MOVL    (R6)+,R2                       ; Get parameter value from P2
   55   51    B1     1BBB  4834          CMPW    R1,R5                          ; Parameters match?
        05    13     1BBE  4835          BEQL    30$                            ; Br if yes
     F2 57    F5     1BC0  4836          SOBGTR  R7,10$                         ; Br if more parameters
                     1BC3  4837
        50    D4     1BC3  4838 20$:     CLRL    R0                             ; Return error
   00E0 8F    BA     1BC5  4839 30$:     POPR    #^M<R5,R6,R7>                  ; Restore registers
              05     1BC9  4840          RSB                                    ; Return to caller
                     1BCA  4841
                     1BCA  4842 XG_END:
                     1BCA  4843          .END
```

```
$$$                             = 00000020 R      02
$$$OFF                          = 00000009
$$$TYP                          = 00005479
$$$WID                          = 00004000
$$OP                            = 00000002
ABORTIO                           000004DD R      03
ACB$L_KAST                      = 00000018
ACCESS                            00000B0B R      03
ADDCMP_VECTOR                     00000078 R      03
ADDFREELIST                       00001078 R      03
ALLOC_BUFFER                      000007A5 R      03
ALLOC_P2BUF                       00000B24 R      03
ALT_RCVFDT                        00000488 R      03
ATS_NULL                          ******** X    02
BUG$_NOBUFPCKT                    ******** X    03
CHECK_BUFS                        00000AE5 R      03
CHECK_P1                          00000B11 R      03
CHECK_P2                          00000AE7 R      03
CHG_TRIB                          00000BE7 R      03
CHG_UCB_NOB                       00000B7C R      03
CLASSS_CLASS_DEF                = 00000028
CLASS_DDT                       = 00000010
CLASS_DISCONNECT                = 00000018
CLASS_DS_TRAN                   = 0000000C
CLASS_FORK                      = 0000001C
CLASS_GETNXT                    = 00000000
CLASS_LENGTH                    = 00000028
CLASS_POWERFAIL                 = 00000020
CLASS_PUTNXT                    = 00000004
CLASS_READERROR                 = 00000014
CLASS_SETUP_UCB                 = 00000008
CLEAR_NO_BUFFER                   00001811 R      03
COM$DELATTNAST                    ******** X    03
COM$DRVDEALMEM                    ******** X    03
COM$FLUSHATTNS                    ******** X    03
COM$POST                          ******** X    03
COM$SETATTNAST                    ******** X    03
COMP_RCV                          00001372 R      03
COMP_RCV_ERROR                    00001432 R      03
COMP_RCV_NOBUFFER                 00001381 R      03
COM_RCVFDT                        000004E5 R      03
COM_XMITFDT                       000003B3 R      03
COP_BUFF                          00000409 R      03
CRB$L_INTD                      = 00000024
CXB$C_TRAILER                   = 00000004
CXB$K_HEADER                    = 00000048
DATA_ERROR                        00001426 R      03
DC$_SCOM                          ******** X    02
DDB$L_DDT                       = 0000000C
DDCMP                             ******** X    03
DDCMP$C_HEADER                  = 00000006
DEF_LINE_PARAM                    000000FE RG     03
DEF_LINE_PARAMSZ                = 00000006
DEF_TRIB_PARAM                    000000F4 RG     03
DEF_TRIB_PARAMSZ                = 0000000A
DEV$M_IDV                       = 04000000
DEV$M_NET                       = 00002000
DEV$M_ODV                       = 08000000
DEVTIMER                          0000164C R      03
DLA$A_POSTQ                       00000004
DLA$A_XMT_INPR                    00000000
DLA$C_ADDR_LENGTH                 00000008
DLA$K_ADDR_LENGTH                 00000008
DLK$B_MAINT                       00000008
DLK$B_MAXREP                      00000002
DLK$B_MAXSEL                      00000003
DLK$B_MRB                         00000009
DLK$B_MSGCNT                      00000001
DLK$B_TRIB                        00000000
DLK$C_ACTNOTCOM                 = 00000009
DLK$C_CHAR                      = 00000005
DLK$C_RCVMSG                    = 00000001
DLK$C_REQEBA                    = 00000003
DLK$C_START_TIMER               = 00000006
DLK$C_STOP_TIMER                = 00000007
DLK$C_USRINT                    = 00000004
DLK$C_XMTMSG                    = 00000002
DLK$M_CLEAR                     = 00000001
DLK$M_DATACRC                   = 00000004
DLK$M_GLOB                      = 00000004
DLK$M_HDRCRC                    = 00000002
DLK$M_HDRERR                    = 00000008
DLK$M_LNTYP                     = 00000008
DLK$M_MAINT                     = 00000004
DLK$M_MNTRCV                    = 00000080
DLK$M_MSGCNT                    = 00000010
DLK$M_MSGSENT                   = 00000001
DLK$M_PRSTERR                   = 00000002
DLK$M_QEMPTY                    = 00000002
DLK$M_QFULERR                   = 00000020
DLK$M_RCVOVR                    = 00000020
DLK$M_REPTIM                    = 00000080
DLK$M_REPWAIT                   = 00000100
DLK$M_SELTIM                    = 00000020
DLK$M_SELWAIT                   = 00000040
DLK$M_SETDEF                    = 00000800
DLK$M_START                     = 00000001
DLK$M_STOP                      = 00000002
DLK$M_STRTRCV                   = 00000040
DLK$M_TMREXPD                   = 00008000
DLK$M_TRIB                      = 00000002
DLK$M_XMTACK                    = 00000200
DLK$V_CRC                       = 00000010
DLK$V_DUPLEX                    = 00000002
DLK$V_MNTRCV                    = 00000007
DLK$V_MSGSENT                   = 00000000
DLK$V_PRSTERR                   = 00000001
DLK$V_RCVACK                    = 00000008
DLK$V_SETDEF                    = 0000000B
DLK$V_STATYP                    = 00000000
DLK$V_TMREXPD                   = 0000000F
DLK$V_TRNLK                     = 0000000E
DLK$V_XMTCMP                    = 0000000A
DLK$V_XMTERR                    = 00000004
```

```
DLK$W_REPWAIT                   00000004
DLK$W_SELWAIT                   00000006
DPT$B_FLAGS                   = 0000000D
DPT$C_LENGTH                  = 00000038
DPT$C_VERSION                 = 00000004
DPT$INITAB                      00000038 R      02
DPT$M_NOUNLOAD                = 00000004
DPT$REINITAB                    00000087 R      02
DPT$TAB                         00000000 R      02
DPT$W_VECTOR                  = 0000001E
DRIVER_END                      ********   X    02
DSC$A_POINTER                 = 00000004
DTS_ADDCMP                    = 000000C8
DUETIM_TABLE                    00000104 R      03
DYN$C_BUFIO                   = 00000013
DYN$C_CRB                     = 00000005
DYN$C_DDB                     = 00000006
DYN$C_DPT                     = 0000001E
DYN$C_NET                     = 00000017
DYN$C_ORB                     = 00000049
DYN$C_UCB                     = 00000010
EXE$ABORTIO                     ********   X    03
EXE$ALLOCBUF                    ********   X    03
EXE$ALONONPAGED                 ********   X    03
EXE$BUFFRQUOTA                  ********   X    03
EXE$BUFQUOPRC                   ********   X    03
EXE$FINISHIO                    ********   X    03
EXE$FORK                        ********   X    03
EXE$GL_ABSTIM                   ********   X    03
EXE$GQ_SYSTIME                  ********   X    03
EXE$QIORETURN                   ********   X    03
EXE$READCHK                     ********   X    03
EXE$WRITECHK                    ********   X    03
EXE$WRITECHKR                   ********   X    03
FILLFREELIST                    00001070 R      03
FILL_DUETIM_TABLE               00000D69 R      03
FINISH_RCV_IO                   00001579 R      03
FINISH_XMT_IO                   0000160F R      03
FORK_DONE                       00001451 R      03
FUNCTAB_LEN                   = 00000040
GET_CHAR_BUFS                   00000A8J R      03
GET_RECEIVE                     000010DB R      03
GET_XMT                         00000CDF R      03
GF$B_BABTIM                     00000016
GF$B_CURSEL                     00000008
GF$B_DIPL                       0000000A
GF$B_DRVCHR                     00000002
GF$B_FIPL                       00000009
GF$B_GH_CRC                     00000029
GF$B_LSE                        00000023
GF$B_MAXSEL                     00000007
GF$B_MDF_CRC                    0000002A
GF$B_RSE                        00000028
GF$B_STATE                      00000000
GF$B_STRTIM                     00000015
GF$B_STRTSEL                    0000002B
GF$B_TIMER_STATE                00000001
GF$C_ERREND                     00000029
GF$C_ERRSRT                     0000001F
GF$C_LENGTH                     0000002C
GF$K_ERREND                     00000029
GF$K_ERRSRT                     0000001F
GF$K_LENGTH                     0000002C
GF$L_BABTMR                     0000001B
GF$L_SELEND                     0000000D
GF$L_STRTMR                     00000017
GF$L_TQE_STS                    00000011
GF$W_GEB                        0000000B
GF$W_LSETYP                     0000001F
GF$W_LSE_BCTRS                  00000021
GF$W_NEXTCNT                    00000005
GF$W_RSETYP                     00000024
GF$W_RSE_BCTRS                  00000026
GF$W_SELQAI                     00000003
HEADER_ERROR                    00001410 R      03
INIT_NO_BUFFER                  000007D1 R      03
IO$M_CLR_COUNT                = 00000400
IO$M_RD_COUNT                 = 00000100
IO$V_ATTNAST                  = 00000008
IO$V_CLR_COUNT                = 0000000A
IO$V_CTRL                     = 00000009
IO$V_NOW                      = 00000006
IO$V_RD_COUNT                 = 00000008
IO$V_RD_MODEM                 = 00000007
IO$V_SHUTDOWN                 = 00000007
IO$V_STARTUP                  = 00000006
IO$_READLBLK                  = 00000021
IO$_READPBLK                  = 0000000C
IO$_READVBLK                  = 00000031
IO$_SENSEMODE                 = 00000027
IO$_SETCHAR                   = 0000001A
IO$_SETMODE                   = 00000023
IO$_VIRTUAL                   = 0000003F
IO$_WRITELBLK                 = 00000020
IO$_WRITEPBLK                 = 0000000B
IO$_WRITEVBLK                 = 00000030
IOC$INITIATE                    ********   X    03
IOC$MNTVER                      ********   X    03
IOC$REQCOM                      ********   X    03
IOC$RETURN                      ********   X    03
IO_DONE                         000015LB R      03
IPL$_POWER                    = 0000001F
IPL$_SYNCH                    = 00000008
IPL$_TIMER                    = 00000008
IRP$B_NOFUNC                    00000021
IRP$B_TYPE                    = 0000000A
IRP$L_DIAGBUF                 = 0000004C
IRP$L_MEDIA                   = 00000038
IRP$L_PID                     = 0000000C
IRP$L_SVAPTE                  = 0000002C
IRP$Q_STATION                 = 00000040
IRP$V_DIAGBUF                 = 00000007
IRP$V_FUNC                    = 00000001
IRP$W_BCNT                    = 00000032
```

```
IRP$W_BOFF                      =  00000030
IRP$W_CHAN                      =  00000028
IRP$W_FUNC                      =  00000020
IRP$W_QUOTA                        00000040
IRP$W_STS                       =  0000002A
JIB$L_BYTCNT                    =  00000020
LINE_PARAM                         000000A0 R      03
LINE_PRM_BUFSIZ                 =  00000024
MASKH                           =  00000080
MASKL                           =  00000000
MAX_RCVS                        =  00000004
MFD$B_ADDR                         00000005
MFD$B_MSGID                        00000000
MFD$B_NUMB                         00000004
MFD$B_RESP                         00000003
MFD$C_LENGTH                       00000006
MFD$K_LENGTH                       00000006
MFD$M_QSYNC                     =  00004000
MFD$M_SELECT                    =  00008000
MFD$W_CNTFLG                       00000001
MFD$W_TYPFLG                       00000001
NMA$C_DPX_FUL                   =  00000000
NMA$C_DPX_HAL                   =  00000001
NMA$C_LINCN_LOO                 =  00000001
NMA$C_LINCN_NOR                 =  00000000
NMA$C_LINPR_CON                 =  00000001
NMA$C_LINPR_POI                 =  00000000
NMA$C_LINPR_TRI                 =  00000002
NMA$C_PCCI_MRB                  =  00000479
NMA$C_PCCI_MST                  =  00000AFA
NMA$C_PCCI_MTR                  =  0000047A
NMA$C_PCCI_TRI                  =  00000474
NMA$C_PCLI_BFN                  =  00000451
NMA$C_PCLI_BUS                  =  00000AF1
NMA$C_PCLI_CON                  =  00000456
NMA$C_PCLI_DUP                  =  00000457
NMA$C_PCLI_PRO                  =  00000458
NMA$C_PCLI_RTT                  =  00000461
NMA$C_STATE_OFF                 =  00000001
NMA$C_STATE_ON                  =  00000000
NO$ALT_ENTRY                       00000425 R      03
NO$CANCEL                          000017E3 R      03
NO$CLASS_DISCONNECT                00001449 R      03
NO$CLASS_PORTFORK                  000013A0 R      03
NO$CLASS_POWERACTION               0000144A R      03
NO$CONTROL_INIT                    00000304 R      03
NO$C_CIR_PAR                    =  FFFFF700
NO$C_DATA                       =  00000003
NO$C_DCRC                       =  00000004
NO$C_DEF_BUFSIZ                 =  00000100
NO$C_DEL                        =  000000FF
NO$C_DLE                        =  00000090
NO$C_DUETIM_TABLE_SIZE          =  00000100
NO$C_EMPTY                      =  00000000
NO$C_ENQ                        =  00000005
NO$C_HCRC                       =  00000002
NO$C_HEADER                     =  00000001
NO$C_HEADER_HCRC                =  00000008
NO$C_HEADER_LEN                 =  00000006
NO$C_LINE_PAR                   =  FFF3F300
NO$C_NOBUFFER                   =  00000004
NO$C_PAD                        =  0000FFFF
NO$C_PADS                       =  00000005
NO$C_SEAR_MSG                   =  00000000
NO$C_SET_LINE                   =  00000018
NO$C_SOH                        =  00000081
NO$C_SYN                        =  00000096
NO$C_XMTERR                     =  0000000A
NO$DDT                             00000000 RG     03
NO$DPT                             00000000 R      02
NO$FUNCTABLE                       00000038 R      03
NO$GETNXT                          000010FA R      03
NO$GL_DPT                          ********   X    03
NO$NULL                            0000144E R      03
NO$PORT_TRANSITION                 000013E9 R      03
NO$PUTNXT                          00001274 R      03
NO$RCVFDT                          000004B5 R      03
NO$READERROR                       000013EA R      03
NO$REGDUMP                         000016DB R      03
NO$SENSEMODEFDT                    00000849 R      03
NO$SETMODEFDT                      0000052F R      03
NO$SETUP_UCB                       000013E8 R      03
NO$STARTIO                         00000C20 R      03
NO$UNIT_INIT                       00000323 R      03
NO$XMITFDT                         00000344 R      03
NOB$A_PRO_BUFFER                   0000000C
NOB$B_BFN                          00000073
NOB$B_CON                          00000072
NOB$B_DUP                          00000071
NOB$B_PRO                          00000070
NOB$B_RSTATE                       00000043
NOB$B_SPD                          00000076
NOB$B_TYPE                         0000000A
NOB$B_XSTATE                       00000042
NOB$C_LENGTH                       000000AA
NOB$C_SETPRM                       00000070
NOB$K_LENGTH                       000000AA
NOB$L_BLINK                        00000004
NOB$L_FLINK                        00000000
NOB$L_PID                          00000058
NOB$L_RCV_DUETIM                   00000038
NOB$L_RCV_INPR                     00000030
NOB$L_XMT_DUETIM                   0000003C
NOB$L_XMT_INPR                     00000034
NOB$Q_ATTN                         00000010
NOB$Q_FREE                         00000020
NOB$Q_POST                         00000028
NOB$Q_RCVS                         00000018
NOB$W_CHANC                        0000005E
NOB$W_CHANL                        0000005C
NOB$W_DEVBUFSIZ                    00000074
NOB$W_ERROR                        00000054
NOB$W_FLAGS                        00000056
NOB$W_INDEX                        00000052
```

```
NOB$W_MSGSIZ                00000050
NOB$W_PADS                  00000044
NOB$W_QUOTA                 00000040
NOB$W_SIZE                  00000008
NOB$W_XMTERR_SIZE           00000046
NOB$Z_CTL_MSG               00000080
NOB$Z_DDCMP                 00000060
NOB$Z_DLA_ADDR              00000078
NOB$Z_HEADER                00000048
NOBUFFER_ERROR              00001441 R      03
NO_DS_M_FORK_PEND         = 00000080
NO_DS_M_ILOOP_SUP         = 00000100
NO_DS_M_INITED            = 00000040
NO_DS_M_MSG_SENT          = 00000200
NO_DS_M_RCVING            = 00000020
NO_DS_M_RCV_TIME          = 00001000
NO_DS_M_SHUTDOWN          = 00000400
NO_DS_M_XMTING            = 00000010
NO_DS_M_XMT_TIME          = 00000800
NO_DS_V_FORK_PEND         = 00000007
NO_DS_V_ILOOP_SUP         = 00000008
NO_DS_V_INITED            = 00000006
NO_DS_V_MSG_SENT          = 00000009
NO_DS_V_RCVING            = 00000005
NO_DS_V_RCV_TIME          = 0000000C
NO_DS_V_SHUTDOWN          = 0000000A
NO_DS_V_XMTING            = 00000004
NO_DS_V_XMT_TIME          = 0000000B
NO_ERROR                    0000140B R      03
NO_FC_V_READ_MODEM        = 00000004
NO_FC_V_STOP_CIR          = 00000002
NO_FC_V_STOP_LIN          = 00000003
NO_FC_V_STRT_CIR          = 00000000
NO_FC_V_STRT_LIN          = 00000001
NO_FS_M_IOFORK            = 00000002
NO_FS_M_PORTFORK          = 00000001
NO_FS_M_POWERFAIL         = 00000004
NO_FS_V_IOFORK            = 00000001
NO_FS_V_PORTFORK          = 00000000
NO_FS_V_POWERFAIL         = 00000002
OFF_M_VALUE               = 00003FFF
OFF_S_VALUE               = 0000000E
OFF_S_WIDTH               = 00000002
OFF_V_VALUE               = 00000000
OFF_V_WIDTH               = 0000000E
ORB$B_FLAGS               = 0000000B
ORB$L_OWNER               = 00000000
ORB$M_PROT_16             = 00000001
ORB$W_PROT                = 00000018
OVERRUN_ERROR               00001420 R      03
P1                        = 00000000
P2                        = 00000004
P2B_B_SPARE                 0000000B
P2B_B_TYPE                  0000000A
P2B_C_LENGTH                0000000C
P2B_L_BUFFER                00000004
P2B_L_POINTER               00000000
P2B_T_DATA                  0000000C
P2B_W_SIZE                  00000008
P3                        = 00000008
P4                        = 0000000C
P5                        = 00000010
PCB$L_JIB                 = 00000080
PCB$L_PID                 = 00000060
POKE_USER                   000016E1 R      03
PORT_ABORT                = 00000020
PORT_DISCONNECT           = 00000004
PORT_FORKRET              = 00000034
PORT_RESUME               = 00000024
PORT_SET_LINE             = 00000008
PORT_STARTIO              = 00000000
PR$_IPL                     ******** X    03
PRM_M_INVALID             = 00004000
PRM_M_MAX                 = 00002000
PRM_M_MIN                 = 00001000
PRM_M_TYPE                = 00000FFF
PRM_S_TYPE                = 0000000C
PRM_V_INVALID             = 0000000E
PRM_V_MAX                 = 000C000D
PRM_V_MIN                 = 0000000C
PRM_V_TYPE                = 00000000
QUEPKT                      00000A70 R      03
RCV_B_BLKTYPE               0000000A
RCV_B_FIPL                  0000000B
RCV_CXB_SPARE               00000014
RCV_DATA                    0000134F R      03
RCV_HEADER                  000012ED R      03
RCV_L_LINK                  00000000
RCV_M_CNTL                = 00000001
RCV_NOBUFFER                00001392 R      03
RCV_T_DATA                  00000048
RCV_V_CNTL                = 00000000
RCV_W_BLKSIZE               00000008
RCV_W_ERROR                 00000010
RCV_W_FLAGS                 00000012
RCV_W_INDEX                 0000000E
RCV_W_MSGSIZ                0000000C
RCV_Z_HEADER                00000040
READ_MODEM                  00000D5A R      03
RECEIVE_DONE                000014DB R      03
RETURN_P2                   00001B3F R      03
SCH$GL_PCBVEC               ******** X    03
SCHED_FORK_IO               000013B8 R      03
SCHED_FORK_POWERFAIL        000013D0 R      03
SEARCH_MESSAGE              0000129E R      03
SEND_DATA                   00001203 R      03
SEND_DCRC                   00001223 R      03
SEND_HCRC                   000011EA R      03
SEND_HEADER                 000011D1 R      03
SEND_PADS                   0000123C R      03
SEND_XMTERR                 00001255 R      03
SENSEMODE_CTRL              00000956 R      03
SENSE_MODEM                 00000A51 R      03
SETMODE_CTRL                00000615 R      03
```

```
SET_CHAR                        0000102B R      03
SHUTDOWN_CIRCUIT                000018E0 R      03
SHUTDOWN_LINE                   0000184E R      03
SHUTDOWN_LINE_ALT               00001851 R      03
SIZ...                        = 00000002
SS$_ABORT                     = 0000002C
SS$_ACCVIO                    = 0000000C
SS$_BADPARAM                  = 00000014
SS$_BUFFEROVF                 = 00000601
SS$_CTRLERR                   = 00000054
SS$_DEVACTIVE                 = 000002C4
SS$_DEVINACT                  = 000020D4
SS$_ENDOFFILE                 = 00000870
SS$_NORMAL                    = 00000001
SS$_POWERFAIL                 = 00000364
START_CIRCUIT                   00000F86 R      03
START_CIRCUIT_COMP              0000101E R      03
START_CIRCUIT_ERROR             00001016 R      03
START_CTRL_ERROR                00000F7D R      03
START_DISP                      00000C35 R      03
START_LINE                      00000EA6 R      03
START_POWERFAIL                 00001011 R      03
START_RECEIVE                   000010CD R      03
START_TRANSMIT                  00000C47 R      03
TF$A_BUFPTR                     00000048
TF$A_DEV_TIMER                  00000050
TF$A_GFB                        0000004C
TF$A_POSTQ                      00000058
TF$A_XMT_INPR                   00000054
TF$B_A                          00000013
TF$B_ADDR                       0000000D
TF$B_CHAR                       0000000C
TF$B_CURRTO                     0000001B
TF$B_DEI                        000001AF
TF$B_DEO                        000001AA
TF$B_LBE                        000001B4
TF$B_LRTO                       000001C1
TF$B_MAXRTO                     0000001A
TF$B_MMCTR                      0000001D
TF$B_MSGCNT                     0000001C
TF$B_N                          00000012
TF$B_NAKRSN                     00000188
TF$B_R                          00000011
TF$B_RADDR                      0000000F
TF$B_RBE                        000001B9
TF$B_REPTIM                     00000017
TF$B_RRTO                       000001C4
TF$B_SELTIM                     00000016
TF$B_SFLAGS                     00000189
TF$B_STO                        000001BE
TF$B_T                          00000014
TF$B_X                          00000015
TF$B_XADDR                      0000000E
TF$B_XQCNT                      00000010
TF$C_ERREND                     000001C5
TF$C_ERRSTRT                    0000018A
TF$C_LENGTH                     000001C8
TF$K_ERREND                     000001C5
TF$K_ERRSTRT                    0000018A
TF$K_LENGTH                     000001C8
TF$L_DBYTR                      0000018C
TF$L_DBYTX                      00000192
TF$L_DMSGR                      00000198
TF$L_DMSGX                      0000019E
TF$L_HDR                        00000000
TF$L_QACK                       0000005C
TF$L_QNAK                       0000008C
TF$L_QREP                       000000BC
TF$L_QSTACK                     0000011C
TF$L_QSTRT                      000000EC
TF$L_TQE                        0000014C
TF$Q_CMPQ                       00000028
TF$Q_CTLQ                       00000020
TF$Q_RTOQ                       00000038
TF$Q_XMTOVF                     00000040
TF$Q_XMTQ                       00000030
TF$W_DBRTYP                     0000018A
TF$W_DBXTYP                     00000190
TF$W_DEIBC                      000001AD
TF$W_DEITYP                     000001AB
TF$W_DEOBC                      000001A8
TF$W_DEOTYP                     000001A6
TF$W_DMRTYP                     00000196
TF$W_DMXTYP                     0000019C
TF$W_LBEBC                      000001B2
TF$W_LBETYP                     000001B0
TF$W_LRTOTYP                    000001BF
TF$W_RBEBC                      000001B7
TF$W_RBETYP                     000001B5
TF$W_REPWAI                     00000018
TF$W_RRTOTYP                    000001C2
TF$W_SELSP                      000001A4
TF$W_SELTYP                     000001A2
TF$W_STOBC                      000001BC
TF$W_STOTYP                     000001BA
TF$W_TEB                        0000001E
TF$W_THRES                      000001C5
TIMEOUT                         00001711 R      03
TIMEOUT_DATA                    0000179A R      03
TIMEOUT_DCRC                    000017AF R      03
TIMEOUT_END                     000017C1 R      03
TIMEOUT_HCRC                    0000177B R      03
TIMEOUT_HEADER                  0000175F R      03
TIMEOUT_PADS                    000017B4 R      03
TIMEOUT_XMTERR                  000017BD R      03
TRANSMIT_DONE                   000015EF R      03
TRANSMIT_IO_DONE                000015BD R      03
TRIB_PARAM                      000000D4 R      03
TRIB_PRM_BUFSIZ               = 00000018
TT$C_BAUD_110                 = 00000003
TT$C_BAUD_1200                = 00000008
TT$C_BAUD_134                 = 00000004
TT$C_BAUD_150                 = 00000005
TT$C_BAUD_1800                = 00000009
```

```
TT$C_BAUD_19200                 = 00000010
TT$C_BAUD_2000                  = 0000000A
TT$C_BAUD_2400                  = 0000000B
TT$C_BAUD_300                   = 00000006
TT$C_BAUD_3600                  = 0000000C
TT$C_BAUD_4800                  = 0000000D
TT$C_BAUD_50                    = 00000001
TT$C_BAUD_600                   = 00000007
TT$C_BAUD_7200                  = 0000000E
TT$C_BAUD_75                    = 00000002
TT$C_BAUD_9600                  = 0000000F
TTY$M_PC_DMAENA                 = 00000002
TTY$M_PC_NOTIME                 = 00000001
TTY$M_PC_PRMMAP                 = 00000008
TTY$V_PC_XOFENA                 = 00000006
UCB$B_DEVCLASS                  = 00000040
UCB$B_DEVTYPE                   = 00000041
UCB$B_DIPL                      = 0000005E
UCB$B_FIPL                      = 0000000B
UCB$B_TT_OUTYPE                 = 0000010B
UCB$B_TT_PARITY                 = 000000F8
UCB$B_TYPE                      = 0000000A
UCB$C_TL_LENGTH                 = 000000B0
UCB$L_DEVCHAR                   = 00000038
UCB$L_DEVDEPEND                 = 00000044
UCB$L_DEVDEPND2                 = 00000048
UCB$L_FR3                       = 00000010
UCB$L_IRP                       = 00000058
UCB$L_NO_AST                      000000AC
UCB$L_NO_BUFFER                   000000A8
UCB$L_TT_OUTADR                 = 0000011C
UCB$L_TT_PORT                   = 00000118
UCB$M_INT                       = 00000002
UCB$M_ONLINE                    = 00000010
UCB$M_POWER                     = 00000020
UCB$M_TIM                       = 00000001
UCB$Q_TL_BRKTHRU                = 000000A8
UCB$S_TT_DEVSTS_FILL            = 0000000B
UCB$V_INT                       = 00000001
UCB$V_ONLINE                    = 00000004
UCB$V_POWER                     = 00000005
UCB$V_TT_DEVSTS_FILL            = 00000004
UCB$V_TT_HANGUP                 = 00000003
UCB$V_TT_NOLOGINS               = 0000000F
UCB$V_TT_NOTIF                  = 00000002
UCB$V_TT_TIMO                   = 00000001
UCB$W_BOFF                      = 0000007C
UCB$W_DEVBUFSIZ                 = 00000042
UCB$W_DEVSTS                    = 00000068
UCB$W_ERRCNT                    = 00000082
UCB$W_REFC                      = 0000005C
UCB$W_SIZE                      = 00000008
UCB$W_STS                       = 00000064
UCB$W_TT_OUTLEN                 = 00000120
UCB$W_TT_PRTCTL                 = 00000122
UCB$W_TT_SPEED                  = 000000F4
UNPACK_P2_BUF                     00001B9D R      03
UPDATE_P2                         00001ACE R      03
VALIDATE_P2                       00001A2C R      03
VALIDATE_P2_TRIB                  00001A08 R      03
VALIDATE_P2_UCB                   00001A1A R      03
VEC$L_INITIAL                   = 0000000C
VEC$L_UNITINIT                  = 00000018
XG_END                            00001BCA R      03
XM$M_CHR_CTRL                   = 00000040
XM$M_CHR_DMC                    = 00000020
XM$M_ERR_START                  = 00800000
XM$M_ERR_TRIB                   = 00400000
XM$M_STS_ACTIVE                 = 00000800
XM$V_CHR_HDPLX                  = 00000002
XM$V_CHR_LOOPB                  = 00000001
XM$V_CHR_MOP                    = 00000000
XM$V_ERR_FATAL                  = 00000010
XM$V_ERR_MAINT                  = 00000013
XM$V_ERR_START                  = 00000017
XM$V_ERR_TRIB                   = 00000016
XM$V_STS_ACTIVE                 = 0000000B
XM$V_STS_BUFFAIL                = 0000000C
XM$V_STS_DISC                   = 0000000E
XMTQ$B_BUFTYP                     0000000A
XMTQ$B_FLAG                       0000001F
XMTQ$B_MSGHDR                     00000024
XMTQ$B_SLOT                       0000001C
XMTQ$C_LENGTH                     0000002A
XMTQ$K_LENGTH                     0000002A
XMTQ$L_BACC                       00000014
XMTQ$L_IRP                        0000000C
XMTQ$L_TIMEND                     00000010
XMTQ$M_CONTROL                  = 00000004
XMTQ$M_INTERNAL                 = 00000080
XMTQ$M_ONQUEUE                  = 00000001
XMTQ$M_SELECT                   = 00000002
XMTQ$Q_LINK                       00000000
XMTQ$V_CONTROL                  = 00000002
XMTQ$V_INTERNAL                 = 00000007
XMTQ$V_SELECT                   = 00000001
XMTQ$W_BUFLEN                     00000008
XMTQ$W_DCRC                       00000022
XMTQ$W_ERROR                      0000001D
XMTQ$W_HCRC                       00000020
XMTQ$W_MSGOFF                     00000018
XMTQ$W_MSGSIZE                    0000001A
```

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	000001C8 (456.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$105_PROLOGUE	00000092 (146.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	00001BCA (7114.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.06	00:00:01.42
Command processing	145	00:00:00.55	00:00:04.37
Pass 1	1439	00:00:39.79	00:02:21.41
Symbol table sort	0	00:00:04.88	00:00:18.87
Pass 2	471	00:00:09.88	00:00:35.17
Symbol table output	2	00:00:00.37	00:00:01.56
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	2097	00:00:55.55	00:03:22.82

The working set limit was 3150 pages.
316805 bytes (619 pages) of virtual memory were used to buffer the intermediate code.
There were 250 pages of symbol table space allocated to hold 4324 non-local and 353 local symbols.
4843 source lines were read in Pass 1, producing 36 object records in Pass 2.
103 pages of virtual memory were used to define 95 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[DRIVER.OBJ]SYNCHLIB.MLB;1	7
_255DUA28:[SHRLIB]NMALIBRY.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	39
_255DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	58

4796 GETS were required to define 58 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:NODRIVER/OBJ=OBJ$:NODRIVER MSRC$:NODRIVER/UPDATE=(ENH$:NODRIVER)+EXECML$/LIB+SHRLIB$:NMALIBRY/LIB+LIB$:SYNCHLIB/LIB

LCDRIVER
LIS

LPDRIVER
LIS

NODRIVER
LIS

MBXDRIVER
LIS

PAEND
LIS

PACONFIG
LIS

PAERROR
LIS