



```

CCCCCCCC NN      NN DDDDDDDD RRRRRRRR I11111 VV      VV EEEEEEEEE RRRRRRRR
CCCCCCCC NN      NN DDDDDDDD RRRRRRRR I11111 VV      VV EEEEEEEEE RRRRRRRR
CC        NN      NN DD      DD RR      RR RR      RR RR      RR
CC        NN      NN DD      DD RR      RR RR      RR RR      RR
CC        NNNN     NN DD      DD RR      RR RR      RR RR      RR
CC        NNNN     NN DD      DD RR      RR RR      RR RR      RR
CC        NN  NN   NN DD      DD RRRRRRRR I11111 VV      VV EEEEEEEEE RRRRRRRR
CC        NN  NN   NN DD      DD RRRRRRRR I11111 VV      VV EEEEEEEEE RRRRRRRR
CC        NN      NNNN DD      DD RR  RR   I11111 VV      VV EE
CC        NN      NNNN DD      DD RR  RR   I11111 VV      VV EE
CC        NN      NN  DD      DD RR      RR I11111 VV      VV EE
CC        NN      NN  DD      DD RR      RR I11111 VV      VV EE
CCCCCCCC NN      NN DDDDDDDD RRR      RR I11111 VV      VV EEEEEEEEE RRR      RR
CCCCCCCC NN      NN DDDDDDDD RRR      RR I11111 VV      VV EEEEEEEEE RRR      RR

```

```

LL        I11111 SSSSSSSS
LL        I11111 SSSSSSSS
LL        I1      SS
LL        I1      SS
LL        I1      SS
LL        I1      SS
LL        I1      SSSSSS
LL        I1      SSSSSS
LL        I1      SS
LL        I1      SS
LL        I1      SS
LLLLLLLLLL I11111 SSSSSSSS
LLLLLLLLLL I11111 SSSSSSSS

```

(2)	116	External and local symbol definitions
(7)	365	Standard tables
(9)	452	P2 buffer verification tables
(10)	548	UNIT_INIT, Unit initialization routine
(11)	575	CLR_IRP - Initialize IRP fields
(12)	600	XMT_FDT, Transmit I/O Operation FDT Routine
(13)	657	RCV_FDT, Read I/O Operation FDT Routine
(16)	725	ALT_START, Alternate Start I/O Routine
(17)	765	XMT_START, Start Transmit Operation
(18)	825	RCV_START, Start Receive Operation
(19)	861	SETMODE_FDT, Set mode I/O operation FDT routine
(20)	963	Complete QIO request routines
(21)	1009	NEW_TRIB - Allocate and init new CDB
(22)	1112	SETMODE_CTRL, Perform setmode FDT operation on controller
(23)	1219	SENSEMODE_FDT, Sense Mode I/O operation FDT routine
(24)	1321	GET_CHAR_RBUF, Get P2 characteristics buffer for read
(24)	1322	GET_CHAR_WBUF, Get P2 characteristics buffer for write
(25)	1369	START_TRIB, Start tributary routine
(26)	1506	LIS_FORK, Listen action routine
(29)	1663	CANCEL, Cancel I/O routine
(30)	1701	CAN_DEV, Device shutdown routine
(31)	1748	ZAP_CDB, Shutdown the tributary
(32)	1880	MSG_FORK, Fork process for receipt of Seq Messages
(33)	1902	DG_FORK, Fork process for receipt of DG
(34)	1997	FINISH_RCV_IO, Finish receive I/O processing
(35)	2048	FILLRCVLIST, Fill receive buffer list
(35)	2049	ADDRCVLIST, Move IRP buffer to free list
(36)	2106	XLATE, Translate Channel to CDB address
(37)	2153	VALIDATE_P2, Validate P2 buffer parameters
(38)	2232	UNPACK_P2_BUF, Unpack a P2 parameter from P2 buffer
(39)	2273	CN_END, End of driver

```
0000 1 .TITLE CNDRIVER - VAX/VMS DECnet-CI Class Driver
0000 2 .IDENT 'V04-000'
0000 3
0000 4 *****
0000 5 *
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 * ALL RIGHTS RESERVED.
0000 9 *
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 * TRANSFERRED.
0000 16 *
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 * CORPORATION.
0000 20 *
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 *
0000 24 *
0000 25 *****
0000 26 **
0000 27 FACILITY:
0000 28
0000 29 VAX/VMS DECnet-CI class driver
0000 30
0000 31 ABSTRACT:
0000 32
0000 33 This module contains the DECnet-CI class driver FDI routines,
0000 34 SCS dispatcher, and fork routines.
0000 35
0000 36
0000 37
0000 38 Kerbey T. Altmann, 17-Aug-1981
0000 39
0000 40 MODIFIED BY:
0000 41
0000 42 V03-016 ADE3004 A. Eldridge 24-Jul-1984
0000 43 Change name back to DECNET$PHASE_III for now. The change in the
0000 44 name must be phased in by updating the receiver to accept the
0000 45 old and new name before the transmitter can be updated to send
0000 46 the new name. The new name should be DECNET$CI rather than
0000 47 DECNET$PHASE_IV.
0000 48
0000 49 V03-015 LMP0275 L. Mark Pilant, 12-Jul-1984 12:26
0000 50 Initialize the ACL info in the ORB to be a null descriptor
0000 51 list rather than an empty queue. This avoids the overhead
0000 52 of locking and unlocking the ACL mutex, only to find out
0000 53 that the ACL was empty.
0000 54
0000 55 V03-014 LMP0221 L. Mark Pilant, 26-Mar-1984 16:43
0000 56 Change UCBSL_OWNER to ORBSL_OWNER.
0000 57
```

```

0000 58 : V03-013 TMK0002      Todd M. Katz      24-Mar-1984
0000 59 :      When connecting to a remote station over a specific local
0000 60 :      port, which is what this DECnet class driver does, the name of
0000 61 :      the local port together with the remote station address must be
0000 62 :      specified as an argument to the CONNECT fork process call. This
0000 63 :      DECnet class driver was specifying the name of the local port
0000 64 :      as PAA. It should now be specifying the name of the local port
0000 65 :      as PAA0. If this is not done, the CONNECT will fail.
0000 66 :
0000 67 :      Change the SCS process name of the DECnet SYSAP from
0000 68 :      DECNET$PHASE_III to DECNET$PHASE_IV.
0000 69 :
0000 70 : V03-012 TMK0001      Todd M. Katz      08-Feb-1984
0000 71 :      Use the macro SEND_DG_BUF_REG to do transmits instead of
0000 72 :      SEND_DG_BUF. This allows me to remove the pseudo-CDRP which
0000 73 :      is currently buried within the CDB. This false CDRP was only
0000 74 :      being used to pass the application data and CDT addresses to
0000 75 :      the fork process call, FPC$SENDDG. The fork process call issued
0000 76 :      by SEND_DG_BUF_REG, FPC$SENDRGDG, requires these addresses to be
0000 77 :      in registers when it is invoked, and thus, doesn't require a
0000 78 :      CDRP in order to obtain them.
0000 79 :
0000 80 : V03-011 ADE3003      Alan D. Eldridge  19-May-1983
0000 81 :      Replaced constants with appropriate SBO$ symbols.
0000 82 :
0000 83 : V03-010 ADE3002      Alan D. Eldridge  19-Apr-1983
0000 84 :      Modified datagram internal SCS header "size" field to
0000 85 :      handle new negative offset processing option.
0000 86 :
0000 87 : V03-009 ADE3001      Alan D. Eldridge  2-Feb-1983
0000 88 :      Simplified connect/disconnect control. Removed the sending of
0000 89 :      XON/XOFF sequenced messages. Issue a CONNECT only if the
0000 90 :      remote sequence number is higher. Redefined the CDB.
0000 91 :
0000 92 : V03-008 NPK3010      N. Kronenberg    24-Nov-1982
0000 93 :      Removed output array specifier from CONFIG_SYS call.
0000 94 :
0000 95 : V03-007 KTA0109      Kerbey T. Altmann 8-Jul-1982
0000 96 :      Fix bug in returning buffer info in SENSEMODE.
0000 97 :
0000 98 : V03-006 NPK3004      N. Kronenberg    2-Jul-1982
0000 99 :      Modify START_TRIB to connect over specific virtual
0000 100 :      circuit instead of looking up the remote system and
0000 101 :      connecting to that.
0000 102 :
0000 103 : V03-005 NPK3003      N. Kronenberg    1-Jul-1982
0000 104 :      Fixed offsets from CONFIG_PTR/SYS for new format
0000 105 :      returned by those routines.
0000 106 :
0000 107 : V03-004 KDM0002      Kathleen D. Morse 28-Jun-1982
0000 108 :      Added $PRDEF.
0000 109 :
0000 110 : V03-003 KTA0097      Kerbey T. Altmann 20-Apr-1982
0000 111 :      Fix bad branch destination.
0000 112 :
0000 113 :
0000 114 :--

```

```
0000 116      .SBTTL External and local symbol definitions
0000 117
0000 118 :
0000 119 : System definitions
0000 120 :
0000 121 :
0000 122      $ADPDEF      : Adapter control block
0000 123      $CDTDEF      : Connection descriptor
0000 124      $CRBDEF      : Channel request block
0000 125      $CXBDEF      : Complex buffers
0000 126      $DCDEF       : Device classes and types
0000 127      $DDDEF       : Device data block
0000 128      $DEVDEF      : Device characteristics
0000 129      $DPTDEF      : Driver prologue table defs
0000 130      $DYNDEF      : Control block defs
0000 131      $IODEF       : I/O function codes
0000 132      $IPLDEF      : Hardware IPL definitions
0000 133      $JIBDEF      : Job info block
0000 134      $IRPDEF      : I/O request packet
0000 135      $NMADEF      : Network Management definitions
0000 136      $ORBDEF      : Object's Rights Block
0000 137      $PBDEF       : Path block definitions
0000 138      $PCBDEF      : Process control block
0000 139      $PDTDEF      : Port Descriptor Table
0000 140      $PRDEF       : Processor register definitions
0000 141      $SBDEF       : System block definitions
0000 142      $SBODEF      : System block output definitions
0000 143      $SSDEF       : System status codes
0000 144      $UCBDEF      : Unit control block
0000 145      $VECDEF      : Interrupt vector block
0000 146      $XMDEF       : XMDRIVER symbols
0000 147
0000 148 :
0000 149 : Local macros
0000 150 :
0000 151 .MACRO SETBIT POS,BAS,?L      : Set a single bit
0000 152      BBSS      POS,BAS,L
0000 153      L:
0000 154 .ENDM SETBIT
0000 155
0000 156 .MACRO CLRBIT POS,BAS,?L      : Clear a single bit
0000 157      BBCC      POS,BAS,L
0000 158      L:
0000 159 .ENDM CLRBIT
0000 160
0000 161 .MACRO PUSHQ ARG      : Push a quadword
0000 162      MOVQ      ARG,-(SP)      : Save argument on stack
0000 163 .ENDM PUSHQ
0000 164
0000 165 .MACRO POPQ ARG      : Pop a quadword
0000 166      MOVQ      (SP)+,ARG      : Restore argument
0000 167 .ENDM POPQ
0000 168
```

```

0000 170
0000 171 $DEFINI PARAM
0000 172
00000004 0000 173     COUNT_C_ENTRY = 2*2           ; COUNT table entry size
00000000C 0000 174     PARAM_C_ENTRY = 2*6           ; PARAM table entry size
0000 175     _VIELD_PRM,0,<-           ; Parameter bits and sizes
0000 176         <TYPE,12,M>,-         ; Parameter type
0000 177         <MIN,1,M>,-           ; Parameter minimum value
0000 178         <MAX,1,M>,-           ; Parameter maximum value
0000 179         <REQUIRED,1,M>,-       ; Parameter required flags
0000 180         <INVALID,1,M>,-       ; Parameter invalid flags
0000 181     >
0000 182
0000 183     _VIELD_OFF,0,<-           ; Offset word fields
0000 184         <VALUE,10,M>,-         ; Offset value
0000 185         <WIDTH,6,M>,-         ; Width of field in structure
0000 186     >
0000 187
0000 188 $DEFEND PARAM
0000 189
0000 190 .MACRO  PARAM  TYPE,OFFSET,WIDTH=0,MIN=0,MAX=-1,REQUIRED,INVALID
0000 191             ; NOTE - The REQUIRED field can only check 1 bit!
0000 192
0000 193     $$$NUM = $$$NUM+1           ; Count number of time executed
0000 194     $$$TYP = NMASC 'TYPE & PRM_M_TYPE ; Isolate type
0000 195     $$$OFF = OFFSET & OFF_M_VALUE  ; Isolate offset only
0000 196
0000 197     .IIF NOT_BLANK <MIN>,<MIN>,<MIN> $$$TYP = $$$TYP ! PRM_M_MIN
0000 198     .IIF NOT_BLANK <MAX>,<MAX>,<MAX> $$$TYP = $$$TYP ! PRM_M_MAX
0000 199     .IIF NOT_BLANK <REQUIRED>,<REQUIRED> $$$TYP = $$$TYP ! PRM_M_REQUIRED
0000 200     .IIF NOT_BLANK <INVALID>,<INVALID> $$$TYP = $$$TYP ! PRM_M_INVALID
0000 201
0000 202     .WORD  $$$TYP
0000 203     .WORD  $$$OFF ! <WIDTH @ OFF_V_WIDTH>
0000 204     .WORD  MIN
0000 205     .WORD  MAX
0000 206     .WORD  REQUIRED
0000 207     .WORD  INVALID
0000 208 .ENDM  PARAM
0000 209
0000 210
0000 211 .MACRO  COUNT  TYPE,OFFSET,WIDTH=32
0000 212
0000 213     $$$NUM = $$$NUM+1           ; Bump number of time executed
0000 214     $$$OFF = OFF_M_VALUE & OFFSET  ; Isolate offset only
0000 215     $$$TYP = PRM_M_TYPE & NMASC_'TYPE ; Isolate type
0000 216
0000 217     .IIF IDN, WIDTH, 8, $$$TYP = $$$TYP ! <1@NMASV_CNT_WID>
0000 218     .IIF IDN, WIDTH, 16, $$$TYP = $$$TYP ! <2@NMASV_CNT_WID>
0000 219     .IIF IDN, WIDTH, 32, $$$TYP = $$$TYP ! <3@NMASV_CNT_WID>
0000 220
0000 221
0000 222     .WORD  $$$TYP ! NMASM_CNT_COU
0000 223     .WORD  $$$OFF ! <WIDTH @ OFF_V_WIDTH>
0000 224 .ENDM  COUNT
0000 225
0000 226 .MACRO  START_TABLE  NAME           ; Start Table declaration

```

```
0000 227 $$$NUM = 0 ; Init number of entries
0000 228 'NAME' TABLE = . ; Define begining of table
0000 229 .ENDM START_TABLE
0000 230
0000 231 .MACRO END TABLE NAME ; Terminate Table declaration
0000 232 :WORD 0 ; Create marker
0000 233 'NAME' NUM = $$$NUM ; Number of entries
0000 234 .ENDM END_TABLE
0000 235
0000 236 ;
0000 237 ; Local symbols
0000 238 ;
0000 239 ;
0000 240 ;
0000 241 ; $QIO parameter offsets
0000 242 ;
00000000 0000 243 P1 = 0 ; Parameter 1
00000004 0000 244 P2 = 1*4 ; Parameter 2
0000 245 ;
0000 246 ;
0000 247 ; Other constants
0000 248 ;
00000009 0000 249 RBFMIN = 9 ; Min size of CDB_B_RCV_CNT
0000001F 0000 250 RBFMAX = 31 ; Max size of CDB_B_RCV_CNT
00000006 0000 251 RBFTHR = 6 ; CND_B_RCV_FQ threshold. Below this
0000 252 ; signal XMSM_STS_BUFFAIL in IOST2
00000010 0000 253 MAX_TRB = 16 ; Max tributaries on CI device
```



```

0000 255 ;
0000 256 ; Overlays of IRP
0000 257 ;
0000 258 ASSUME IRP$L_SEGVBN EQ IRP$Q_NT_PRVMSK+8
0000 259
0000 260 $DEFINI IRP
0000 261
00000040 0000 262 = IRP$Q_NT_PRVMSK ; Overlay network priv mask
0040 263 $DEF IRP$B_INDEX .BLKB 1 ; Vector index for CDB
0041 264
00000054 0041 265 = IRP$L_EXTEND
0054 266 $DEF IRP$L_CDB .BLKL 1
0058 267
0058 268 $DEFEND IRP ; End of IRP overlays
0000 269
0000 270
0000 271 ;
0000 272 ; Definitions that follow the standard UCB fields
0000 273 ;
0000 274
0000 275 $DEFINI UCB ; Start of UCB definitions
0000 276
00000090 0000 277 . = UCB$C_LENGTH ; Position at end of UCB
0090 278
0090 279 $DEF UCB$L_LIS_CDT .BLKL 1 ; Addr of listening CDT
0094 280 $DEF UCB$L_TWIR_CDT .BLKL 1 ; Addr of loopbacked accept CDT
0098 281 $DEF UCB$L_DGHDRSZ .BLKL 1 ; Size of the SCS header for DG's
009C 282 $DEF UCB$W_DUMMY .BLKW 1 ; Dummy location for unwanted param's
009E 283 $DEF UCB$B_CN_PORT .BLKB 1 ; Our port number
009F 284 $DEF UCB$B_RCV_CNT .BLKB 1 ; Number of receive buffers
00A0 285 $DEF UCB$L_VEC_CDB .BLKL MAX_TRB ; CDB address vector
00E0 286 $DEF UCB$W_VEC_CHAN .BLKW MAX_TRB ; User channel lookup vector
0100 287
00000100 0100 288 UCB$C_CN_LENGTH = <.+15>&-16 ; Size of UCB padded to a quadword
0100 289
0100 290 ;
0100 291 ; Define device status bits
0100 292 ;
0100 293 $VIELD UCB,0,<- ; CNDRIVER UCB$W_DEVSTS bits
0100 294 > <CN_INITED,,M>,- ; Device init'ed
0100 295 ;
0100 296 $DEFEND UCB ; End of UCB definitions
0000 297

```

```

0000 299 :
0000 300 : CNDRIVER CDB definitions
0000 301 :
0000 302 $DEFINI CDB
0000 303 :
0000 304 $DEF CDB_Q_FORK .BLKQ 1 : Fork Queue Linkage
0008 305 $DEF CDB_W_SIZE .BLKW 1 : Structure size
000A 306 $DEF CDB_B_TYPE .BLKB 1 : Structure type
000B 307 $DEF CDB_B_FIPL .BLKB 1 : Fork IPL (not UCB FIPL)
00000006 000C 308 CDB_C_FIPL = 6 : Must be less than SCS's IPL (8)
000C 309 $DEF CDB_L_FPC .BLKL 1 : Fork PC
0010 310 $DEF CDB_L_FR3 .BLKL 1 : Fork R3
0014 311 $DEF CDB_L_FR4 .BLKL 1 : Fork R4
0018 312 :
0018 313 $DEF CDB_Q_XMT_IRP .BLKQ 1 : Transmit IRP's awaiting completion
0020 314 $DEF CDB_Q_RCV_IRP .BLKQ 1 : Receive IRP's awaiting buffers
0028 315 $DEF CDB_Q_RCV_MSG .BLKQ 1 : Receive buffers containing messages
0030 316 :
0030 317 $DEF CDB_L_SETMODE .BLKL 1 : Ptr to IOS SETMODE
0034 318 $DEF CDB_L_ABSTIME .BLKL 1 : Time last DISCONNECT was issued
0038 319 $DEF CDB_W_BUFSIZ .BLKW 1 : Buffer size
003A 320 $DEF CDB_W_STS .BLKW 1 : Circuit status
003C 321 $DEF CDB_B_RCV_CNT .BLKB 1 : Receive buffer count
003D 322 $DEF CDB_B_RCV_FQ .BLKB 1 : Receive buffers on free queue
003E 323 $DEF CDB_B_TRB_ADDR .BLKB 1 : Tributary address
003F 324 $DEF CDB_B_STA .BLKB 1 : Circuit state
0040 325 :
0040 326 :
0040 327 : Circuit counters
0040 328 :
0040 329 :
0040 330 $DEF CDB_L_BRC .BLKL 1 : Receive byte count
0044 331 $DEF CDB_L_BSN .BLKL 1 : Transmit byte count
0048 332 $DEF CDB_L_DBR .BLKL 1 : Data buffers received
004C 333 $DEF CDB_L_DBS .BLKL 1 : Data buffers sent
0050 334 :
0050 335 $DEF CDB_L_UCB .BLKL 1 : Addr of UCB
0054 336 $DEF CDB_L_CDT .BLKL 1 : Ptr to CDT
0058 337 $DEF CDB_B_REMVER .BLKB 1 : Remote's protocol version
0059 338 $DEF CDB_B_REMSYS .BLKB 1 : Remote's operating system
00000058 005A 339 CDB_W_REMPROT = CDB_B_REMVER : Label combining two fields above
005A 340 $DEF CDB_B_DUMMY .BLKB 1 : Dummy location for unwanted param's
005B 341 $DEF CDB_B_RSTCNT .BLKB 1 : Restart counter for slowing down
005C 342 :
00000060 005C 343 CDB_C_LENGTH = <. +15>&-16 : restart frequency
005C 344 : Pad structure out to a quadword
005C 345 :
005C 346 : Define status bits used in CDB_W_STS and CDB_B_STA values
005C 347 :
005C 348 _VIELD CDB,0,<- : Tributary status bits for CDB_W_STS
005C 349 <RUN,,M>,- : Tributary is in RUN state
005C 350 <CONN,,M>,- : Call to CONNECT pending
005C 351 <ACPT,,M>,- : CALL to ACCEPT pending
005C 352 <DISC,,M>,- : Call to DISCONNECT or FORK pending
005C 353 <REJECT,,M>,- : Call to REJECT pending
005C 354 >
005C 355

```

CNDRIVER  
V04-000

- VAX/VMS DECnet-CI Class Driver 1 10  
External and local symbol definitions

16-SEP-1984 01:19:27 VAX/VMS Macro V04-00 Page 8  
5-SEP-1984 00:11:06 [DRIVER.SRC]CNDRIVER.MAR;1 (6)

```
00000000 005C 356 CDB_C_IDLE = 0
00000001 005C 357 CDB_C_OPEN = 1
00000002 005C 358 CDB_C_CONN = 2
00000003 005C 359 CDB_C_LSTN = 3
00000004 005C 360 CDB_C_ACPT = 4
          005C 361
          005C 362 $DEFEND CDB
          0000 363
```

```
: Tributary is idle
: Tributary connection has been made
: Tributary has CONNECT pending
: Tributary is listening for connect
: Tributary has ACCEPT pending
```

```

0000 365      .SBTTL Standard tables
0000 366
0000 367      :
0000 368      : Driver prologue table
0000 369      :
0000 370
0000 371      DPTAB      -      : DPT-creation macro
0000 372      END      = CN_END,-      : End of driver label
0000 373      ADAPTER = NUCL,-      : Adapter type
0000 374      FLAGS  = DPTSM_SCS,-      : Driver requires SCS
0000 375      UCBSIZE = UCBS%CN_LENGTH,-      : Length of UCB
0000 376      NAME   = CNDRIVER,-      : Driver name
0000 377
0038 378      DPT_STORE INIT      : Start of load
0038 379      : initialization table
0038 380      DPT_STORE UCB,UCBSB_FIPL,B,8      : Device fork IPL
003C 381      DPT_STORE UCB,UCBSB_DIPL,B,8      : Device interrupt IPL
0040 382      DPT_STORE ORB,ORBSB_FLAGS,B,-      : Protection block flags
0040 383      <ORBSM_PROT_16>      : SOGW protection word
0044 384      DPT_STORE ORB,ORBSW_PROT,W,0      : default protection
0049 385      DPT_STORE ORB,ORBSL_OWNER,L,<^X010001>      : [1,1] owns the device
0050 386      DPT_STORE UCB,UCBSL_DEVCHAR,L,-      : Device characteristics
0050 387      <DEVSM_NET!-      : e.g., network device
0050 388      DEVSM_REC!-      : record oriented
0050 389      DEVSM_IDV!-      : input device
0050 390      DEVSM_ODV-      : output device
0050 391      >
0057 392      DPT_STORE UCB,UCBSB_DEVCLASS,B,DCS_SCOM      : Sample device class
0058 393      DPT_STORE UCB,UCBSW_DEVBUSIZ,W,-      : Default buffer size
0058 394      SCSSGW_MAXDG
0062 395
0062 396      DPT_STORE REINIT      : Start of reload
0062 397      : initialization table
0062 398      DPT_STORE DDB,DDBSL_DDT,D,CNSDDT      : Address of DDT
0067 399      DPT_STORE CRB,-      : Address of device
0067 400      CRBSL_INTD+VECSL_UNITINIT,-      : unit initialization
0067 401      D,UNIT_INIT      : routine
006C 402
006C 403      DPT_STORE END      : End of initialization
0000 404      : tables

```

```

0000 406 :
0000 407 : Driver dispatch table
0000 408 :
0000 409 : DDTAB - ; DDT-creation macro
0000 410 : DEVNAM = CN,- ; Name of device
0000 411 : FUNCTB = CN_FUNCTABLE,- ; FDT address
0000 412 : CANCEL = CANCEL,- ; Cancel I/O routine
0000 413 : ALTSTART= ALT_START ; Alternate start I/O
0038 414 :
0038 415 : Function dispatch table
0038 416 :
0038 417 CN_FUNCTABLE: ; FDT for driver
0038 418 FUNCTAB ,- ; Valid I/O functions
0038 419 <READLBLK,- ; Read logical
0038 420 WRITELBLK,- ; Write logical
0038 421 SETMODE,- ; Set device mode
0038 422 SENSEMODE,- ; Sense mode
0038 423 SETCHAR - ; Set device chars.
0038 424 >
0040 425 FUNCTAB ,- ; Buffered functions:
0040 426 <READLBLK,- ; Read logical
0040 427 WRITELBLK,- ; Write logical
0040 428 SETMODE,- ; Set device mode
0040 429 SENSEMODE,- ; Sense mode
0040 430 SETCHAR - ; Set device chars.
0040 431 >
0048 432 FUNCTAB CLR_IRP,- ; Init IRP fields
0048 433 <READLBLK,- ; Read logical
0048 434 WRITELBLK,- ; Write logical
0048 435 SETMODE,- ; Set device mode
0048 436 SENSEMODE,- ; Sense mode
0048 437 SETCHAR - ; Set device chars.
0048 438 >
0054 439 FUNCTAB RCV_FDT,- ; FDT read routine for
0054 440 <READLBLK,- ; read logical,
0054 441 >
0060 442 FUNCTAB XMT_FDT,- ; FDT write routine for
0060 443 <WRITELBLK,- ; write logical,
0060 444 >
006C 445 FUNCTAB SETMODE_FDT,- ; FDT set mode routine
006C 446 <SETMODE,- ; set mode
006C 447 SETCHAR - ; set characteristics
006C 448 >
0078 449 FUNCTAB SENSEMODE_FDT,- ; FDT sense mode routine
0078 450 <SENSEMODE> ; for sensemode

```

```

0084 452          .SBTTL P2 buffer verification tables
0084 453
0084 454      :
0084 455      : Define CDB parameters
0084 456      :
0084 457  START_TABLE TRIB_PRM      ; Start of tributary parameter table
0084 458
0084 459          PARAM PCCI_MST, OFFSET = CDB_B_DUMMY,-          ; Trib maint state
0084 460          WIDTH  = 0,-          ; Dummy location
0084 461          MIN    = NMASC_STATE_ON,-
0084 462          MAX    = NMASC_STATE_OFF,-
0084 463          REQUIRED= 0,-
0084 464          INVALID = CDB_M_RUN
0090 465
0090 466          PARAM PCCI_TRI, OFFSET = CDB_B_TRB_ADDR,-      ; Trib address
0090 467          WIDTH  = 8,-
0090 468          MIN    = 0,-
0090 469          MAX    = 15,-
0090 470          REQUIRED= 0,-
0090 471          INVALID = CDB_M_RUN
009C 472
009C 473          PARAM PCCI_MRB, OFFSET = CDB_B_RCV_CNT,-      ; Trib max buf
009C 474          WIDTH  = 8,-
009C 475          MIN    = 0,-
009C 476          MAX    = 255,-
009C 477          REQUIRED= 0,-
009C 478          INVALID = CDB_M_RUN
00A8 479
00A8 480  END_TABLE  TRIB_PRM      ; End of tributary parameter table
00AA 481
00AA 482      :
00AA 483      : Define UCB parameters
00AA 484      :
00AA 485      :
00AA 486  START_TABLE LINE_PRM     ; Start of device parameter table
00AA 487
00AA 488          PARAM PCLI_DUP, OFFSET = UCBSW_DUMMY,-          ; Duplex
00AA 489          WIDTH  = 0,-          ; Dummy location
00AA 490          MIN    = NMASC_DPX_FUL,-
00AA 491          MAX    = NMASC_DPX_HAL,-
00AA 492          REQUIRED= 0,-
00AA 493          INVALID = UCBSM_CN_INITED
00B6 494
00B6 495          PARAM PCLI_CON, OFFSET = UCBSW_DUMMY,-          ; Controller mode
00B6 496          WIDTH  = 0,-          ; Dummy location
00B6 497          MIN    = NMASC_LINCN_NOR,-
00B6 498          MAX    = NMASC_LINCN_LOO,-
00B6 499          REQUIRED= 0,-
00B6 500          INVALID = UCBSM_CN_INITED
00C2 501
00C2 502          PARAM PCLI_BUS, OFFSET = UCBSW_DEVBUFSIZ,-      ; Block size
00C2 503          WIDTH  = 16,-
00C2 504          MIN    = 32,-
00C2 505          MAX    = 948,-
00C2 506          REQUIRED= 0,-
00C2 507          INVALID = UCBSM_CN_INITED
00CE 508

```

```
00CE 509          PARAM PCLI_BFN, OFFSET = UCBSB_RCV_CNT,-          ; Maximum receive buffers
00CE 510                WIDTH  = 8,-
00CE 511                MIN    = 1,-
00CE 512                MAX    = 255,-
00CE 513                REQUIRED = 0,-
00CE 514                INVALID = UCBSM_CN_INITED
00DA 515
00DA 516  END_TABLE LINE_PRM          ; End of device parameter tables
00DC 517
00DC 518 ;
00DC 519 ; Tributary counter type codes
00DC 520 ;
00DC 521  START_TABLE TRIB_CNT          ; Start of Tributary COUNTER table
00DC 522
00DC 523                COUNT  CTCIR_BRC, WIDTH=32, OFFSET=CDB_L_BRC    ; Bytes received
00E0 524                COUNT  CTCIR_BSN, WIDTH=32, OFFSET=CDB_L_BSN    ; Bytes sent
00E4 525                COUNT  CTCIR_DBR, WIDTH=32, OFFSET=CDB_L_DBR    ; Data blocks received
00E8 526                COUNT  CTCIR_DBS, WIDTH=32, OFFSET=CDB_L_DBS    ; Data blocks sent
00EC 527
00EC 528  END_TABLE  TRIB_CNT          ; End of Tributary COUNTER table
00EE 529
00EE 530
00EE 531  START TABLE LINE_CNT          ; Start of device COUNTER table
00EE 532  END_TABLE  LINE_CNT          ; - null table
00FO 533
00FO 534 ;
00FO 535 ; Our SCS process name and connect data
00FO 536 ;
00FO 537  PROC_C_NAM = 6                  ; How much of PROC_NAM must match
00FO 538  PROC_NAM:
45 53 41 48 50 24 54 45 4E 43 45 44 00FO 539  .ASCII 'DECNET$PHASE_III'          ; How SCS knows us -- 16 characters long
49 49 49 5F 00FO
00FC
0100 540  CONN_DATA:
01 0100 541  .BYTE 1                  ; Protocol version
00 0101 542  .BYTE 0                  ; Operating system (VMS) id
00'00'00'00'00'00'00'00'00'00'00'00'00'00'00'00'00'00' 0102 543  .BYTE 0[14]          ; Remaining fields must be zero
00'00' 010E
0110 544
00000000 0110 545  OLD_C_PROT = 0          ; Use for original protocol
0110 546
```

```
0110 548 .SBTTL UNIT_INIT, Unit initialization routine
0110 549
0110 550 :++
0110 551 : UNIT_INIT - Readies unit for I/O operations
0110 552 :
0110 553 :
0110 554 : The operating system calls this routine after calling the
0110 555 : controller initialization routine:
0110 556 :
0110 557 : - at system startup
0110 558 : - during driver loading
0110 559 : - during recovery from a power failure
0110 560 :
0110 561 : The unit is put online.
0110 562 :
0110 563 : Inputs: R5 = UCB address
0110 564 :
0110 565 : Outputs: All registers are preserved
0110 566 :
0110 567 :
0110 568 :--
0110 569
0110 570 UNIT_INIT: ; Initialize unit
64 A5 10 A8 0110 571 BISW #UCBSM_ONLINE,UCBSW_STS(R5) ; Set unit online
05 0114 572 RSB ; Return
0115 573
```



```
0115 575 .SBTTL CLR_IRP - Initialize IRP fields
0115 576
0115 577 :++
0115 578 : CLR_IRP - Initialize IRP fields
0115 579
0115 580
0115 581 : Selected IRP fields are initialized. The function code with modifiers
0115 582 : is setup.
0115 583
0115 584 : Inputs: R3 IRP address
0115 585
0115 586 : Outputs: All other registers are preserved.
0115 587
0115 588 : IPL may be FIPL or ASTDEL
0115 589
0115 590 :--
0115 591 CLR_IRP:
38 A3 7C 0115 592 CLRQ IRP$I_OST1(R3) ; Initialize IRP fields
2C A3 D4 0118 593 CLRL IRP$I_SVAPTE(R3) ; Clear IOSB image
30 A3 B4 011B 594 CLRW IRP$I_BOFF(R3) ; Init buffer pointer
54 A3 D4 011E 595 CLRL IRP$I_CDB(R3) ; No quota to return yet at I/O post
40 A3 7C 0121 596 CLRQ IRP$I_INDEX(R3) ; No CDB yet
05 0124 597 RSB ; No trib i.d. yet
0125 598 ; Done
```





			0190	700		
			0190	701	XMT_RCV_FDT CO:	
51	50	14	3C	0190	MOVZWL	S^#SS\$_BADPARAM,R0 ; Assume bad parameters
	04	AC	3C	0193	MOVZWL	P2(AP),R1 ; Get buffer size
		12	13	0197	BEQL	10\$ ; If zero, abort I/O request
42	A5	51	B1	0199	CMPL	R1,UCB\$W_DEVBUSIZ(R5) ; Is buffer too big?
		0C	1A	019D	BGTRU	10\$ ; If GTRU yes, abort I/O request
	50	6C	D0	019F	MOVL	P1(AP),R0 ; Get user buffer virt address
3C	A3	50	D0	01A2	MOVL	R0,IRP\$L_IOST2(R3) ; Save it for MOVC
		9E	16	01A6	JSB	@(\$?)+ ; Call back our caller
	03	50	E8	01A8	BLBS	R0,20\$ ; If LBS, continue
		010D	31	01AB	BRW	ABORT_REQ ; Abort the request
				01AE	712	10\$:
				01AE	713	20\$:
				01AE	714	;
				01AE	715	Okay so far. Setup to return to EXE\$QIORETURN -- which returns to
				01AE	716	the user with SS\$NORMAL in R0. This means that all subsequent
				01AE	717	errors must be reported via the IOSB.
				01AE	718	;
00000000'GF			9F	01AE	719	PUSHAB G^EXE\$QIORETURN ; Setup return address on stack
				01B4	720	SETIPL UCB\$B_FIPL(R5) ; Raise IPL to fork level
				01B8	721	;
				01B8	722	;
				01B8	723	Fall thru to ALT_START ; to lock the data base
						;

```

0188 725      .SBTTL ALT_START, Alternate Start I/O Routine
0188 726
0188 727      :++
0188 728      : ALT_START - Alternate Start I/O Routine
0188 729      :
0188 730      :
0188 731      : This entry point is used to dispatch IOS_READBLK and IOS_WRITEBLK requests.
0188 732      : The IRP is either built by our own FDI routines, or by some higher level
0188 733      : Executive agent (e.g, NETDRIVER). All I/O status, including errors, must
0188 734      : be passed via IOPOST in the IOSB.
0188 735      :
0188 736      :
0188 737      : NOTE: The CHAN field of the IRP is sufficient to map to a CDB.
0188 738      :
0188 739      : Inputs:      R3 - IRP address
0188 740      :              R5 - UCB address
0188 741      :
0188 742      : All pertinent fields of the IRP are assumed to be valid.
0188 743      :
0188 744      : IPL = FIPL
0188 745      :
0188 746      : Outputs:   R0-R4  Garbage
0188 747      :
0188 748      :--
0188 749      ALT_START:
0210 8F  BB 0188 750      PUSHR  #*M<R4,R9>      : Save reg
0188 751      BSBB   S$      : Process request
0210 8F  BA 018E 752      POPR   #*M<R4,R9>      : Restore regs
0188 753      RSB      : Return to caller with garbage in R0
0188 754      :
0188 755      S$:      BSBW   XLATE      : Get CDB from IRP$W_CHAN
0188 756      BLBC   R0,ABORT_START : If LBC then error
0188 757      ASSUME  CDB_V_RUN EQ 0
0188 758      BLBC   CDB_W_STS(R9),ABORT_START : If LBC then not in RUN state
0188 759      BBS     #IRP$V_FUNC -
0188 760      IRP$W_STS(R3),RCV_START : If BS then IOS_READ else IOS_WRITE
0188 761      :
0188 762      : Fall thru to XMT_START
0188 763      :
  
```

```

01D2 765      .SBTTL XMT_START, Start Transmit Operation
01D2 766
01D2 767      :++
01D2 768      : XMT_START - Start Transmit Operation
01D2 769      :
01D2 770      :
01D2 771      : This routine is called to start a transmit operation. The tributary is
01D2 772      : known to be up and running at this point. All status must be returned via
01D2 773      : the IOSB.
01D2 774      :
01D2 775      :
01D2 776      : Inputs:      R3 = IRP address
01D2 777      :                R5 = UCB address
01D2 778      :                R9 = CDB address
01D2 779      :
01D2 780      :                IPL = FIPL
01D2 781      :
01D2 782      : Outputs:     R0 = status of transmit request
01D2 783      :
01D2 784      :                R5-R7 are preserved.
01D2 785      :
01D2 786      :--
01D2 787      XMT_START:
51 32 A3 3C 01D2 788      MOVZWL  IRPSL_BCNT(R3),R1      ; Pick up length
50 2C A3  D0 01D6 789      MOVL    IRPSL_SVAPTE(R3),R0      ; Pick up head of buffer
   52 60  D0 01DA 790      MOVL    (R0),R2                ; Get beginning of user message
01DD 791      10$:
01DD 792      : Add CI padding to keep beginning quadword aligned
01DD 793      :
   52 07 93 01DD 794      BITB    #^X<07>,R2      ; Need padding ?
   72 07 13 01E0 795      BEQL    20$                ; If EQL no
   72 01 8E 01E2 796      MNEGB   #1,-(R2)        ; Pad
   51 D6 01E5 797      INCL    R1                ; Adjust byte count
   F4 11 01E7 798      BRB     10$
01E9 799      20$:
01E9 800      : Send it to SCS requesting that the datagram be returned when done.
01E9 801      :
   38 A9 51 B1 01E9 802      PUSHQ   R2                ; Save user msg & IRP addresses
   52 31 1A 01EC 803      CMPW    R1,CDB_W_BUFSIZ(R9)    ; Msg size within bounds?
54 52 20 C2 01F0 804      BGTRU   60$                ; If GTRU then no
   28 54 D1 01F2 805      SUBL    #32,R2            ; Go to beginning of PPD header
   08 A2 54 AE 01F5 806      SUBL3   R0,R2,R4          ; Get offset to top of buffer
   OA A2 38 B0 01F9 807      CMPL   R4,#CXBSC_HEADER-32 ; Is header big enough ?
54 0084 C5 D0 01FC 808      BLSS   60$                ; If LSS then header too small
   08 50 E9 01FE 809      MNEGW   R4,8(R2)        ; Neg. offset to top of buffer
   0A A2 38 B0 0202 810      MOVW   #DYN$C_CIDG,10(R2) ; Structure type
   08 50 E9 0206 811      MOVL   UCBSL_PDT(R5),R4   ; Recover the PDT
   08 50 E9 0208 812      SEND_DG_BUF_REG #1,-    ; Control returns immediately
   08 50 E9 0208 813      CDf=CDB_L_CDT(R9),BUFFER=(SP)
   08 50 E9 0218 814      BLBC   R0,60$          ; If LBC, datagram not queued
   1C B9 63 0E 021B 815      POPQ   R2                ; Restore IRP address
   05 05 0E 021E 816      INSQUE (R3),@CDB_Q_XMT_IRP+4(R9) ; Queue IRP
   05 05 0E 0222 817      40$:  RSB                ; Return to await completion
   05 05 0E 0223 818      :
   05 05 0E 0223 819      60$:  POPQ   R2                ; Restore IRP address
   05 05 0E 0226 820      :
   05 05 0E 0226 821      ABORT_START:

```

CNDRIVER  
V04-000

- VAX/VMS DECnet-CI Class Driver H 11  
XMT\_START, Start Transmit Operation

16-SEP-1984 01:19:27  
5-SEP-1984 00:11:06

VAX/VMS Macro V04-00  
[DRIVER.SRC]CNDRIVER.MAR;1

Page 20  
(17)

06B6 31 0226 822 BRW ABORT\_IRP\_POST  
0229 823

; Report SSS\_ABORT via IOSB

```

0229 825      .SBTTL RCV_START, Start Receive Operation
0229 826
0229 827      :++
0229 828      : RCV_START - Start Receive Operation
0229 829
0229 830      :
0229 831      : This routine is called to start a receive operation. The tributary is
0229 832      : known to be up and running at this point. If IRP$L_SVAOTE is none zero
0229 833      : then it is assumed to be system buffer to be added to the receive free list.
0229 834      : All status must be returned via the IOSB.
0229 835      :
0229 836      : Inputs:      R3 = IRP address
0229 837      :           R5 = UCB address
0229 838      :           R9 = CDB address
0229 839      :
0229 840      :           IPL = FIPL
0229 841      :
0229 842      : Outputs:    R0 = return status of receive request
0229 843      :
0229 844      :--
0229 845      RCV_START:
54 59 D0 0229 846      MOVL   R9,R4          ; Transfer CDB pointer
06D3 30 022C 847      BSBW   ADDRCLIST      ; Add IRP buffer to free list
022F 848      :
022F 849      : Check to see if message is available
022F 850      :
52 28 B9 OF 022F 851      REMQUE  @CDB_Q_RCV_MSG(R9),R2    ; Dequeue a received message
03 1D 0233 852      BVS    100$          ; Br if none
0677 31 0235 853      BRW    FINISH_RCV_IO      ; Complete the I/O request
0238 854 100$:      :
0238 855      : No message available. Queue IRP to await arrival of message.
0238 856      :
24 B9 63 OE 0238 857      INSQUE  (R3),@CDB_Q_RCV_IRP+4(R9) ; Queue IRP to await message
05 023C 858      RSB                    ; Return
023D 859

```



```

023D 861      .SBTTL SETMODE_FDT, Set mode I/O operation FDT routine
023D 862
023D 863      :++
023D 864      : SETMODE_FDT - Set mode I/O operation FDT routine
023D 865
023D 866
023D 867      : Setup control parameters.  Optionally startup/shutdown the device or one
023D 868      : of the tributaries.  The subfunction modifiers are as follows:
023D 869
023D 870      :     IOSM_CTRL      - If set, request is for device.  Else, for tributary.
023D 871      :     IOSM_STARTUP   - Start device or establish tributary connection.
023D 872      :     IOSM_SHUTDOWN  - Shutdown device or disconnect tributary.
023D 873
023D 874
023D 875      : The QIO parameter for SETMODE is:
023D 876
023D 877      :     P2 = Optional address of buffer descriptor for extended characteristics
023D 878
023D 879
023D 880      : Inputs:
023D 881      :     R3 = IRP address
023D 882      :     R4 = PCB address
023D 883      :     R5 = UCB address
023D 884      :     R6 = CCB address
023D 885      :     R7 = Function code
023D 886      :     AP = address of first QIO parameter
023D 887      : Outputs:
023D 888      :     R0 = status of setmode request
023D 889
023D 890      :     R3-R5 are preserved.
023D 891      :     R7-R9 = destroyed
023D 892
023D 893      :-- SETMODE_FDT:                                     ; Setmode FDT processing
023D 894
023D 895      :     Copy the characteristics buffer, if any.  No return on error.
023D 896      :     On return, there's a buffer attached to IRP$L SVAPTE containing a
023D 897      :     copy of the user buffer -- hence we cannot "abort" the QIO passed
023D 898      :     this point but must return all errors via the IOSB.
023D 899
023D 900      :     Upon return, the IPL has been raised to FIPL
023D 901
023D 902      : BSBW  GET_CHAR_WBUF                                     ; Get P2 characteristics buffer
023D 903      :                                     ; - no return on error
023D 904      : MOVZWL IRP$W_FUNC(R3),R7                               ; Get full function code.
023D 905      : BBC   S^#IOSV_CTRL,R7,10$                               ; Br if not controller request
023D 906      : BRW   SETMODE_CTRL                                     ; Process controller request
023D 907      : 10$:
023D 908      :     Perform setmode request on a tributary
023D 909
023D 910      : BSBW  XLATE                                             ; Get CDB address if any
023D 911      : BBC   S^#IOSV_SHUTDOWN,R7,40$                          ; Branch if not trib shutdown
023D 912
023D 913      :     Shutdown tributary modifier specified -- always successful.
023D 914      :     Shutdown may complete ahead of other queued I/O for this tributary.
023D 915
023D 916      : BLBC  R0,FINISH_SUC                                     ; If LBC then no CDB
023D 917      : BSBW  ZAP_CDB_R9                                       ; Do the dirty work
  
```

```

029D 30
57 20 A3 3C 0240 904
03 57 09 E1 0244 905
0113 31 0248 906
024B 907
024B 908
024B 909
08 57 0715 30 024B 910
07 07 E1 024E 911
0252 912
0252 913
0252 914
0252 915
6C 50 E9 0252 916
0513 30 0255 917
  
```

```

        67 11 0258 918 BRB FINISH_SUC ; Always return 'success'
          025A 919 40$:
          025A 920 : IOSM_STARTUP tributary modifier specified or no modifier.
          025A 921 : Validate the P2 buffer and its contents.
          025A 922 :
51 FE26 CF 9E 025A 923 MOVAB TRIB_PTM_TABLE,R1 ; Set address of verification table
          52 D4 025F 924 CLRL R2 ; No status flags yet
          04 59 E8 0261 925 BLBS R9,50$ ; If LBS then no CDB
52 3A A9 3C 0264 926 MOVZWL CDB_W_STS(R9),R2 ; Get status flags
          072E 30 0268 927 50$: BSBW VALIDATE_P2 ; Validate the P2 buffer
          67 50 E9 026B 928 BLBC R0,FINISH_REQ ; If LBC, report error via IOSB
          026E 929 :
          026E 930 : Check trib address. If this is a trib address change for this
          026E 931 : channel (in which case unconditionally give up the old CDB even if
          026E 932 : the QIO subsequently fails), or if there is no current CDB, then
          026E 933 : attempt to bind this channel to the CDB for the new trib address.
          026E 934 :
51 0474 8F 3C 026E 935 MOVZWL #NMASC_PCCI TRI,R1 ; Get trib address param i.d.
          0796 30 0273 936 BSBW UNPACK_P2_BUF ; From P2 buffer
          05 50 E8 0276 937 BLBS R0,60$ ; If LBS, trib was specified
          59 59 E8 0279 938 BLBS R9,FINISH_REQ ; If LBS, no CDB - return R0,R1
          16 11 027C 939 BRB 80$ ; No trib addr, use current CDB
          OE 59 E8 027E 940 60$: BLBS R9,70$ ; If LBS, no CDB
3E A9 52 91 0281 941 CMPB R2,CDB_B_TRB_ADDR(R9) ; Address being changed ?
          OD 13 0285 942 BEQL 80$ ; If EQL no
          00E0 C542 B4 0287 943 CLRW UCBSW_VEC_CHAN(R5)[R2] ; Give-up previous CDB
          04DC 30 028C 944 BSBW ZAP_CDB_R9 ; Shut it down
          4A 10 028F 945 70$: BSBW NEW_TRIB ; Init/allocate CDB
          30 50 E9 0291 946 BLBC R0,FINISH_ERR ; If LBC, report error
          0294 947 80$:
          0294 948 : Tributary now exists change its characteristics and set them
          0294 949 : if trib is established.
          0294 950 :
29 57 06 E1 0294 951 BBC S^#IOSV STARTUP,R7,FINISH_SUC ; Br if not startup request
50 02C4 8F 3C 0298 952 MOVZWL #SS$ DEACTIVE,R0 ; Assume trib already active
51 3A A9 3C 029D 953 MOVZWL CDB_Q_STS(R9),R1 ; Get current status
          02A1 954 ASSUME CDB_C_IDLE EQ 0 ;
51 3F A9 88 02A1 955 BISB CDB_B_STA(R9),R1 ; OR in the state
51 30 A9 C8 02A5 956 BISL CDB_L_SETMODE(R9),R1 ; OR in pending SETMODE address
          19 12 02A9 957 BNEQ FINISH_ERR ; If NEQ then can't do startup
30 A9 53 D0 02AB 958 MOVL R3,CDB_L_SETMODE(R9) ; Save IRP address
          027C 30 02AF 959 BSBW START_TRIB ; Startup the trib
          02B2 960 :
          02B2 961 : Fall thru to QIORET

```

```

02B2 963      .SBTTL Complete QIO request routines
02B2 964
02B2 965      :+
02B2 966      : The following routines all exit the $QIO system service with status in R0.
02B2 967      : If an error is not being returned, further status will be eventually passed
02B2 968      : via the IOSB when the IRP undergoes post processing.
02B2 969
02B2 970
02B2 971      Inputs:      R3  IRP address
02B2 972      R5  UCB address
02B2 973
02B2 974      IPL may be FIPL or IPL$_ASTDEL
02B2 975
02B2 976      -
00000000'GF 17 02B2 977 QIORET: JMP      G^EXE$QIORETURN      ; Return success in R0 to user
          50 2C D0 02B8 978 ABORT_REQ1:
02B8 979      MOVL      S^#SS$_ABORT,R0      ; Setup error status
00000000'GF 17 02B8 980 ABORT_REQ:
02B8 981      JMP      G^EXE$ABORTIO      ; Exit QIO service with error
02C1 982
02C1 983
02C1 984
02C1 985      :+
02C1 986      : The following routines exit the $QIO system service with SSS_NORMAL and
02C1 987      : send the IRP back to IOPOST to return final status via the IOSB.
02C1 988
02C1 989      Inputs:      R3  IRP address
02C1 990      R5  UCB address
02C1 991
02C1 992      IPL may be either FIPL or IPL$_ASTDEL
02C1 993
02C1 994      -
          50 01 3C 02C1 995 FINISH_SUC:
02C1 996      MOVZWL   S^#SS$_NORMAL,R0      ; Set success
          51 D4 02C4 997 FINISH_ERR:
          09 E0 02C4 998      CLRL      R1      ; Clear second IOSB longword
          0A 20 A3 02C6 999      BBS      S^#IOSV_CTRL,-      ; Skip for controllers
          07 50 E9 02CB 1000      IRPSW_FUNC(R3),FINISH_REQ
          00002800 8F D0 02CB 1001      ASSUME   CDB V-RUN EQ 0
          51 00002800 8F D0 02CB 1002      BLBC      R0,FINISH_REQ      ; If LBC then circuit not up
          02D5 1003      MOVL      #XMSM_STS_ACTIVE!-      ; Indicate circuit up
          00000000'GF 17 02D5 1004      XMSM_STS_RUNNING,R1
          02D5 1005 FINISH_REQ:
          02D5 1006      JMP      G^EXE$FINISHIO      ; Complete the I/O
          02DB 1007

```

```

02DB 1009 .SBTTL NEW_TRIB - Allocate and init new CDB
02DB 1010
02DB 1011 :+
02DB 1012 : NEW_TRIB - allocate and init new CDB
02DB 1013 :
02DB 1014 :
02DB 1015 : A CDB is allocated and initialized and stored in the UCB CDB vector.
02DB 1016 : The address is also stored in IRP$L_CDB.
02DB 1017 :
02DB 1018 :
02DB 1019 : Inputs: R2 Trib address
02DB 1020 : R3 IRP address
02DB 1021 : R5 UCB address
02DB 1022 :
02DB 1023 : Outputs: R9 CDB address
02DB 1024 :
02DB 1025 : R0-R2 are clobbered. All other registers are preserved.
02DB 1026 :
02DB 1027 :-
02DB 1028 NEW_TRIB:
40 A3 52 90 02DB 1029 MOVB R2,IRP$B_INDEX(R3) ; Set the trib number
50 0641 8F 3C 02DF 1030 MOVZWL #SS$_DEV$LRALLOC,R0 ; Assume error
00E0 C542 B5 02E4 1031 TSTW UCB$W_VEC_CHAN(R5)[R2] ; Claimed by another channel?
1E 12 02E9 1032 BNEQ 40$ ; If NEQ yes, report error
59 00A0 C542 D0 02EB 1033 MOVL UCB$L_VEC_CDB(R5)[R2],R9 ; Get associated CDB
0F 12 02F1 1034 BNEQ 30$ ; If NEQ, CDB exists - claim it
15 10 02F3 1035 BSBB NEW_CDB ; Create a new CDB
11 50 E9 02F5 1036 BLBC R0,40$ ; If LBC then error
52 40 A3 9A 02F8 1037 MOVZBL IRP$B_INDEX(R3),R2 ; Restore trib address
00A0 C542 59 D0 02FC 1038 MOVL R9,UCB$L_VEC_CDB(R5)[R2] ; Store CDB address in UCB
28 A3 B0 0302 1039 30$: MOVW IRP$W_CHAN(R3),- ; Save channel index in UCB
00E0 C542 0305 1040 40$: RSB ; Done
05 0309 1041
030A 1042
030A 1043
030A 1044 NEW_CDB: ; Create new CDB
51 0060 8F 3C 030A 1045 MOVZWL #CDB_C_LENGTH,R1 ; Get size of CDB
53 DD 030F 1046 PUSHL R3 ; Save reg
00000000 GF 16 0311 1047 JSB G^EXE$ALONONPAGED ; Allocate the CDB
53 8ED0 0317 1048 POPL R3 ; Restore reg
40 50 E9 031A 1049 BLBC R0,100$ ; Br if error
031D 1050
031D 1051 ; Initialize CDB
031D 1052
031D 1053
59 52 D0 031D 1053 MOVL R2,R9 ; Copy CDB address
54 A3 52 D0 0320 1054 MOVL R2,IRP$L_CDB(R3) ; Save it in IRP
52 08 A2 9E 0324 1055 MOVAB CDB_W_SIZE(R2),R2 ; Setup ptr to init CDB
0328 1056
0328 1057 ASSUME CDB_B_TYPE EQ 2+CDB_W_SIZE
0328 1058 ASSUME CDB_B_FIPL EQ 1+CDB_B_TYPE
0328 1059 ASSUME CDB_L_FPC EQ 1+CDB_B_FIPL
0328 1060 ASSUME CDB_L_FR3 EQ 4+CDB_L_FPC
0328 1061 ASSUME CDB_L_FR4 EQ 4+CDB_L_FR3
0328 1062
82 82 51 B0 0328 1063 MOVW R1,(R2)+ ; CDB_W_SIZE
0617 8F B0 0328 1064 MOVW #<CDB_C_FIPL*8>+DYN$C_NET,(R2)+ ; CDB_B_TYPE and CDB_B_FIPL
52 0C C0 0330 1065 ADDL #3*4,R2 ; Advance passed CDB_L_FR4

```

```

0333 1066
0333 1067 ASSUME CDB_Q_XMT_IRP EQ 4+CDB_L_FR4
0333 1068 ASSUME CDB_Q_RCV_IRP EQ 8+CDB_Q_XMT_IRP
0333 1069 ASSUME CDB_Q_RCV_MSG EQ 8+CDB_Q_RCV_IRP
0333 1070
51 03 D0 0333 1071 MOVL #3,R1 ; Set number of queue heads
62 62 DE 0336 1072 20$: MOVAL (R2),(R2) ; Init forward link pointer
82 82 DE 0339 1073 MOVAL (R2)+,(R2)+ ; Init backward link pointer
F7 51 F5 033C 1074 SOBGTR R1,20$ ; Loop if more queues
033F 1075
033F 1076 ASSUME CDB_L_SETMODE EQ 8+CDB_Q_RCV_MSG
033F 1077 ASSUME CDB_L_ABSTIME EQ 4+CDB_L_SETMODE
033F 1078 ASSUME CDB_W_BUFSIZ EQ 4+CDB_L_ABSTIME
033F 1079 ASSUME CDB_W_STS EQ 2+CDB_W_BUFSIZ
033F 1080
82 42 82 7C 033F 1081 CLRQ (R2)+ ; Init CDB_L_SETMODE,ABSTIME
3C 0341 1082 MOVZWL UCB$W_DEVBUFSIZ(R5),(R2)+ ; CDB_W_BUFSIZ and CDB_W_STS
0345 1083
0345 1084 ASSUME CDB_B_RCV_CNT EQ 2+CDB_W_STS
0345 1085 ASSUME CDB_B_RCV_FQ EQ 1+CDB_B_RCV_CNT
0345 1086 ASSUME CDB_B_TRB_ADDR EQ 1+CDB_B_RCV_FQ
0345 1087 ASSUME CDB_B_STA EQ 1+CDB_B_TRB_ADDR
0345 1088 ASSUME CDB_C_IDLE EQ 0
0345 1089
62 009F C5 90 0345 1090 MOVB UCB$B_RCV_CNT(R5),(R2) ; CDB_B_RCV_CNT (default)
82 82 90 034A 1091 MOVB (R2)+,(R2)+ ; CDB_B_RCV_FQ (default)
82 40 A3 9B 034D 1092 MOVZBW IRP$B_INDEX(R3),(R2)+ ; CDB_B_TRB_ADDR, CDB_B_STA
0351 1093
0351 1094 ASSUME CDB_L_BRC EQ 1+CDB_B_STA
0351 1095 ASSUME CDB_L_BSN EQ 4+CDB_L_BRC
0351 1096 ASSUME CDB_L_DBR EQ 4+CDB_L_BSN
0351 1097 ASSUME CDB_L_DBS EQ 4+CDB_L_DBR
0351 1098
82 7C 0351 1099 CLRQ (R2)+ ; CDB_L_BRC and CDB_L_BSN
82 7C 0353 1100 CLRQ (R2)+ ; CDB_L_DBR and CDB_L_DBS
0355 1101
0355 1102 ASSUME CDB_L_UCB EQ 4+CDB_L_DBS
0355 1103 ASSUME CDB_L_CDT EQ 4+CDB_L_UCB
0355 1104
82 55 D0 0355 1105 MOVL R5,(R2)+ ; CDB_L_UCB
82 82 D4 0358 1106 CLRL (R2)+ ; CDB_L_CDT
50 01 D0 035A 1107 MOVL #1,R0 ; Indicate success
05 035D 1108 100$: RSB ; Done
035E 1109
035E 1110

```

```

035E 1112 .SBTTL SETMODE_CTRL, Perform setmode FDT operation on controller
035E 1113
035E 1114 :++
035E 1115 : SETMODE_CTRL - Perform setmode FDT operation on controller
035E 1116 :
035E 1117 :
035E 1118 : This routine performs the SETMODE FDT setup for the controller.
035E 1119 :
035E 1120 : Inputs: R3 = IRP address
035E 1121 : R4 = PCB address
035E 1122 : R5 = UCB address
035E 1123 : R7 = IRP function word
035E 1124 :
035E 1125 : Outputs: R0 = status of setmode request
035E 1126 :
035E 1127 : R3-R5 are preserved.
035E 1128 :
035E 1129 :--
035E 1130 SETMODE_CTRL: ; Perform setmode on controller
05 57 07 E1 035E 1131 BBC S^#IOSV_SHUTDOWN,R7,10$ ; Br if not shutdown request
0362 1132 :
0362 1133 : Shutdown modifier specified
0362 1134 :
03D0 30 0362 1135 BSBW CAN_DEV ; Shutdown the device
3F 11 0365 1136 BRB 50$ ; Finish the QIO with success
0367 1137 10$:
0367 1138 : Startup line modifier specified or no modifier
0367 1139 :
31 68 A5 E0 0367 1140 BBS #UCB$V_CN_INITED,- ; Br if controller up already
0369 1141 UCB$W_DEVSTS(R5),40$
036C 1142 :
036C 1143 : Validate P2
036C 1144 :
51 FD3A CF 9E 036C 1145 MOVAB LINE_P2_TABLE,R1 ; Address of verif table
59 55 D0 0371 1146 MOVL R5,R9 ; Address of current param's
52 68 A5 3C 0374 1147 MOVZWL UCB$W_DEVSTS(R5),R2 ; Status flags
061E 30 0378 1148 BSBW VALIDATE_P2 ; Validate P2 buffer
2B 50 E9 037B 1149 BLBC R0,70$ ; If LBC, return R0,R1 in IOSB
037E 1150 :
037E 1151 : Setup Maximum receive buffers
037E 1152 :
51 0451 8F 3C 037E 1153 MOVZWL #NMASC_PCLI_BFN,R1 ; Set to find MAX RCV
0686 30 0383 1154 BSBW UNPACK_P2_BOF ; In P2 buffer
05 50 E9 0386 1155 BLBC R0,30$ ; Br if not found
009F C5 52 90 0389 1156 MOVB R2,UCB$B_RCV_CNT(R5) ; Initialize number of RCV
038E 1157 30$:
038E 1158 : Setup Blocksize
038E 1159 :
51 0AF1 8F 3C 038E 1160 MOVZWL #NMASC_PCLI_BUS,R1 ; Get buffer size
0676 30 0393 1161 BSBW UNPACK_P2_BOF ; From P2 buffer
04 50 E9 0396 1162 BLBC R0,40$ ; Br if not found
42 A5 52 B0 0399 1163 MOVW R2,UCB$W_DEVBUFSIZ(R5) ; and in UCB
039D 1164 40$:
039D 1165 : Device initialized - then do a LISTEN if IOSV_STARTUP
039D 1166 :
05 57 06 E1 039D 1167 BBC S^#IOSV_STARTUP,R7,50$ ; Finish up if not starting
OC 10 03A1 1168 BSBB LISTEN ; Do a LISTEN
    
```

```

06 50   E9 03A3 1169          BLBC   R0,100$           ; If LBC then failed
F:18   31 03A6 1170 50$:    BRW   FINISH_SUC        ; Finish - SS$ NORMAL fo: IOSB
FF29   31 03A9 1171 70$:    BRW   FINISH_REQ        ; Finish - R0,R1 for IOSB
FF0C   31 03AC 1172 100$:   BRW   ABORT_REQ         ; Abort the I/O request
      03AF 1173
      03AF 1174 LISTEN:   ;
      03AF 1175           ; Do all the wonderful SCS magic needed to start up. The buffer
      03AF 1176           ; created on the stack is pointed to by R7 in case the CONFIG_SYS
      03AF 1177           ; macro is updated someday to modify SP as it pushes arguments.
      03AF 1178           ;
      03AF 1179           ; NOTE: The following code assumes that we have only 1 CI port on
      03AF 1180           ; the current system!!!!
      03AF 1181           ;
      03AF 1182           ;
      00000070 03AF 1183      SBO_LNG = SBO$C_LENGTH + 32           ; SBO length plus random amount
      03AF 1184           ; of padding merely for merely
      03AF 1185           ; hysterical purposes.
SE 00000070 59 53  D0 03AF 1186      MOVL   R3,R9           ; Save R3
      8F  C2 03B2 1187      SUBL   #SBO_LNG,SP      ; Create buffer on stack
      57  5E  D0 03B9 1188      MOVL   SP,R7           ; Preserve value of buffer
      03BC 1189      CONFIG_SYS G^SCS$GB_SYSTEMID,(R7)        ; Get our system block
      08 50   E9 03CC 1191      BLBC   R0,200$         ; If LBC, not ready yet
      3C A7   90 03CF 1192      MOVB  SBO$B_RSTATION1(R7),- ; Get our port number
      009E C5           03D2 1193      UCB$B_CN_PORT(R5)
      50 0084 07 18 03D5 1194      BGEQ  210$           ; If LSS then not ready yet
      36 3C 03D7 1195 200$:  MOVZWL #SS$_DEVOFFLINE,R0 ; Device offline error (no PA)
      56 14 A1  D0 03DE 1196      BRB   220$           ; Exit
      56 2C A6  D0 03E2 1198      MOVL  SB$S_PBCONNX(R1),R6 ; Get path block
      0084 C5 56  D0 03E6 1199      MOVL  PB$S_PDT(R6),R6    ; Pick up PDT
      03E8 1200      MOVL  R6,UCB$S_PDT(R5) ; Save in UCB
      03E8 1201      LISTEN - ; Setup a LISTEN
      03E8 1202      MSGADR = W^LIS_FORK,-
      03E8 1203      ERRADR = W^LIS_ERR,-
      03E8 1204      LPRNAM = PROC_NAM,-
      03E8 1205      PRINFO = PROC_NAM
      0090 C5 0D 50   E9 0404 1205      BLBC   R0, 220$         ; If LBC then error
      5C A3 55  D0 0407 1206      MOVL  R3,UCB$S_LIS_CDT(R5) ; Save listen CDT
      68 A5 01  A8 040C 1207      MOVL  R5,CDT$S_AUXSTRUC(R3) ; Set addr of UCB into CDT
      SE 00000070 8F  C0 0410 1208      BISW  #UCB$M_CN_INITED,UCB$W_DEVSTS(R5) ; Indicate device initied
      53 59  D0 0414 1209           ;
      0414 1210 220$:  ADDL   #SBO_LNG,SP ; Restore stack
      041B 1211      MOVL  R9,R3 ; Restore IRP addr
      05 041E 1212      RSB
      041F 1213
      041F 1214 LIS_ERR: ; Error on LISTEN CDT
      041F 1215      DISCONNECT ; Put it back to listen
      05 0425 1216      RSB ; Leave
      0426 1217

```

```

0426 1219      .SBTTL SENSEMODE_FDT, Sense Mode I/O operation FDT routine
0426 1220
0426 1221      :++
0426 1222      : SENSEMODE_FDT - Sense Mode FDT routine
0426 1223      :
0426 1224      : This routine returns information to the caller about the configuration
0426 1225      : and status of the CI device. Depending on the function modifier,
0426 1226      : either the device characteristics or error counters contents are returned.
0426 1227      :
0426 1228      : The QIO parameters for SENSEMODE are:
0426 1229      :
0426 1230      :     P2 = optional address of buffer descriptor for extended characteristics
0426 1231      :
0426 1232      :
0426 1233      : Inputs:      R3 = IRP address
0426 1234      :             R4 = PCB address
0426 1235      :             R5 = UCB address
0426 1236      :             R6 = CCB address
0426 1237      :             R7 = Function code
0426 1238      :             AP = Address of first function-dependent QIO parameter
0426 1239      :
0426 1240      : Outputs:     R0 = status return of sensemode request
0426 1241      :
0426 1242      :             R3-R5 are preserved.
0426 1243      :
0426 1244      :--
0426 1245      SENSE_TABLE:
03A2 0426 1246      .WORD SENSE_TABLE - TRIB_PRM_TABLE      : Setup list of offset to
034A 0428 1247      .WORD SENSE_TABLE - TRIB_CNT_TABLE      : parameter tables with using
037C 042A 1248      .WORD SENSE_TABLE - LINE_PRM_TABLE      : the following 2 bit index:
0338 042C 1249      .WORD SENSE_TABLE - LINE_CNT_TABLE      :
042E 1250      :             bit 0 set => counters
042E 1251      :             bit 1 set => non-trib
042E 1252      SENSEMODE_FDT:      : Sensemode FDT I/O processing
00000080 042E 1253      SENSE_C_BUF = 128
042E 1254      ASSUME TRIB_PRM_NUM*6 LE SENSE_C_BUF      : Make sure buffer can hold all
042E 1255      ASSUME LINE_PRM_NUM*6 LE SENSE_C_BUF      : info for all cases
042E 1256      ASSUME TRIB_CNT_NUM*6 LE SENSE_C_BUF
042E 1257      ASSUME LINE_CNT_NUM*6 LE SENSE_C_BUF
042E 1258      :
042E 1259      : Check user buffer. Get system buffer. Setup IRP
042E 1260      :
042E 1261      :
042E 1262      MOVW #SENSE_C_BUF,IRP$I_OST1+2(R3)      : Setup buff size needed
3A A3 0080 8F B0 042E 1263      BSBW GET_CHAR_RBUF      : Setup "read" buff for IOPOST
0434 1264      : - no return on error
0437 1265      MOVZWL IRP$W_FUNC(R3),R7      : Get full function code.
04  A2  57  20 A3  3C 0437 1265      MOVL IRP$I_OST2(R3),4(R2)      : Store user buffer virt addr
0440 1266      : in standard place in buffer
0440 1267      MOVL (R2),R2      : Get pointer to data area
0443 1268      :
0443 1269      :
0443 1270      : Locate parameter/counter table
0443 1271      :
0443 1272      MOVL #3,R6      : Init SENSE_TABLE index
0446 1273      MOVL #COUNT_C_ENTRY-2,R8      : Bias COUNTER table entry size
09  57  58  02 D0 0446 1273      BBS #IOSV_RD_COUNT,R7,10$      : If BS, "read counter" request
0449 1274      DECB R6      : Erase "read counter" bit
044D 1275

```



```

00 58 0A D0 044F 1276      MOVL   #PARAM_C_ENTRY-2,R8      ; Bias PARAM table entry size
    57 08 E5 0452 1277      BBCC   #IOSV_RD_COUNT,R7,10$    ; Clear out garbage modifier
    59 55 D0 0456 1278 10$:  MOVL   R5,R9                    ; If IOSV_CTRL, use UCB source
09 57 09 E0 0459 1279      BBS    #IOSV_CTRL,R7,20$       ; If BS, not for a tributary
    56 02 8A 045D 1280      BICB   #2,R6                    ; Erase 'non-tributary' flag
        0500 30 0460 1281      BSBW   XLATE                    ; Locate CDB, use CDB source
        6C 50 E9 0463 1282      BLBC   R0,100$                 ; If LBC then CDB not found
50  BC AF46 32 0466 1283 20$:  CVTWL  SENSE_TABLE[R6],R0      ; Get offset to parameter table
    56  BB AF 9E 046B 1284      MOVAB  SENSE_TABLE,R6          ; Get address of base
    56  50  C2 046F 1285      SUBL   R0,R6                    ; Calculate table address
        0472 1286 30$:      ;
        0472 1287      ; Fill buffer with requested information
        0472 1288      ;
    51  86 B0 0472 1289      MOVW   (R6)+,R1                 ; Get parameter i.d.
        2E 13 0475 1290      BEQL   60$                      ; If EQL, at end of table
    05 57 08 E0 0477 1291      BBS    #IOSV_RD_COUNT,R7,40$    ; If BS then counter i.d.
54 51 F000 8F AA 047B 1292      BICW   #^C<PRM # TYPE>,R1      ; Else param i.d., clear junk
50 66 0A 00 EF 0480 1293 40$:  EXTZV  #OFF_V_VALUE,#OFF_S_VALUE,(R6),R4 ; Get source offset
    66 06 0A EF 0485 1294      EXTZV  #OFF_V_WIDTH,#OFF_S_WIDTH,(R6),R0 ; Get source width
        14 13 048A 1295      BEQL   50$                      ; If EQL, ignore this param
        54 59 C0 048C 1296      ADDL   R9,R4                    ; Calculate source address
        82 51 B0 048F 1297      MOVW   R1,(R2)+                 ; Enter parameter i.d.
82 64 50 00 EF 0492 1298      EXTZV  #0,R0,(R4),(R2)+         ; Enter parameter value
    05 57 0A E1 0497 1299      BBC    #IOSV_CLR_COUNT,R7,50$   ; If BC, don't clear source
64 50 00 00 F0 049B 1300      INSV   #0,#0,R0,(R4)           ; Clear counter
    56 58 C0 04A0 1301 50$:  ADDL   R8,R6                    ; Advance to next entry
        CD 11 04A3 1302      BRB    30$                      ; Loop
        04A5 1303 60$:      ;
        04A5 1304      ; Setup status and transfer size
        04A5 1305      ;
    52 2C B3 C2 04A5 1306      SUBL   @IRPSL_SVAPTE(R3),R2     ; Calculate bytes moved
00000080 8F 52 D1 04A9 1307      CMPL   R2,#SENSE_C_BUF         ; Was our buffer large enough?
        23 1A 0480 1308      BGTRU  200$                     ; If GTRU no
        38 A3 01 B0 0482 1309      MOVW   #SS$ NORMAL,IRPSL_IOST1(R3) ; Assume success
    52 32 A3 B1 0486 1310      CMPW   IRPSW_BCNT(R3),R2       ; User buffer big enough?
        0A 1E 048A 1311      BGEQU  80$                      ; If GEQU then yes
38 A3 0601 8F B0 048C 1312      MOVW   #SS$ BUFFEROVF,IRPSL_IOST1(R3) ; Show warning
    52 32 A3 B0 04C2 1313      MOVW   IRPSW_BCNT(R3),R2       ; Shrink xfer size
    3A A3 52 B0 04C6 1314 80$:  MOVW   R2,IRPSL_IOST1+2(R3)     ; Move xfer size to IOSB image
    32 A3 52 B0 04CA 1315      MOVW   R2,IRPSW_BCNT(R3)       ; Setup xfer size for IOPOST
    50 38 A3 D0 04CE 1316 100$: MOVL   IRPSL_IOST1(R3),R0      ; Set length/status
        FDEF 31 04D2 1317      BRW    FINISH_ERR              ; Leave setting R0 in IOSB
        04D5 1318      ;
    04D5 1319 200$:  BUG_CHECK INCONSTATE,FATAL ; We've corrupted pool

```

```

04D9 1321      .SBTTL GET_CHAR_RBUF, Get P2 characteristics buffer for read
04D9 1322      .SBTTL GET_CHAR_WBUF, Get P2 characteristics buffer for write
04D9 1323
04D9 1324      :++
04D9 1325      : GET_CHAR_RBUF - Get P2 characteristics buffer for read
04D9 1326      : GET_CHAR_WBUF - Get P2 characteristics buffer for write
04D9 1327      :
04D9 1328      : This routine saves the address of P2 buffer for later use by the driver.
04D9 1329      : The P2 buffer address is saved in IRP$I_OST2 of the IRP, and the size
04D9 1330      : in IRP$I_BCNT.
04D9 1331      :
04D9 1332      : Inputs:      R3 = IRP address
04D9 1333      :           R4 = PCB address
04D9 1334      :           R5 = UCB address
04D9 1335      :
04D9 1336      : Outputs:     R0 = Garbage
04D9 1337      :           R1 = User buffer size
04D9 1338      :           R3-R5 are preserved.
04D9 1339      :
04D9 1340      :--
04D9 1341      GET_CHAR_RBUF:      : Get P2 char buffer for "read"
04D9 1342      BISH      #IRP$I_FUNC,IRP$I_STS(R3)      : Mark IRP for "read"
04DD 1343      GET_CHAR_WBUF:      : Get P2 char buffer for "write"
04DD 1344      CLRQ      R0      : Setup null user buffer
04DF 1345      MOVL     P2(AP),R2      : Get address of P2 desc
04E3 1346      BEQL     10$      : If EQL, no P2 was specified
04E5 1347      EXTZV   #0,#2,IRP$I_RMOD(R3),R0      : Get access mode
04EB 1348      IFNORD   #8,(R2),50$,MODE=R0      : Br if no read access
04F1 1349      MOVZWL  (R2),R1      : Get buffer length in bytes
04F4 1350      MOVL     4(R2),R0      : Get buffer address
04F8 1351      10$:      MOVL     R0,IRP$I_IOST2(R3)      : Save it for later
04FC 1352      TSTW     R1      : Null user buffer?
04FE 1353      BEQL     30$      : If EQL yes, don't probe
0500 1354      PUSHAB  B^30$      : Setup return address
0503 1355      BBC      #IRP$I_FUNC,IRP$I_STS(R3),20$      : If BC then "write"
0508 1356      JMP      G^EXE$READCHK      : Check user buffer, setup IRP
050E 1357      20$:      JMP      G^EXE$WRITECHK      : Check user buffer, setup IRP
0514 1358      :           - no return on error
0514 1359      30$:      BBC      #IRP$I_FUNC,IRP$I_STS(R3),40$      : If BC then "write"
0519 1360      MOVZWL  IRP$I_IOST1+2(R3),R1      : Get required buffer size
051D 1361      40$:      BSBW     GET_BOF      : Get buffer
0520 1362      SETIPL  UCBSB FIPL(R5)      : Raise IPL
0524 1363      BLBS     R0,60$      : Okay if LBS
0527 1364      50$:      MOVZWL  #SS$ ACCVIO,R0      : Set error status
052A 1365      BRW     ABORT_REQ      : Abort the I/O request
052D 1366      60$:      RSB
052E 1367
  
```

```

052E 1369      .SBTTL START_TRIB, Start tributary routine
052E 1370
052E 1371      :++
052E 1372      : START_TRIB - Start tributary routine
052E 1373      :
052E 1374      :
052E 1375      : This routine is called when a tributary is to be established and started.
052E 1376      : The control parameters are initialized also.
052E 1377      :
052E 1378      : Inputs:      R3 = IRP address
052E 1379      :           R5 = UCB address
052E 1380      :           R9 = CDB address
052E 1381      :
052E 1382      :           IPL = FIPL.
052E 1383      :
052E 1384      : Outputs:    R5 is preserved.
052E 1385      :
052E 1386      :
052E 1387      :--
052E 1388      START_TRIB:                                ; Start tributary
052E 1389      :
052E 1390      : Setup number of receive buffers
052E 1391      :
052E 1392      :
167 0479 8F 3C 052E 1392      MOVZWL #NMASC_PCCI MRB,R1          ; Get param i.d.
052E 1393      BSBW UNPACK_P2_BUF          ; Get param value
052E 1394      BLBS RO,20$              ; If LBS then param was there
168 009F C5 9A 0539 1395      MOVZBL UCBSB_RCV_CNT(R5),R2      ; Else, get default
052E 1396      CMPB R2,#RBFMIN          ; Are there enough buffers to
169 09 52 91 053E 1396      20$:      ; reduce datagram loss?
052E 1397      BGTRU 30$                ; If not, do user a favor
052E 1398      MOVL #RBFMIN,R2          ; Too many?
170 52 09 D0 0543 1398      30$:      ; If GTRU then no
052E 1399      CMPL #RBFMAX,R2          ; Use safer minimum
171 52 1F D1 0546 1399      ;
052E 1400      BGTRU 40$                ; Setup receive pool accounting
172 52 03 1A 0549 1400      40$:      ; List starts out full
052E 1401      MOVL #RBFMAX,R2
173 3C A9 52 90 054E 1402      MOVVB R2,CDB_B_RCV_CNT(R9)
174 3D A9 52 90 0552 1403      MOVVB R2,CDB_B_RCV_FQ(R9)
052E 1404      :
052E 1405      : Init CDB state.
052E 1406      :
052E 1407      :
175 48 A3 30414150 8F D0 0556 1407      MOVL #*A/PAA0/,IRPSB_INDEX+8(R3) ; Set to connect over local
052E 1408      : port PAA0
052E 1409      : Assume "listen" state
176 3F A9 03 90 055E 1409      MOVVB #CDB_C_LSTN,CDB_B_STA(R9) ; Compare our address to
052E 1410      CMPB UCBSB_CN_PORT(R5),- ; remote's address
177 009E C5 91 0562 1410      :
052E 1411      : IRPSB_INDEX(R3)
178 40 A3 22 1F 0568 1412      BLSSU 100$ ; If LSSU, stay in "listen"
052E 1413      BGTRU 50$                ; If GTRU, initiate connect
179 04 1A 056A 1413      :
052E 1414      CLRL R2                ; Else we're talking to
180 52 D4 056C 1414      : ourselves -- zero rcv buffers
052E 1415      :
181 04 11 056E 1416      BRB 60$ ; CONNECT from "LSTN" state
052E 1417      MOVVB #CDB_C_CONN,CDB_B_STA(R9) ; Else, go to "connect" state
182 3F A9 02 90 0570 1417      50$:      ; Indicate waiting return from
183 14 3A A9 01 E2 0574 1418      60$:      ; CONNECT
052E 1419      :
052E 1420      PUSHL R5                ; Save UCB address
052E 1421      MOVL R9,R5                ; Use CDB for CONNECT context
052E 1422      BSBB CONN                    ; Post connect request to SCS
052E 1423      POPL R5                    ; Restore UCB address
184 04 3A A9 01 8E D0 0580 1423      ;
052E 1424      BBC #CDB_V_CONN,CDB_W_STS(R9),100$ ; If BC, completed synchronously
185 54 A9 53 D0 0588 1425      MOVL R3,CDB_L_CDI(R9) ; Set pointer to open CDI

```

```

05 058C 1426 100$: RSB ; Done
058D 1427
058D 1428 200$: BUG_CHECK INCONSTATE,FATAL ; Bug if already set
0591 1429
50 3C A9 9A 0591 1430 CONN: MOVZBL CDB_B_RCV_CNT(R9),R0 ; Pick up rcv buffer count
0595 1431 CONNECT - ; Request a CONNECT
0595 1432 MSGADR = W^MSG FORK,- ; Message address
0595 1433 DGADR = W^DG FORK,- ; Psuedo interrupt routine
0595 1434 ERRADR = W^CONN_ERR,- ; Connect errors
0595 1435 RSYSID = 0,- ; No remote system specified
0595 1436 RSTADR = IRPSB INDEX(R3),- ; Virtual circ to connect over
0595 1437 RPRNAM = PROC_NAM,- ; To whom we will speak
0595 1438 LPRNAM = PROC_NAM,- ; Our name
0595 1439 INITCR = #1,- ; Allow for messages
0595 1440 INITDG = R2,- ; Number of receive buffers
0595 1441 CONDAT = CONN_DATA,- ; Connect data
0595 1442 AUXSTR = (R5) ; Auxiliary structure
05CB 1443
05CB 1444 : Control returns to caller's caller - the JMP G^EXESQIORETURN.
05CB 1445 : When the connection completes, the following is called as a fork
05CB 1446 : process NOT necessarily in the context of process.
05CB 1447
05CB 1448 R0 = Status code
05CB 1449 R1 = Reject reason if status = reject
05CB 1450 R2 --> ACCEPT_REQ msg if status = success
05CB 1451 R3 --> Connection CDT
05CB 1452 R4 --> PDT
05CB 1453 R5 --> CDB
05CB 1454
0E 50 E9 05CB 1455 CLRBIT #CDB_V_CONN,CDB_W_STS(R5) ; No longer awaiting CONN return
00F9 30 05D0 1456 BLBC R0,20$ ; If LBS then error
03 12 05D3 1457 BSBW CHECK_REMOTE ; Check remote's connect info
00B8 31 05D6 1458 BNEQ 10$ ; If NEQ, can't talk to remote
05DB 1459 BRW CONN_FIN ; Else okay, complete setup
54 A5 D4 05DB 1460 10$: DISCONNECT ; Break the connection
05E1 1461 20$: CLRL CDB_L_CDT(R5) ; Forget about CDT, if any
05E4 1462
05E4 1463 CONN_ABO:
05E4 1464
05E4 1465 : CONNECT or ACCEPT failed.
05E4 1466
05E4 1467 : If we were to return an error immediately every time, the higher
05E4 1468 : level user (NETACP) would consume too much time trying to restart
05E4 1469 : the circuit. This is because the CI, unlike other devices, will
05E4 1470 : return immediately if the partner is not ready on a CONNECT
05E4 1471 : attempt. For all other devices, the connect remains pending
05E4 1472 : indefinitely.
05E4 1473
05E4 1474 : Therefore, in order to save CPU cycles, simply return and allow the
05E4 1475 : IOSM_STARTUP $QIO to hang indefinitely. This forces NETACP to
05E4 1476 : initiate the subsequent cleanup via a $CANCEL, $DASSGN, or
05E4 1477 : IOSM_SHUTDOWN. This should be fixed someday to have CNDRIVER retry
05E4 1478 : every 5 seconds or so without reporting an error.
05E4 1479
05E4 1480
05E4 1481
05E4 1482 :

```

\*\*\* NOTE: This logic here has been retained in case it needs to be reactivated someday. However, it has been

```

05E4 1483      :
05E4 1484      :
05E4 1485      :
05E4 1486      :
05E4 1487      :
05E4 1488      :
05E4 1489      :
05E4 1490      :
    3A A5 B5 05E4 1491      TSTW CDB_W_STS(R5)      ; All quiet yet ?
    18 12 05E7 1492      BNEQ 100$      ; If NEQ, just wait
09 50 50 A5 D0 05E9 1493      MOVL CDB_L_UCB(R5),R0      ; Get UCB address
64 A0 05 E0 05ED 1494      BBS #UCB$V_POWER,UCB$W_STS(R0),50$ ; If BS, powerfial recovery
    5B A5 96 05F2 1495      INCB CDB_B_RSTCNT(R5)      ; Another restart attempt
5B A5 03 93 05F5 1496      BITB #3,CDB_B_RSTCNT(R5) ; Is this the 4th phase ?
    01 05F9 1497      BEQL 100$      ; If EQL yes, wait.
    01 05F9 1498      NOP
    54 55 D0 05FB 1500 50$: MOVL R5,R4      ; Copy CDB address
    016D 30 05FE 1501      BSBW ZAP_CDB      ; Report the error immediately
    05 0601 1502 100$: RSB      ; Wait the gio until contacted
    0602 1503      ; by user via $CANCEL, etc
    0602 1504

```

found that not returning an error immediately can cause some confusion since it can delay a circuit initialization for 3 minutes or so in some cases. In addition, the time spent by NETACP to continually reinitialize the circuit has been found to be small enough that it presents no real problem.

```

0602 1506          .SBTTL LIS_FORK, Listen action routine
0602 1507
0602 1508      :++
0602 1509      : LIS_FORK - Listen action routine
0602 1510      :
0602 1511      :
0602 1512      : This routine is entered as a fork process activated by the PADRIVER
0602 1513      : when some other process has sent a CONNECT to us. We can then decide
0602 1514      : to ACCEPT or REJECT the connection.
0602 1515      :
0602 1516      : Inputs:      R2 -->  CONNECT_REQ message
0602 1517      :                R3 -->  Listening CDT
0602 1518      :                R4 -->  PDT
0602 1519      :
0602 1520      :                IPL = FIPL
0602 1521      :
0602 1522      :--
0602 1523      LIS_FORK:
55 5C A3 D0 0602 1524      MOVL      CDT$AUXSTRUC(R3),R5      : Pick up UCB from Listen CDT
51 20 A3 9A 0606 1525      MOVZBL   CDT$B_RSTATION(R3),R1      : Get other guy's port
55 00A0 C541 D0 060A 1526      MOVL      UCBS$L_VEC_CDB(R5)[R1],R5      : Pick up the CDB
03 3F A5 13 0610 1527      BEQL      REJECT      : We don't have one, reject
03 3F A5 91 0612 1528      CMPB     CDB_B_STA(R5),#CDB_C_LSTN      : Are we listening on this trib
03 3F A5 12 0616 1529      BNEQ     REJECT      : If NEQ no, reject connection
03 3F A5 00B4 30 0618 1530      BSBW     CHECK_REMOTE      : Process connect data
03 3F A5 09 13 061B 1531      BEQL      ACCEPT      : If NEQ then from DECnet SYSAP
061D 1532      REJECT:      :
061D 1533      : REJECT the connection.
061D 1534      :
061D 1535      CLRL      R5      : Forget about CDB
50 55 D4 D0 061F 1536      MOVL      #SS$_NORMAL,R0      : Reject reason
50 01 D0 0622 1537      REJECT      : Yes, reject him - return to
0625 1538      : caller's caller.
0625 1539      RSB      : Return to SCS (nop)
0626 1540      :
0626 1541      ACCEPT:      :
0626 1542      : ACCEPT the connection.
0626 1543      :
0626 1544      MOVL      CDB_B_RCV_CNT(R5),R0      : Pick up rec buf count
50 3C A5 9A 062A 1545      MOVZBL   #CDB_C_ACPT,CDB_B_STA(R5)      : Change state to "accept"
3F A5 04 90 062E 1546      SETBIT   #CDB_V_ACPT,CDB_W_STS(R5)      : Indicate ACCEPT pending
0633 1547      ACCEPT      : ACCEPT the connection
0633 1548      MSGADR   = W^MSG FORK,-      : Message address
0633 1549      DGADR    = W^DG FORK,-      : Psuedo interrupt rtn
0633 1550      ERRADR   = W^CONN_ERR,-      : Error address
0633 1551      INITCR   = #1,-      : Allow for messages
0633 1552      INITDG   = R0,-      : Receive buffers
0633 1553      CONDAT   = CONN_DATA,-      : Accept data
0633 1554      AUXSTR   = (R5)      : Auxiliary structure (CDB)
065C 1555      :
065C 1556      : Control returns to caller's caller if this request does not complete
065C 1557      : synchronously. In that case, when the ACCEPT completes, the
065C 1558      : following is called as a fork process NOT necessarily in the context
065C 1559      : of user's process.
065C 1560      :
065C 1561      : R0 = Status
065C 1562      : R1 = ?

```

```

065C 1563      :      R2 --> ACCEPT_RSP message
065C 1564      :      R3 --> CDT
065C 1565      :      R4 --> PDT
065C 1566      :      R5 --> CDB
065C 1567      :
065C 1568      CLRBIT #CDB_V_ACPT,CDB_W_STS(R5)      : ACCEPT no longer pending
50 13 50 E9 0661 1569 BLBC R0,50$                : If LBC then failed
50 50 A5 D0 0664 1570 MOVL CDB_L_UCB(R5),R0        : Get UCB
20 A3 91 0668 1571 CMPB CDT$B_RSTATION(R3),-      : Are we talking to ourselves?
009E C0 23 12 066B 1572 UCB$B_CN_PORT(R0)
0094 C0 53 D0 066E 1573 BNEQ CONN_FIN              : If NEQ no, complete setup
50 53 52 D0 0670 1574 MOVL R3,UCB$L_TWIN_CDT(R0)   : Else setup TWIN CDT
20 11 0675 1575 BRB CONN_FINT                       : Finish processing without
50 53 52 D0 0677 1576 50$: MOVL R2,R3              : storing CDT in CDB
50 206C 8F 3C 067A 1578 MOVZWL #SS$_REMRSRC,R0      : Copy CDT to right register
067F 1579 SETBIT #CDB_V_REJECT,CDB_W_STS(R5)      : REJECT reason
0684 1580 REJECT                                     : Set REJECT in progress
0687 1581 CLRBIT #CDB_V_REJECT,CDB_W_STS(R5)      : Must REJECT on ACCEPT failure
068C 1582                                     : Return is to caller's caller
068C 1583                                     : - return here after a delay
FF55 31 068C 1584 BRW CONN_ABO                       : with R5 pointing to CDB
068F 1585                                     : Go to common code.
068F 1586 100$: BUG_CHECK INCONSTATE,FATAL

```

```

0693 1588
0693 1589 :
0693 1590 : CONNECT (or ACCEPT) succeeded
0693 1591 :
0693 1592 : If no status bits are set then enter the 'run' state and complete the
0693 1593 : pending IOSM_STARTUP request. If any status bits are set -- which can
0693 1594 : happen if we are talking to ourselves since we do both an ACCEPT and a
0693 1595 : CONNECT in that case -- then wait.
0693 1596 :
0693 1597 CONN_FIN:
54 A5 53 D0 0693 1598 MOVL R3,CDB_L_CDT(R5) ; Set ptr to CDT
0697 1599 CONN_FIN1:
54 55 D0 0697 1600 MOVL R5,R4 ; Copy CDT address
5B A4 94 069A 1601 CLR B CDB_B_RSTCNT(R4) ; Init failed restart counter
34 A4 D4 069D 1602 CLRL CDB_L_ABSTIME(R4) ; Don't inhibit DISCONNECT
55 50 A4 D0 06A0 1603 MOVL CDB_L_UCB(R4),R5 ; Restore UCB pointer
64 A5 20 AA 06A4 1604 BICW #UCBSM_POWER,UCBSW_STS(R5) ; Any powerfail recovery is done
3A A4 B5 06A8 1605 TSTW CDB_W_STS(R4) ; All quiet?
1D 12 06AB 1606 BNEQ 40$ ; If NEQ no, wait
53 30 A4 D0 06AD 1607 MOVL CDB_L_SETMODE(R4),R3 ; Get SETMODE IRP
18 13 06B1 1608 BEQL 50$ ; If EQL then none
30 A4 D4 06B3 1609 CLRL CDB_L_SETMODE(R4) ; Detach IRP from CDB
10 20 A3 06 E1 06B6 1610 BBC #IOSM_STARTUP,IRPSW_FUNC(R3),50$ ; If BC then wrong IRP
3F A4 01 90 06BB 1611 MOV B #CDB_C_OPEN,CDB_B_STA(R4) ; Update current state
06BF 1612 SETBIT #CDB_V_RUN,CDB_W_STS(R4) ; Allow data message traffic
50 01 3C 06C4 1613 MOVZWL #SS$NORMAL,R0 ; Setup status
021A 30 06C7 1614 BSBW SUC_TRB_IOPOST ; Post IRP with 'success'
05 06CA 1615 40$: RSB
06CB 1616
06CB 1617 50$: BUG_CHECK INCONSTATE,FATAL
06CF 1618
06CF 1619 CHECK_REMOTE: ; Check remote connect data
06CF 1620 :
06CF 1621 :
06CF 1622 : 0-15(R2) Contain our process name (who remote is connecting to)
06CF 1623 : 16-31(R2) Contain remote's process name
06CF 1624 : 32-47(R2) Contain connect data
06CF 1625 :
1D BB 06CF 1626 PUSHR #*M<R0,R2,R3,R4> ; Save some registers
06D1 1627 :
54 52 D0 06D1 1628 MOVL R2,R4 ; Make stable msg pointer
5B A5 00 B0 06D4 1629 MOVW #OLD_C_PROT,CDB_W_REMPROT(R5) ; Assume remote is old protocol
20 A4 FA13 CF 06 29 06D8 1630 CMPC3 #PROC_C_NAM,PROC_NAM,32(R4) ; Check the connect data
05 13 06DF 1631 BEQL 10$ ; If EQL then old style
5B A5 20 A4 B0 06E1 1632 MOVW 32(R4),CDB_W_REMPROT(R5) ; Pickup version + system id's
10 A4 FA05 CF 06 29 06E6 1633 10$: CMPC3 #PROC_C_NAM,PROC_NAM,16(R4) ; Check the connect proc nam
06ED 1634 :
1D BA 06ED 1635 POPR #*M<R0,R2,R3,R4> ; Restore regs (but save CC's)
05 06EF 1636 RSB ; Return condition codes
06F0 1637 :
06F0 1638 :
06F0 1639 :
06F0 1640 : Error after connection established - VC disconnect most likely.
06F0 1641 :
06F0 1642 : If the CDT is the UCBSL_TWIN_CDT then simply do a DISCONNECT. This CDT is
06F0 1643 : used for receives on connects to ourselves. SCS will call us again for the
06F0 1644 : other half of that connection with the local CDB's CDT -- at that time, as

```



```

06F0 1645 ; in all other cases, we will run-down the CDB.
06F0 1646 :
06F0 1647 :   Inputs:
06F0 1648 :     R0 = Status
06F0 1649 :     R3 --> CDT
06F0 1650 :     R4 --> PDT
06F0 1651 :
06F0 1652 CONN_ERR:
54   5C   A3   D0   06F0 1653   MOVL   CDT$L_AUXSTRUC(R3),R4   ; Pick up associated CDB
55   50   A4   D0   06F4 1654   MOVL   CDB_L_UCB(R4),R5       ; Pick up UCB address
0094 C5    53   D1   06F8 1655   CMPL   R3,UCB$L_TWIN_CDT(R5)  ; Is this the "local receive" CDT ?
      6F    12   06FD 1656   BNEQ   ZAP CDB               ; If NEQ no, ZAP the CDB
      0094 C5   D4   06FF 1657   CLRL   UCB$L_TWIN_CDT(R5)    ; Else, detach it from the UCB
54   10   A3   D0   0703 1658   MOVL   CDT$L_PDT(R3),R4     ; Recover the PDT
      0707 1659   DISCONNECT ; Tell SCS to cleanup.
05   070D 1660   RSB ; Done
      070E 1661

```

```

070E 1663      .SBTTL CANCEL, Cancel I/O routine
070E 1664
070E 1665      :++
070E 1666      : CANCEL, Cancels an I/O operation in progress
070E 1667      :
070E 1668      :
070E 1669      : This routine cancels all I/O on the tributary.
070E 1670      :
070E 1671      : Inputs:      R2 = channel number
070E 1672      :                   R3 = current IRP address
070E 1673      :                   R4 = PCB address
070E 1674      :                   R5 = UCB address
070E 1675      :                   R8 = Cancel reason code: 0 => $CANCEL; 1 => $DASSGN
070E 1676      :
070E 1677      :                   IPL = FIPL
070E 1678      :
070E 1679      : Outputs:    R0-R3 are destroyed.
070E 1680      :
070E 1681      :
070E 1682      :--
070E 1683      CANCEL:
0210 8F  BB 070E 1684      PUSHR  #^M<R4,R9>      : Cancel an I/O operation
0712 1685      : Save registers
50 52  D0 0712 1686      MOVL  R2,R0      : Copy channel number
025A 30 0715 1687      BSBW  XLATE CHAN : Translate channel
10 50  E9 0718 1688      BLBC  R0,20$    : Br if none
01 58  D1 071B 1689      CMPL  R8,#1    : $DASSGN ?
09 12 071E 1690      BNEQ  10$      : If NEQ then no
50 3E A9 9A 0720 1691      MOVZBL CDB_B_TRB_ADDR(R9),R0 : Pick up trib address
00E0 C540 B4 0724 1692      CLRW  UCBSW_VEC_CHAN(R5)[R0] : Zero channel entry
40 10 0729 1693 10$:  BSBB  ZAP_CDB_R9 : Clear all CDB I/O
072B 1694      :
0210 8F  BA 072B 1695 20$:  POPR  #^M<R4,R9> : Restore registers
5C A5  B5 072F 1696      TSTW  UCBSW_REFC(R5) : Last reference to unit?
01 13 0732 1697      BEQL  CAN_DEV  : If EQL yes, shutdown the device
05 0734 1698      RSB      : Return to caller
0735 1699

```

```

0735 1701      .SBTTL CAN_DEV, Device shutdown routine
0735 1702
0735 1703      :++
0735 1704      : CAN_DEV - Device shutdown routine
0735 1705
0735 1706
0735 1707      : This routine is called to shutdown the CI device. All tributaries are
0735 1708      : zapped so that they will eventually run-down and be deleted.
0735 1709
0735 1710      : Inputs:      R3 = IRP address
0735 1711      :              R5 = UCB address
0735 1712
0735 1713      :              IPL = FIPL
0735 1714
0735 1715      : Outputs:   R0-R2 are clobbered.
0735 1716
0735 1717
0735 1718      :--
0735 1719 CAN_DEV:
0735 1720      BBCC      #UCBSV_CN_INITED,-          ; Shutdown the device
30 68 A5 E5 0735 1721      UCB$W_DEVSTS(R5),50$          ; Br if dev not initied
38 BB 073A 1722      PUSHR      #^M<R3,R4,R5>          ;
073C 1723      :
073C 1724      : Zap each tributary
073C 1725
54 53 OF DO 073C 1726      MOVL      #MAX_TRB-1,R3          ; Loop counter (zero indexed)
00A0 C543 DO 073F 1727 20$:      MOVL      UCB$_VEC_CDB(R5)[R3],R4      ; Get next CDB
02 13 0745 1728      BEQL      30$          ; Br if none
25 10 0747 1729      BSBB      ZAP_CDB          ; Cancel all I/O on trib
F3 53 F4 0749 1730 30$:      SOBGEQ   R3,20$          ; Loop
074C 1731      :
074C 1732      : Remove our listener
074C 1733
53 0090 C5 DO 074C 1734      MOVL      UCB$_LIS_CDT(R5),R3      ; Pick up listening CDT
OF 13 0751 1735      BEQL      40$          ; None
0090 C5 D4 0753 1736      CLRL      UCB$_LIS_CDT(R5)      ; and clear any trace
54 0084 C5 DO 0757 1737      MOVL      UCB$_PDT(R5),R4      ; PDT address, just in case
075C 1738      DISCONNECT          ; Clear our name out of table
0762 1739 40$:      :
0762 1740      : Clean up the UCB
0762 1741
FFCF 8F AA 0762 1742      BICW2     #^C<UCBSM_ONLINE!UCBSM_POWER>,- ;
64 A5 0766 1743      UCB$_STSTR5)          ; Reset status
0768 1744
38 BA 0768 1745      POPR      #^M<R3,R4,R5>          ; Restore registers
05 076A 1746 50$:      RSB          ; Return

```





```
3F 10 0804 1862      BSBB  DEA_LMEM      ; Get rid of it
F6 11 0806 1863      BRB   70$          ; Get next entry
      0808 1864
      0808 1865 80$:  ;
      0808 1866      ; Complete all Transmit IPR's
      0808 1867      ;
53 18 B4 0F 0808 1868  REMQUE @CDB_Q_XMT_IRP(R4),R3 ; Get next IRP
      05 1D 080C 1869  BVS   90$          ; If VS then none
      00CE 30 080E 1870  BSBW  ABORT_IRP_POST ; Abort the I/O request
      F5 11 0811 1871  BRB   80$          ; Loop
      0813 1872
      0813 1873 90$:  ;
      0813 1874      ; Idle the CDB
      0813 1875      ;
3F A4 00 90 0813 1876  MOVB  #CDB_C_IDLE,CDB_B_STA(R4) ; Reinit CDB state
      05 0817 1877  RSB
      0818 1878
```

```
0818 1880 .SBTTL MSG_FORK, Fork process for receipt of Seq Messages
0818 1881
0818 1882 :++
0818 1883 : MSG_FORK - Process received MSG
0818 1884 :
0818 1885 : Inputs:
0818 1886 :
0818 1887 : R1 = Bytes send/received
0818 1888 : R2 --> Start of user data
0818 1889 : R3 --> CDT
0818 1890 : R4 --> PDT
0818 1891 :
0818 1892 : IPL = FIPL
0818 1893 :
0818 1894 : Outputs:
0818 1895 :
0818 1896 : --
0818 1897 MSG_FORK:
0818 1898 DEALLOC_MSG_BUF_REG ; Deallocate the message buffer
05 0818 1899 RSB
081C 1900
```

```
081C 1902 .SBTTL DG_FORK, Fork process for receipt of DG
081C 1903
081C 1904 :++
081C 1905 : DG_FORK - Process received DG
081C 1906 :
081C 1907 : Inputs:
081C 1908 :
081C 1909 : R0 = 0 - Received a DG
081C 1910 : = 1 - Transmit finished
081C 1911 : R1 = Bytes send/received
081C 1912 : R2 --> Start of user data
081C 1913 : R3 --> CDT
081C 1914 : R4 --> PDT
081C 1915 :
081C 1916 : IPL = FIPL
081C 1917 :
081C 1918 : Outputs:
081C 1919 :
081C 1920 : --
081C 1921 DG_FORK:
54 5C A3 D0 081C 1922 MOVL CDT$L_AUXSTRUC(R3),R4 ; Pick up pointer to CDB
0820 1923 BEQL EMPTY ; Closed CDB, discard
53 52 01 C3 0822 1924 SUBL3 #1,R2,R3 ; Make a biased copy of msg ptr
55 52 20 C2 0826 1925 SUBL #32,R2 ; Reset R2 to head of PPD buffer
0829 1926 CVTWL 8(R2),R5 ; Get offset to CXB header
082D 1927 BGEQ 20$ ; If GEQ then bug
52 55 C0 082F 1928 ADDL R5,R2 ; Reset R2 to head of CXB buffer
0832 1929
0832 1930 10$: INCL R3 ; Advance to next byte
63 FF 8F 91 0834 1931 CMPB #-1,(R3) ; Pad byte ?
0838 1932 BNEQ DG ; If NEQ not pad byte
083A 1933 DECL R1 ; Reduce count
083C 1934 BGTR 10$ ; If LEQ then no data
083E 1935
083E 1936 20$: BUG_CHECK INCONSTATE,FATAL ; Illegal offset
0842 1937
50 52 D0 0842 1938 EMPTY: MOVL R2,R0 ; Pick up buffer
0845 1939 DEALLMEM:
00000000*GF 17 0845 1940 JMP G^COM$DRVDEALMEM ; Deallocate buffer
084B 1941
084B 1942 DG:
084B 1943 ; Update counters
084B 1944
084B 1945 ASSUME CDB_L_BSN EQ 4+CDB_L_BRC
084B 1946 ASSUME CDB_L_DBR EQ 4+CDB_L_BSN
084B 1947 ASSUME CDB_L_DBS EQ 4+CDB_L_DBR
084B 1948
55 40 A4 9E 084B 1949 MOVAB CDB_L_BRC(R4),R5 ; Point to receive counter base
084F 1950 BLBC R0,5$ ; If LBC, then rcv
55 03 50 E9 084F 1950
55 04 C0 0852 1951 ADDL #4,R5 ; Advance to xmt counter base
65 51 C0 0855 1952 5$: ADDL R1,(R5) ; Update byte count
0858 1953 BCC 10$ ; Br if no overflow
65 03 1E 0858 1953 MNEGL #1,(R5) ; Else, latch it
085A 1954 10$: INCL 8(R5) ; Update message count
085D 1955 BCC 20$ ; If CC, no overflow
08  A5 01 CE 0860 1956 MNEGL #1,8(R5) ; Else, latch it
55 50 A4 D0 0862 1957 MOVL CDB_L_UCB(R4),R5 ; Pick up ptr to UCB
0866 1958
```



```

1D 50 E8 086A 1959 BLBS R0,SEND_FORK ; IF LBS then xmt complete
      086D 1960
      086D 1961 ; RECEIVE complete - if there is a pending receive I/O request,
      086D 1962 ; complete it. Otherwise, queue the buffer.
      086D 1963
      086D 1964 ASSUME CDB_V_RUN EQ 0 ;
D1 3A A4 E9 086D 1965 BLBC CDB_W_STS(R4),EMPTY ; Br if trib not in RUN state
      3D A4 97 0871 1966 DECB CDB_B_RCV_FQ(R4) ; Dec the buffer count
OE A2 53 52 A3 0874 1967 SUBW3 R2,R3,CXB$W_OFFSET(R2) ; Store offset to message
      OC A2 51 B0 0879 1968 MOVW R1,CXB$W_LENGTH(R2) ; Set size
      53 20 B4 0F 087D 1969 REMQUE @CDB_Q_RCV_IRP(R4),R3 ; Remove waiting IRP
      2C 1C 0881 1970 BVC FINISH_RCV_IO ; If VC then gone one, finish
      0883 1971 ; the I/O & exit
      2C B4 62 OE 0883 1972 INSQUE (R2),@CDB_Q_RCV_MSG+4(R4) ; Queue receive msg for late
      0086 31 0887 1973 BRW FILLRCVLIST ; Fill the receive buffer pool
      088A 1974
      088A 1975 SEND_FORK:
      088A 1976 ;
      088A 1977 ; TRANSMIT completed. Locate and deque XMIT IRP and post it.
      088A 1978 ;
      088A 1979 ; NOTE: the IRP's may be returned out of sequence on a power fail.
      088A 1980 ;
50 51 10 9C 088A 1981 ROTL #16,R1,R0 ; Size in R0 high word
      50 01 B0 088E 1982 MOVW #SS$_NORMAL,R0 ; Status in low word
      51 18 A4 9E 0891 1983 MOVAB CDB_Q_XMT_IRP(R4),R1 ; Address queue header
      53 51 D0 0895 1984 20$: MOVL R1,R3 ; Make a copy
      53 63 D0 0898 1985 MOVL (R3),R3 ; Get next IRP
      51 53 D1 089B 1986 CMPL R3,R1 ; Back to head of queue?
      2C A3 0B 13 089E 1987 BEQL 50$ ; If EQL then yes, bugcheck
      53 52 D1 08A0 1988 CMPL R2,IRP$L_SVAPE(R3) ; Buffer address match?
      53 63 OF 08A4 1989 BNEQ 20$ ; If NEQ no, try again
      39 11 08A6 1990 REMQUE (R3),R3 ; Remove IRP from queue
      08A9 1991 BRB SUC_TRB_IOPOST ; Complete the I/O with trib
      08AB 1992 ; info stuffed into IOSI2
      08AB 1993
      08AB 1994 50$: BUG_CHECK INCONSTATE,FATAL
      08AF 1995

```

```

08AF 1997      .SBTTL FINISH_RCV_IO, Finish receive I/O processing
08AF 1998
08AF 1999      :++
08AF 2000      : FINISH_RCV_IO - Finish receive I/O processing
08AF 2001
08AF 2002
08AF 2003      : This routine finishes receive processing and sends the IRP back to IOPOST.
08AF 2004      : The receive free list is filled and a receive is started if needed.
08AF 2005
08AF 2006      : Inputs:      R2 = message buffer address
08AF 2007      :                R3 = IRP address
08AF 2008      :                R4 = CDB address
08AF 2009      :                R5 = UCB address
08AF 2010
08AF 2011      :                IPL = FIPL
08AF 2012
08AF 2013      : Outputs:     R0-R4 are clobbered. All other registers are preserved.
08AF 2014
08AF 2015
08AF 2016      :--
08AF 2017      : FINISH_RCV_IO:
08AF 2018      : MOVL R2,IRPSL_SVAPTE(R3)      : Finish receive I/O request
08AF 2019      : MOVZWL CXBSW_OFFSET(R2),(R2)  : Save block address
08AF 2020      : ADDL R2,(R2)                  : Store offset to message
08AF 2021      : MOVL IRPSL_IOST2(R3),4(R2)    : Make it a pointer
08AF 2022      : MOVW CXBSW_LENGTH(R2),R1      : Set address of user buffer
08AF 2023      : CMPW R1,IRPSW_BCNT(R3)        : Get size of transfer
08AF 2024      : BGTRU 10$                     : Request larger than actual?
08AF 2025      : MOVW R1,IRPSW_BCNT(R3)        : Br GTRU then yes
08AF 2026      : 10$: MOVL IRPSW_BCNT-2(R3),R0  : Set size to transfer
08AF 2027      : MOVW #SS$_NORMAL,R0          : Setup size of xfer in high word
08AF 2028      : BNEQ SUC_TRB_IOPOST          : Setup status in low word
08AF 2029      : MOVZWL #SS$_CTRLERR,R0       : Br if success
08AF 2030      : CLRL R1                       : Set data path error
08AF 2031      : BRB IOPOST                    : Init second longword
08AF 2032      :                               : Post it
08AF 2033      : ABORT_IRP_POST:
08AF 2034      : MOVQ S^#SS$_ABORT,R0         : Setup IOSB image
08AF 2035      : BRB IOPOST                    : Finish up
08AF 2036
08AF 2037
08AF 2038      : SUC_TRB_IOPOST:
08AF 2039      : MOVL #XMSM_STS_ACTIVE!-      : Successful Trib I/O completion
08AF 2040      : XMSM_STS_RUNNING,R1         : Set device dependent bits to indicate
08AF 2041      : CMPB #RBFTR,CDB_B_RCV_FQ(R4) : that the circuit is running
08AF 2042      : BLEQU IOPOST                 : Receive queue under threshold?
08AF 2043      : #XMSM_STS_BUFFAIL,R1        : If LEQU then no
08AF 2044      : MOVQ R0,IRPSL_IOST1(R3)      : Signal buffer threshold problems
08AF 2045      : IOPOST: MOVQ R0,IRPSL_IOST1(R3) : Store IOSB image
08AF 2046      : JMP G^COM$POST              : Post IRP
0902 2046

```

```

2C A3 52 DO
62 OE A2 3C
04 A2 3C A3 DO
51 OC A2 B0
32 A3 51 B1
32 A3 51 B0
50 30 A3 DO
50 01 B0
50 0054 8F 3C
51 D4
19 11
50 2C 7D
14 11
51 00002800 8F DO
3D A4 06 91
07 1B
51 00001000 8F C8
38 A3 50 7D
00000000 GF 17

```

```

0902 2048 .SBTTL FILLRCVLIST, Fill receive buffer list
0902 2049 .SBTTL ADDRCLIST, Move IRP buffer to free list
0902 2050 :++
0902 2051 : FILLRCVLIST - Add to the receive buffer list
0902 2052 : ADDRCLIST - Add IRP buffer to free list
0902 2053 :
0902 2054 : This routine is entered to make sure that the receive buffer pool is full.
0902 2055 : If it is not, buffers are allocated and queued to the list until it is.
0902 2056 :
0902 2057 : For ADDRCLIST, any buffer attached to the IRP is added to the free list
0902 2058 : even if the list is already filled.
0902 2059 :
0902 2060 : Inputs: R3 - IRP address (ADDRCLIST only)
0902 2061 : R4 - CDB address
0902 2062 : R5 - UCB address
0902 2063 :
0902 2064 : Outputs: Only R0-R2 are clobbered.
0902 2065 :
0902 2066 :--
0902 2067 .ENABL LSB
0902 2068 ADDRCLIST:
52 2C A3 DO 0902 2069 MOVL IRP$S_SVAPE(R3),R2 ; Add IRP buffer to free list
      08 13 0906 2070 BEQL FILLRCVLIST ; Get buffer, if any
      2C A3 D4 0908 2071 CLRL IRP$S_SVAPE(R3) ; If none, fill rcv list if needed
      1E 11 090B 2072 PUSHQ R3 ; Detach the buffer
      090E 2073 BRB 20$ ; Save regs
      0910 2074 ; Add buffer to free list
0910 2075 FILLRCVLIST:
3C A4 3D A4 91 0910 2076 PUSHQ R3 ; Save regs
51 38 A4 004C 8F A1 0913 2077 10$: CMPB CDB_B_RCV_FQ(R4),CDB_B_RCV_CNT(R4) ; Should new block be added?
      00000000 GF 16 0918 2078 BGEQU 50$ ; If GEQU no - list filled
      08 A2 51 B0 091A 2079 ADDW3 #CXBS$OVERHEAD,CDB_W_BUFSIZ(R4),R1 ; Compute block size need
      0921 2080 JSB G^EXE$ALONONPAGED ; Allocate nonpaged memory
      0927 2081 BLBC R0,50$ ; If LBC then failure
      092A 2082 MOVW R1,IRP$W_SIZE(R2) ; Insert block size
      092E 2083 20$: ;
      092E 2084 ; Give SCS receive datagram
      092E 2085 ;
20 A3 53 54 A4 DO 092E 2086 MOVL CDB_L_CDT(R4),R3 ; Pick up CDT address
      009E C5 91 0932 2087 CMPB UCBS$B_CN_PORT(R5),CDT$B_RSTATION(R3) ; Talking to ourselves?
      05 12 0938 2088 BNEQ 30$ ; If NEQ, no
      53 0094 C5 DO 093A 2089 MOVL UCBS$L_TWING_CDT(R5),R3 ; Yes, use other CDT
      54 0084 C5 DO 093F 2090 30$: MOVL UCBS$L_PDT(R5),R4 ; and PDT address
      0A A2 1B 9B 0944 2091 MOVZBW S^#DYN$C_CXB,IRP$B_TYPE(R2) ; Insert block type
      0948 2092 QUEUE_DG_BUF ; Put the block on the free que
      094B 2093 BLBC R0,40$ ; Br if failure
      54 04 AE DO 094E 2094 MOVL 4(SP),R4 ; Pick up CDB pointer
      3D A4 96 0952 2095 INCB CDB_B_RCV_FQ(R4) ; Bump free que count
      BC 11 0955 2096 BRB 10$ ; Try for more
      50 52 DO 0957 2097 40$: MOVL R2,R0 ; Pick up the buffer
      03 13 095A 2098 BEQL 50$ ; There is none
      FEE6 30 095C 2099 BSBW DEALLMEM ; Get rid of the buffer
      095F 2100 ;
      095F 2101 50$: POPQ R3 ; Restore regs
      05 0962 2102 RSB ; Return
      0963 2103 ;
      0963 2104 .DSABL LSB

```

```

0963 2106          .SBTTL XLATE, Translate Channel to CDB address
0963 2107
0963 2108      :++
0963 2109      : XLATE - Translate Channel to CDB address
0963 2110
0963 2111      : This routine is called to return the CDB address for a particular
0963 2112      : channel.
0963 2113
0963 2114
0963 2115      : Inputs:      R3 = IRP address
0963 2116      :                R5 = UCB address
0963 2117
0963 2118      : Outputs:     R0 - status return for success of call.
0963 2119
0963 2120      :                R9 = CDB address if successful
0963 2121      :                1 otherwise
0963 2122
0963 2123      :                R1,R2 are clobberd, all other registers are preserved.
0963 2124
0963 2125      :--
0963 2126 XLATE:
50  28  A3  3C 0963 2127      MOVZWL  IRP$W_CHAN(R3),R0      : Translate CHAN into CDB address
                                BSBB      XLATE_CHAN      : Get channel
40  A3   51  90 0967 2128      BSBB      XLATE_CHAN      : Translate channel
54  A3   59  D0 0969 2129      MOVVB    R1,IRP$B_INDEX(R3)  : Save index in IRP
                                DO 096D 2130      MOVL     R9,IRP$L_CDB(R3)  : Store CDB address in IRP
                                05 0971 2131      RSB          : Return to caller
0972 2132
0972 2133 XLATE_CHAN:
00E0 C541 51 0F 9A 0972 2134      MOVZBL  #MAX_TRB-1,R1      : Setup loop counter (zero indexed)
                                B1 0975 2135 10$:  CMPW    R0,UCB$W_VEC_CHAN(R5)[R1] : Channels match?
                                13 0978 2136      BEQL    40$          : Br if yes - got it
                                F4 097D 2137      SOBGEQ  R1,10$        : Loop
50  20D4 8F 3C 0980 2138 30$:  MOVZWL  #$$$_DEVINACT,R0    : Return channel offline
                                59  01  D0 0985 2139      MOVL     #1,R9          : Setup 'R9 invalid' flag
                                05 0988 2140      RSB          : And leave
0989 2141
0989 2142 40$:  :
0989 2143      : Found match on channel
0989 2144      :
59  00A0 C541 51 0F 9A 0989 2145      MOVL     UCB$L_VEC_CDB(R5)[R1],R9 : Get CDB address
                                13 098F 2146      BEQL    50$          : Br if no CDB address - error
                                3C 0991 2147      MOVZWL  S^#$$$_NORMAL,R0    : Set successful return status
                                05 0994 2148      RSB          : Return
0995 2149
0995 2150 50$:  BUG_CHECK  INCONSTATE,FATAL
0999 2151
  
```

```

0999 2153          .SBTTL VALIDATE_P2, Validate P2 buffer parameters
0999 2154
0999 2155      :++
0999 2156      : VALIDATE_P2 - Validate P2 buffer parameters
0999 2157      :
0999 2158      : This routine is called to validate the P2 buffer parameters. The parameters
0999 2159      : are checked against a parameter table which verifies that the minimum value
0999 2160      : and maximum value is not violated, and that status flags are set or clear
0999 2161      : as required.
0999 2162      :
0999 2163      : The way in this routine is written, the require word of the verification
0999 2164      : table can only have 1 bit set at a time.
0999 2165      :
0999 2166      : Inputs:          R1 = Address of parameter verification table
0999 2167                  R2 = Status word from UCB or CDB
0999 2168                  R3 = IRP address
0999 2169                  R5 = UCB address
0999 2170                  R9 = If low bit clear then ptr to context block (CDB or UCB)
0999 2171                      If low bit set then no context block exists
0999 2172      :
0999 2173                  IPL = FIPL or ASTDEL
0999 2174      :
0999 2175      : Outputs:       R0 = status return of parameters
0999 2176                  R1 = i.d. of parameter causing problem on error
0999 2177      :
0999 2178                  All other registers are preserved.
0999 2179      :
0999 2180      :--
0999 2181 VALIDATE_P2:
0999 2182     01EA 8F  BB 0999 2182     PUSHR    #^M<R1,R3,R5,R6,R7,R8>    : Save registers
0999 2183     56  2C B3  DO 0999 2183     : NB:R1 must be on top of stack
0999 2184     58  32 A3  3C 0999 2184     MOVL     @IRPSL_SVAPE(R3),R6    : Get system P2 buffer address
0999 2185     58  06  C6  09A1 2185     MOVZWL  IRPSW_BCNT(R3),R8    : Get size of P2 buffer
0999 2186     4F  11  09A5 2186     DIVL    #6,R8              : Get number of params in P2
0999 2187     09AB 2187     BRB     40$              : Treat as none if too few bytes
0999 2188     10$:
0999 2189     09AA 2189     :
0999 2190     09AA 2190     : Loop to check next parameter in P2 buffer
0999 2191     50  86  3C 09AA 2191     MOVZWL  (R6)+,R0            : Get parameter type from P2
0999 2192     55  86  DO 09AD 2192     MOVL    (R6)+,R5          : Get parameter value from P2
0999 2193     57  6E  DO 0980 2193     MOVL    (SP),R7           : Get parameter table address
0999 2194     20$:
0999 2195     09B3 2194     :
0999 2196     09B3 2196     : Loop to check P2 buffer parameter to circuit parameter table
0999 2197     51  87  F000 8F AB 09B3 2197     BICW3   #^C<PRM_M_TYPE>,(R7)+,R1 : Get next param i.d.
0999 2198     46  13  09B9 2198     BEQL    50$              : If EQL, at end of table
0999 2199     51  50  B1 09BB 2199     CMPW    R0,R1            : Parameters match?
0999 2200     05  13  09BE 2200     BEQL    30$              : Br if yes
0999 2201     57  0A  C0 09C0 2201     ADDL2   #PARAM_C_ENTRY-2,R7 : Else, skip to next parameter
0999 2202     EE  11  09C3 2202     BRB     20$              : Try next parameter
0999 2203     30$:
0999 2204     09C5 2203     :
0999 2205     09C5 2204     : Match found - check min,max,valid,invalid
0999 2206     53  87  3C 09C5 2206     MOVZWL  (R7)+,R3          : Get offset/width
0999 2207     15  59  EB 09C8 2207     BLBS    R9,33$           : If LBS then no current block
0999 2208     50  53  0A 00  EF 09CB 2208     EXTZV   #OFF_V_VALUE,#OFF_S_VALUE,R3,R0 : Get offset
0999 2209     53  53  06 0A  EF 09D0 2209     EXTZV   #OFF_V_WIDTH,#OFF_S_WIDTH,R3,R3 : Get width

```

53	6940	53	00	EF	09D5	2210		EXTZV	#0,R3,(R9)[R0],R3	:	Get current value
		53	55	B1	09DB	2211		CMPW	R5,R3	:	Value change ?
			19	13	09DE	2212		BEQL	40\$	:	If EQL no, try next param
		87	55	B1	09E0	2213	33\$:	CMPW	R5,(R7)+	:	Is the value too small?
			1C	1F	09E3	2214		BLSSU	50\$	:	Br if yes - error
		87	55	B1	09E5	2215		CMPW	R5,(R7)+	:	Is the value too big?
			17	1A	09E8	2216		BGTRU	50\$	:	Br if yes - error
		55	87	B0	09EA	2217		MOVW	(R7)+,R5	:	Pick up required
			05	13	09ED	2218		BEQL	35\$	:	None
		52	55	B3	09EF	2219		BITW	R5,R2	:	Check required bit
			0D	13	09F2	2220		BEQL	50\$	:	Br if not on - error
		52	87	B3	09F4	2221	35\$:	BITW	(R7)+,R2	:	Check invalid bits
			08	12	09F7	2222		BNEQ	50\$	:	Br if on - error
		AE	58	F5	09F9	2223	40\$:	SOBGTR	R8,10\$	:	Br if more parameters
		50	01	3C	09FC	2224		MOVZWL	S^#SS\$_NORMAL,R0	:	Set success return
			06	11	09FF	2225		BRB	60\$	:	And return
					0A01	2226				:	
		6E	51	3C	0A01	2227	50\$:	MOVZWL	R1,(SP)	:	Return bad parameter type
		50	14	D0	0A04	2228		MOVL	#SS\$_BADPARAM,R0	:	Set error return
		01EA	8F	BA	0A07	2229	60\$:	POPR	#*M<R1,R3,R5,R6,R7,R8>	:	Restore registers
				05	0A0B	2230		RSB		:	Return to caller

```

OAOC 2232      .SBTTL UNPACK_P2_BUF, Unpack a P2 parameter from P2 buffer
OAOC 2233
OAOC 2234      :++
OAOC 2235      : UNPACK_P2_BUF - Unpack a P2 parameter from P2 buffer
OAOC 2236
OAOC 2237      :
OAOC 2238      : This routine is called to get a P2 parameter from the P2 buffer.
OAOC 2239      :
OAOC 2240      : Inputs:      R1 = Parameter type code
OAOC 2241      :                R3 = IRP address
OAOC 2242      :                R5 = UCB address
OAOC 2243      :
OAOC 2244      :                IPL = IPL$_ASTDEL to allow user paging.
OAOC 2245      :
OAOC 2246      : Outputs:     R0 = $$$_NORMAL if successful
OAOC 2247      :                $$$_INSFARG otherwise
OAOC 2248      :                R2 = Parameter value if success else destroyed
OAOC 2249      :
OAOC 2250      :                All other registers are preserved.
OAOC 2251      :
OAOC 2252      :--
OAOC 2253      UNPACK_P2_BUF:                                ; Unpack P2 buffer
56 00E0 8F  BB OAOC 2254      POSHR   #*M<R5,R6,R7>                ; Save registers
57 2C B3  DO OA10 2255      MOVL   @IRP$L_SVAPTE(R3),R6        ; Get system P2 buffer address
57 32 A3  3C OA14 2256      MOVZWL IRP$W_BCNT(R3),R7         ; Get size of P2 buffer
57 06 C6  OA18 2257      DIVL   #6,R7                    ; Get number of params in P2
57 11 13  OA1B 2258      BEQL   20$                          ; Treat as none if too few bytes
50 01 3C  OA1D 2259      MOVZWL S*$_NORMAL,R0           ; Assume success
OA20 2260      10$:
OA20 2261      : Loop to check next parameter in P2 buffer
OA20 2262      :
55 86 3C  OA20 2263      MOVZWL (R6)+,R5                ; Get parameter type from P2
52 86 DO  OA23 2264      MOVL   (R6)+,R2                ; Get parameter value from P2
55 51 B1  OA26 2265      CMPW   R1,R5                    ; Parameters match?
55 08 13  OA29 2266      BEQL   30$                          ; Br if yes
F2 57 F5  OA2B 2267      SOBGTR R7,10$                ; Br if more parameters
OA2E 2268      :
50 0114 8F 3C OA2E 2269      20$: MOVZWL #$_INSFARG,R0           ; Return error
00E0 8F BA OA33 2270      30$: POPR   #*M<R5,R6,R7>        ; Restore registers
05 OA37 2271      RSB                                ; Return to caller

```

```
0A38 2273 .SBTTL CN_END, End of driver
0A38 2274
00000A40 0A38 2275 . = <.+15>8<-16>
0A40 2276
00000018 0A40 2277 PATCH:: .LONG 32-8
00000A48 0A44 2278 .LONG PATCH+8
00000A60 0A48 2279 .BLKB 32-8
0A60 2280
0A60 2281 :++
0A60 2282 : Label that marks the end of the driver
0A60 2283 :--
0A60 2284
0A60 2285 CN_END: ; Last location in driver
0A60 2286
0A60 2287 .END
```



CNDRIVER  
Symbol table

- VAX/VMS DECnet-CI Class Driver

C 14

16-SEP-1984 01:19:27 VAX/VMS Macro V04-00  
5-SEP-1984 00:11:06 [DRIVER.SRC]CNDRIVER.MAR;1

Page 54  
(39)

CC  
V(

\$\$\$	=	00000020	R	02	CDB_W_STS	0000003A		
\$\$\$NUM	=	00000000			CDT\$B-RSTATION	=	00000020	
\$\$\$OFF	=	0000004C			CDT\$L-AUXSTRUC	=	0000005C	
\$\$\$TYP	=	000063F3			CDT\$L-PBT	=	00000010	
\$\$OP	=	00000002			CHECK-REMOTE	000006CF	R	03
ABORT_IRP_POST		000008DF	R	03	CLR_IRP	00000115	R	03
ABORT_REQ		000002B8	R	03	CN\$DDT	00000000	RG	03
ABORT_REQ1		000002B8	R	03	CN_END	00000A60	R	03
ABORT_START		00000226	R	03	CN-FUNCTABLE	00000038	R	03
ACCEPT		00000626	R	03	COM\$DRVDEALMEM	*****	X	03
ADDRCVLIST		00000902	R	03	COM\$POST	*****	X	03
ALT_START		000001B8	R	03	CONN	00000591	R	03
AT\$ NULL	=	00000005			CONN_ABO	000005E4	R	03
BUG\$ INCONSTATE		*****	X	03	CONN_DATA	00000100	R	03
CANCEL		0000070E	R	03	CONN_ERR	000006F0	R	03
CAN_DEV		00000735	R	03	CONN_FIN	00000693	R	03
CDB_B_DUMMY		0000005A			CONN_FIN1	00000697	R	03
CDB_B_FIPL		0000000B			COUNT_C_ENTRY	=	00000004	
CDB_B_RCV_CNT		0000003C			CR\$SL_INTD	=	00000024	
CDB_B_RCV_FQ		0000003D			CXB\$C_HEADER	=	00000048	
CDB_B_REMSYS		00000059			CXB\$C_OVERHEAD	=	0000004C	
CDB_B_REMVER		00000058			CXB\$W_LENGTH	=	0000000C	
CDB_B_RSTCNT		0000005B			CXB\$W_OFFSET	=	0000000E	
CDB_B_STA		0000003F			DC\$ SCOM	=	00000020	
CDB_B_TRB_ADDR		0000003E			DDB\$L_DDT	=	0000000C	
CDB_B_TYPE		0000000A			DEALLMEM	00000845	R	03
CDB_C_ACPT	=	00000004			DEVSM_IDV	=	04000000	
CDB_C_CONN	=	00000002			DEVSM_NET	=	00002000	
CDB_C_FIPL	=	00000006			DEVSM_ODV	=	08000000	
CDB_C_IDLE	=	00000000			DEVSM_REC	=	00000001	
CDB_C_LENGTH	=	00000060			DG	0000084B	R	03
CDB_C_LSTN	=	00000003			DG_FORK	0000081C	R	03
CDB_C_OPEN	=	00000001			DPT\$C_LENGTH	=	00000038	
CDB_L_ABSTIME		00000034			DPT\$C_VERSION	=	00000004	
CDB_L_BRC		00000040			DPT\$INITAB	00000038	R	02
CDB_L_BSN		00000044			DPT\$M_SCS	=	00000008	
CDB_L_CDT		00000054			DPT\$REINITAB	00000062	R	02
CDB_L_DBR		00000048			DPT\$TAB	00000000	R	02
CDB_L_DBS		0000004C			DYN\$C_CIDG	=	0000003B	
CDB_L_FPC		0000000C			DYN\$C_CRB	=	00000005	
CDB_L_FR3		00000010			DYN\$C_CXB	=	0000001B	
CDB_L_FR4		00000014			DYN\$C_DDB	=	00000006	
CDB_L_SETMODE		00000030			DYN\$C_DPT	=	0000001E	
CDB_L_UCB		00000050			DYN\$C_NET	=	00000017	
CDB_M_RUN	=	00000001			DYN\$C_ORB	=	00000049	
CDB_Q_FORK		00000000			DYN\$C_UCB	=	00000010	
CDB_Q_RCV_IRP		00000020			EMPTY	00000842	R	03
CDB_Q_RCV_MSG		00000028			EXESABORTIO	*****	X	03
CDB_Q_XMT_IRP		00000018			EXESALONONPAGED	*****	X	03
CDB_V_ACPT	=	00000002			EXESBUFFRQUOTA	*****	X	03
CDB_V_CONN	=	00000001			EXESFINISHIO	*****	X	03
CDB_V_DISC	=	00000003			EXESFORK	*****	X	03
CDB_V_REJECT	=	00000004			EXESGL_ABSTIM	*****	X	03
CDB_V_RUN	=	00000000			EXESQIDRETURN	*****	X	03
CDB_W_BUFSIZ		00000038			EXESREADCHK	*****	X	03
CDB_W_REMPROT	=	00000058			EXESWRITECHK	*****	X	03
CDB_W_SIZE		00000008			FILLRCVLIST	00000910	R	03

CNDRIVER  
Symbol table

- VAX/VMS DECnet-CI Class Driver

D 14

16-SEP-1984 01:19:27 VAX/VMS Macro V04-00  
5-SEP-1984 00:11:06 [DRIVER.SRC]CNDRIVER.MAR;1

Page 55  
(39)

C  
V

FINISH_ERR	000002C4	R	03	NMASC_CTCIR_DBR	=	000003F2		
FINISH_RCV_IO	000008AF	R	03	NMASC_CTCIR_DBS	=	000003F3		
FINISH_REQ	000002D5	R	03	NMASC_DPX_FDL	=	00000000		
FINISH_SUC	000002C1	R	03	NMASC_DPX_HAL	=	00000001		
FUNCTAB_LEN	= 0000004C			NMASC_LINCN_LOO	=	00000001		
GET_BUF	0000012D	R	03	NMASC_LINCN_NOR	=	00000000		
GET_CHAR_RBUF	000004D9	R	03	NMASC_PCCI_MRB	=	00000479		
GET_CHAR_WBUF	000004DD	R	03	NMASC_PCCI_MST	=	00000AFA		
IOSV_CLR_COUNT	= 0000000A			NMASC_PCCI_TRI	=	00000474		
IOSV_CTRC	= 00000009			NMASC_PCLI_BFN	=	00000451		
IOSV_RD_COUNT	= 00000008			NMASC_PCLI_BUS	=	00000AF1		
IOSV_SHUTDOWN	= 00000007			NMASC_PCLI_CON	=	00000456		
IOSV_STARTUP	= 00000006			NMASC_PCLI_DUP	=	00000457		
IOS_READBLK	= 00000021			NMASC_STATE_OFF	=	00000001		
IOS_SENSEMODE	= 00000027			NMASC_STATE_ON	=	00000000		
IOS_SETCHAR	= 0000001A			NMASH_CNT_COU	=	00008000		
IOS_SETMODE	= 00000023			NMASV_CNT_WID	=	0000000D		
IOS_VIRTUAL	= 0000003F			OFF_M_VALUE	=	000003FF		
IOS_WRITEBLK	= 00000020			OFF_S_VALUE	=	0000000A		
IOCSMNTVER	*****	X	03	OFF_S_WIDTH	=	00000006		
IOCSRETURN	*****	X	03	OFF_V_VALUE	=	00000000		
IOPOST	000008F8	R	03	OFF_V_WIDTH	=	0000000A		
IRPSB_INDEX	00000040			OLD_C_PROT	=	00000000		
IRPSB_RMOD	= 0000000B			ORBSB_FLAGS	=	0000000B		
IRPSB_TYPE	= 0000000A			ORBSL_OWNER	=	00000000		
IRPSL_BCNT	= 00000032			ORBSM_PROT_16	=	00000001		
IRPSL_CDB	00000054			ORBSW_PROT	=	00000018		
IRPSL_EXTEND	= 00000054			P1	=	00000000		
IRPSL_IOST1	= 00000038			P2	=	00000004		
IRPSL_IOST2	= 0000003C			PARAM_C_ENTRY	=	0000000C		
IRPSL_SEGVBN	= 00000048			PATCH	=	00000A40	RG	03
IRPSL_SVAPE	= 0000002C			PBSL_PDT	=	0000002C		
IRPSM_FUNC	= 00000002			PCBSL_JIB	=	00000080		
IRPSQ_NT_PRVMSK	= 00000040			PDTSL_DEALRMSG	=	00000024		
IRPSV_FUNC	= 00000001			PDTSL_QUEUEDG	=	0000003C		
IRPSW_BCNT	= 00000032			PDTSL_REJECT	=	0000004C		
IRPSW_BOFF	= 00000030			PDTSL_SENDRGDG	=	0000007C		
IRPSW_CHAN	= 00000028			PR\$ IPL	=	00000012		
IRPSW_FUNC	= 00000020			PRM_M_INVALID	=	00008000		
IRPSW_SIZE	= 00000008			PRM_M_MAX	=	00002000		
IRPSW_STS	= 0000002A			PRM_M_MIN	=	00001000		
JIBSL_BYTCNT	= 00000020			PRM_M_REQUIRE	=	00004000		
LINE_CNT_NUM	= 00000000			PRM_M_TYPE	=	00000FFF		
LINE_CNT_TABLE	= 000000EE	R	03	PROC_C_NAM	=	00000006		
LINE_PRH_NUM	= 00000004			PROC_NAM	=	000000F0	R	03
LINE_PRM_TABLE	= 000000AA	R	03	QIORET	=	000002B2	R	03
LISTEN	000003AF	R	03	RBFMAX	=	0000001F		
LIS_ERR	0000041F	R	03	RBFMIN	=	00000009		
LIS_FORK	00000602	R	03	RBFTHR	=	00000006		
MASKH	= 00000080			RCV_FDT	=	00000184	R	03
MASKL	= 00000000			RCV_START	=	00000229	R	03
MAX_TRB	= 00000010			REJECT	=	0000061D	R	03
MSG_FORK	00000818	R	03	SBSL_PBCONNX	=	00000014		
NEW_CDB	0000030A	R	03	SBCSB_RSTATION1	=	0000003C		
NEW_TRIB	000002DB	R	03	SBO\$C_LENGTH	=	00000050		
NMASC_CTCIR_BRC	= 000003E8			SBO_LNG	=	00000070		
NMASC_CTCIR_BSN	= 000003E9			SCSBACCEPT	=	*****	GX	03

CNDRIVER  
Symbol table

- VAX/VMS DECnet-CI Class Driver E 14

16-SEP-1984 01:19:27 VAX/VMS Macro V04-00  
5-SEP-1984 00:11:06 [DRIVER.SRC]CNDRIVER.MAR;1

SCSS\$CONFIG SYS	*****	X	03	VALIDATE_P2	00000999	R	03
SCSS\$CONNECT	*****	X	03	VECSL_UNITINIT	= 00000018		
SCSS\$DISCONNECT	*****	X	03	XLATE-	00000963	R	03
SCSS\$GB_SYSTEMID	*****	X	03	XLATE_CHAN	00000972	R	03
SCSS\$GW_MAXDG	*****	X	02	XMSM_STS_ACTIVE	= 00000800		
SCSS\$LISTEN	*****	GX	03	XMSM_STS_BUFFAIL	= 00001000		
SEND_FORK	0000088A	R	03	XMSM_STS_RUNNING	= 00002000		
SENSEMODE_FDT	0000042E	R	03	XMT_FDT	00000125	R	03
SENSE_C_BUF	= 00000080			XMT_RCV_FDT_CO	00000190	R	03
SENSE_TABLE	00000426	R	03	XMT_START	000001D2	R	03
SETMODE_CTRL	0000035E	R	03	ZAP_CDB	0000076E	R	03
SETMODE_FDT	0000023D	R	03	ZAP_CDB_R9	0000076B	R	03
SIZ...	= 00000001						
SS\$ABORT	= 0000002C						
SS\$ACCVIO	= 0000000C						
SS\$BADPARAM	= 00000014						
SS\$BUFFEROVF	= 00000601						
SS\$CTRLERR	= 00000054						
SS\$DEACTIVE	= 000002C4						
SS\$DEVALRALLOC	= 00000641						
SS\$DEVINACT	= 000020D4						
SS\$DEVOFFLINE	= 00000084						
SS\$INSFARG	= 00000114						
SS\$NORMAL	= 00000001						
SS\$REMRSRC	= 0000206C						
START_TRIB	0000052E	R	03				
SUC_TRB_IOPST	000008E4	R	03				
TRIB_CNT_NUM	= 00000004						
TRIB_CNT_TABLE	= 000000DC	R	03				
TRIB_PRM_NUM	= 00000003						
TRIB_PRM_TABLE	= 00000084	R	03				
UCBS\$CN_PORT	0000009E						
UCBS\$DEVCLASS	= 00000040						
UCBS\$DIPL	= 0000005E						
UCBS\$FIPL	= 0000000B						
UCBS\$RCV_CNT	0000009F						
UCBS\$CN_LENGTH	= 00000100						
UCBS\$LENGTH	= 00000090						
UCBS\$DEVCHAR	= 00000038						
UCBS\$DGHDRSZ	00000098						
UCBS\$LIS_CDT	00000090						
UCBS\$PDT	= 00000084						
UCBS\$TWIN_CDT	00000094						
UCBS\$VEC_CDB	000000A0						
UCBS\$CN_INITED	= 00000001						
UCBS\$ONLINE	= 00000010						
UCBS\$POWER	= 00000020						
UCBS\$CN_INITED	= 00000000						
UCBS\$POWER	= 00000005						
UCBS\$DEVBUFSIZ	= 00000042						
UCBS\$DEVSTS	= 00000068						
UCBS\$DUMMY	0000009C						
UCBS\$REFC	= 0000005C						
UCBS\$STS	= 00000064						
UCBS\$VEC_CHAN	000000E0						
UNIT_INIT	00000110	R	03				
UNPACK_P2_BUF	00000A0C	R	03				

-----  
! Psect synopsis !  
-----

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$AB\$\$	00000100 ( 256.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$105_PROLOGUE	0000006D ( 109.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$115_DRIVER	00000A60 ( 2656.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

-----  
! Performance indicators !  
-----

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.07	00:00:00.85
Command processing	113	00:00:00.39	00:00:04.55
Pass 1	811	00:00:26.09	00:01:43.79
Symbol table sort	0	00:00:03.73	00:00:16.44
Pass 2	417	00:00:05.69	00:00:23.01
Symbol table output	15	00:00:00.20	00:00:00.82
Psect synopsis output	0	00:00:00.02	00:00:00.06
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1387	00:00:36.20	00:02:29.55

The working set limit was 2400 pages.  
 217048 bytes (424 pages) of virtual memory were used to buffer the intermediate code.  
 There were 200 pages of symbol table space allocated to hold 3593 non-local and 117 local symbols.  
 2287 source lines were read in Pass 1, producing 22 object records in Pass 2.  
 68 pages of virtual memory were used to define 62 macros.

-----  
! Macro library statistics !  
-----

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	36
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	14
TOTALS (all libraries)	50

3866 GEIS were required to define 50 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:CNDRIVER/OBJ=OBJ\$:CNDRIVER MSRC\$:CNDRIVER/UPDATE=(ENH\$:CNDRIVER)+EXECMLS/LIB

The image displays a dense grid of small, illegible text fragments, likely representing a large document or code page. The text is arranged in a regular pattern across the page. Some larger, faint text is visible, including "XDRIVER MAR", "CONTERR LIS", and "NORTVER LIS". The overall appearance is that of a high-resolution scan of a document with very small characters.