

RRRRRRRRRRRR		EEEEEEEEEEEEEE	MMM	MMM
RRRRRRRRRRRR		EEEEEEEEEEEEEE	MMM	MMM
RRRRRRRRRRRR		EEEEEEEEEEEEEE	MMM	MMM
RRR	RRR	EEE	MMMMMM	MMMMMM
RRR	RRR	EEE	MMMMMM	MMMMMM
RRR	RRR	EEE	MMMMMM	MMMMMM
RRR	RRR	EEE	MMM	MMM
RRR	RRR	EEE	MMM	MMM
RRR	RRR	EEE	MMM	MMM
RRRRRRRRRRRR		EEEEEEEEEEEEEE	MMM	MMM
RRRRRRRRRRRR		EEEEEEEEEEEEEE	MMM	MMM
RRRRRRRRRRRR		EEEEEEEEEEEEEE	MMM	MMM
RRR	RRR	EEE	MMM	MMM
RRR	RRR	EEE	MMM	MMM
RRR	RRR	EEE	MMM	MMM
RRR	RRR	EEE	MMM	MMM
RRR	RRR	EEE	MMM	MMM
RRR	RRR	EEE	MMM	MMM
RRR	RRR	EEE	MMM	MMM
RRR	RRR	EEEEEEEEEEEEEE	MMM	MMM
RRR	RRR	EEEEEEEEEEEEEE	MMM	MMM
RRR	RRR	EEEEEEEEEEEEEE	MMM	MMM

[illegible]

```
RRRRRRRR  EEEEEEEEE  MM      MM  XX      XX  PPPPPPPP  000000  RRRRRRRR  TTTTTTTTTT  DDDDDDDD
RRRRRRRR  EEEEEEEEE  MM      MM  XX      XX  PPPPPPPP  000000  RRRRRRRR  TTTTTTTTTT  DDDDDDDD
RR      RR  EE      MM  MM  MM  XX      XX  PP      PP  00      00  RR      RR  TT      DD      DD
RR      RR  EE      MM  MM  MM  XX      XX  PP      PP  00      00  RR      RR  TT      DD      DD
RR      RR  EE      MM  MM  MM  XX      XX  PP      PP  00      00  RR      RR  TT      DD      DD
RRRRRRRR  EEEEEEEEE  MM      MM  XX      XX  PPPPPPPP  00      00  RRRRRRRR  TT      DD      DD
RRRRRRRR  EEEEEEEEE  MM      MM  XX      XX  PPPPPPPP  00      00  RRRRRRRR  TT      DD      DD
RR      RR  EE      MM      XX      XX  PP      PP  00      00  RR      RR  TT      DD      DD
RR      RR  EE      MM      XX      XX  PP      PP  00      00  RR      RR  TT      DD      DD
RR      RR  EE      MM      XX      XX  PP      PP  00      00  RR      RR  TT      DD      DD
RR      RR  EE      MM      XX      XX  PP      PP  00      00  RR      RR  TT      DD      DD
RR      RR  EEEEEEEEE  MM      MM  XX      XX  PP      PP  000000  RR      RR  TT      DD      DD
RR      RR  EEEEEEEEE  MM      MM  XX      XX  PP      PP  000000  RR      RR  TT      DD      DD
                                         ....
                                         ....
                                         ....
                                         ....
```

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

```

0000 1      .TITLE  REXPORT - PROVIDE THE ACP TRANSPORT MECHANISM
0000 2      .IDENT  'V04-000'
0000 3      :
0000 4      :*****
0000 5      :
0000 6      :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7      :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8      :*  ALL RIGHTS RESERVED.
0000 9      :
0000 10     :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11     :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12     :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13     :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14     :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15     :*  TRANSFERRED.
0000 16     :
0000 17     :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18     :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19     :*  CORPORATION.
0000 20     :
0000 21     :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22     :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23     :
0000 24     :
0000 25     :*****
0000 26     :
0000 27     :++
0000 28     : FACILITY:      REMOTE I/O ACP
0000 29     :
0000 30     : ABSTRACT:
0000 31     :   This module handles the mechanism for transporting
0000 32     :   ACP messages.
0000 33     :
0000 34     :   This version of the module uses:
0000 35     :   *****
0000 36     :   DECnet
0000 37     :   *****
0000 38     :   as its transport mechanism.
0000 39     :
0000 40     : ENVIRONMENT:
0000 41     :   MODE = KERNEL
0000 42     :--

```

```

0000 44
0000 45 : AUTHOR: SCOTT G. DAVIS, CREATION DATE: 11-JUL-79
0000 46 :
0000 47 : MODIFIED BY:
0000 48 :
0000 49 : V03-001 JLV0218 Jake VanNoy 7-Jan-1982
0000 50 : Add code to declare object type 42. TSA requires
0000 51 : this new object type. Store object type requested
0000 52 : in UCB.
0000 53 :
0000 54 : V02-005 SPF0050 Steve Forgey 31-Dec-1981
0000 55 : Store logical link number in RT UCB.
0000 56 :
0000 57 : V02-004 DJD2002 Darrell Duffy 2-May-1981
0000 58 : Fix to ignore link error mailbox messages.
0000 59 :
0000 60 : V02-003 DJD2001 Darrell Duffy 27-Feb-1981
0000 61 : Allow for max links at init time.
0000 62 :
0000 63 : V02-002 SGD2001 Scott G. Davis 23-Sep-1980
0000 64 : Remove multiple devices, ignore unknown messages

```

```

0000 66
0000 67 : INCLUDE FILES:
0000 68 :
0000 69      $AQBDEF
0000 70      $DIBDEF
0000 71      $MSGDEF
0000 72      $NFBDEF
0000 73      $REMDEF
0000 74      $UCBDEF
0000 75      $RTTUCBEXT
0000 76
0000 77 :
0000 78 : MACROS:
0000 79 :
0000 80
0000 81 :
0000 82 : EQUATED SYMBOLS:
0000 83 :
00000096 0000 84 MBX_MSG_LTH = 150      ; Size of a mailbox message
00000017 0000 85 RT_OBJ = 23      ; Object number for network cmd term
0000002A 0000 86 RT_OBJ2 = 42      ; Object number for TSA remote term
0000 87
0000 88 :
0000 89 : OWN STORAGE:
0000 90 :
00000000 0000 91      .PSECT REM_XP_PURE,NOWRT,NOEXE
0000 92
50 43 41 4D 45 52 00000008'010E0000' 0000 93 MBXDESC:      .ASCID /REMACP_MBX/      ; Logical name of mailbox
58 42 4D 5F 000E
3A 54 45 4E 5F 0000001A'010E0000' 0012 94 NETDESC:      .ASCID /_NET:/      ; Device name for channel assignment
00000005 001F 95 NFBDESC:      .LONG 5      ; NFB (FIB) descriptor
00000000 0023 96      .ADDRESS NFB
00000074 0027 97 DIBDESC:      .LONG DIB$K_LENGTH      ; Descriptor for device info bfr
000000A7 002B 98      .ADDRESS DIBBUF
002F 99
00000000 0000 100
00000000 0000 101      .PSECT REM_XP_IMPURE NOSHR,NOEXE,RD,WRT
0000 102
00000017 0001 103 NFB:      .BYTE NFB$C_DECLOBJ      ; Function code (DECLARE OBJ)
0005 104 NFB_OBJ:      .LONG RT_OBJ      ; Object number
0005 105 MBX_Q_IOSB:      ; I/O status block
0000000D 0005 106 NCBDESC:      ; Multiplex with MBX IOSB
000D 107      .BLKQ 1      ; Status of I/O completion
0000000F 000D 108 MBX_T_BUFFER:      ; Buffer for mailbox message
00000011 000F 109      .BLKW 1      ; Message identification
0011 110      .BLKW 1      ; Not used
000000A7 0011 111 MBX_T_DATA:      ; On connect initiates, et cetera
0000011B 00A7 112      .BLKB MBX_MSG_LTH      ; Leave room for message
113 DIBBUF: .BLKB DIB$K_LENGTH      ; Device information buffer

```

```

011B 115
00000000 116 .PSECT REM_XP_CODE,NOWRT
0000 117 :++
0000 118 :
0000 119 : REM$XPRT_START - initialize the transport mechanism
0000 120 :
0000 121 : FUNCTIONAL DESCRIPTION:
0000 122 :
0000 123 : This routine gets a channel for rejecting connection requests and
0000 124 : for declaring object types, to receive inbound connections.
0000 125 :
0000 126 : OUTPUTS:
0000 127 :
0000 128 : RO - LBC=>NG, LBS=>OK
0000 129 :
0000 130 :--
0000 131 :
0000 132 REM$XPRT_START::
0000 133 :
0000 134 : Create a mailbox for inbound connects, etc.
0000 135 :
0000 136 $CREMBX_S-
0000 137 CHAN= W^REM$GW_MBX_CHAN- ;
0000 138 MAXMSG= #MBX_MSG_LTH- ;
0000 139 BUFQUO= #<MBX_MSG_LTH*16>- ;
0000 140 LOGNAM= W^MBXDESC- ; Mailbox's logical name
0000 141 PROMSK= #0 ;
33 50 E9 0021 142 BLBC RO,10$ ; If LBC error
0024 143 :
0024 144 : Get a channel for rejects and mailbox info
0024 145 :
015A 30 0024 146 BSBW REM$XPRT_CHAN ; Get a channel to the network
2D 50 E9 0027 147 BLBC RO,10$ ; If LBC couldn't find one slot!
002A 148 :
002A 149 : Declare both object types for multiple inbound connects
002A 150 :
002A 151 5$:
002A 152 $QIOW_S CHAN= @W^REM$GL_CHANVEC[R11],-
002A 153 FUNC= S^#IOS$ACPCONTROL,-
002A 154 IOSB= W^MBX_Q_IOSB,-
002A 155 P1= W^NFBDESC
004C 156 BLBC RO,10$ ; If LBC NG
50 08 50 E9 0005'CF 3C 004F 157 MOVZWL W^MBX_Q_IOSB,R0 ; Get the I/O completion code
01 50 E8 0054 158 BLBS RO,20$ ; If LBS ok
0057 159 10$:
0057 160 RSB ; Done
0058 161 20$:
0001'CF 2A D1 0058 162 CMPL #RT_OBJ2,W^NFB_OBJ ; Done both?
07 13 005D 163 BEQL 25$ ; Branch if yes
0001'CF 2A 9A 005F 164 MOVZBL #RT_OBJ2,W^NFB_OBJ ; Set second object type
C4 11 0064 165 BRB 5$ ; Loop
0066 166 25$:
0000'DF4B 01 D0 0066 167 MOVL #1,@W^REM$GL_UCBVEC[R11] ; Stuff dummy value to show busy
006C 168 30$:
5A 01 90 006C 169 MOVB #REM$C_MBX_READ,R10 ; Set up for a mailbox read
FF8E' 31 006F 170 BRW REM$ENQUEUE ; Queue the request and return
0072 171

```

```

0072 172
0072 173
0072 174 ;+
0072 175 ; REM$MBX_READ - This routine puts a read out on the mailbox for
0072 176 ; process termination and inbound connect notifications
0072 177 ; -
0072 178 REM$MBX_READ:: ; Post read to mailbox
0072 179 $QIO_S CHAN = W^REM$GW MBX_CHAN,-
0072 180 FUNC = S^#IOS$ READVBLK,-
0072 181 ASTADR = B^MBX_AST,-
0072 182 IOSB = W^MBX_Q_IOSB,-
0072 183 P1 = W^MBX_T_BUFFER,-
0072 184 P2 = #MBX_MSG_LTH
5C 50 E9 009A 185 BLBC ; If LBC error
05 009D 186 RSB ; return
009E 187 ;++
009E 188 ;
009E 189 ; MBX_AST - This routine services process terminations
009E 190 ; and inbound connect notifications.
009E 191 ; -
009E 192 ;
009E 193 MBX_AST: .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
00A0 194 CMPW W^MBX_Q_IOSB,S^#SS$ _ABORT ; Was the I/O cancelled?
00A5 195 BEQL 50$ ; If EQL yes
00A7 196 CMPW W^MBX_Q_IOSB,#SS$ _CANCEL ; Try this code, too
00AE 197 BEQL 50$ ; If EQL yes
59 000D'CF 9E 00B0 198 MOVAB W^MBX_T_BUFFER,R9 ; Point to message buffer
56 89 3C 00B5 199 MOVZWL (R9)+,R6 ; Save mailbox message code
57 89 B0 00B8 200 MOVW (R9)+,R7 ; Get device dev unit number
50 69 9A 00BB 201 MOVZBL (R9),R0 ; Get length of device name
00BE 202 ; Counted ASCII string
59 50 D6 00BE 203 INCL R0
59 50 C0 00C0 204 ADDL R0,R9 ; Pass device name string
00C3 205 ;
00C3 206 ; Determine device index
00C3 207 ;
5B 0000'CF 9A 00C3 208 MOVZBL W^REM$GB_MAXLINKS,R11 ; Init count
00C8 209 10$:
0000'DF4B 57 B1 00C8 210 CMPW R7,@W^REM$GL_MBX_UNITS[R11] ; Match?
03 13 00CE 211 BEQL 20$ ; If EQL yes
F5 5B F5 00D0 212 SOBGR R11,10$ ; Loop
00D3 213 ;*****
00D3 214 ; FATAL - unit number not found
00D3 215 ;*****
00D3 216 ;
00D3 217 20$:
00D3 218 ;
00D3 219 ; Dispatch
00D3 220 ;
07 10 00D3 221 BSBB 100$ ; Allow for easy return
00D5 222 ; by action routines
5A 01 90 00D5 223 MOVB #REM$C MBX_READ,R10 ; Set up for a mailbox read
FF25' 30 00D8 224 BSBW REM$ENQUEUE ; Queue the request
04 00DB 225 50$: RET ; Done
00DC 226 ;
00DC 227 ; Dispatch to action routine matching logical link event
00DC 228 ;

```

```

00DC 229 ; R9 points to mailbox data
00DC 230 ; R11 contains device index
00DC 231 ;
00DC 232 100$: ; Entry point
00DC 233 $DISPATCH R6,- ; Case on event code
00DC 234
00DC 235 <-; event action
00DC 236
00DC 237 <MSG$_CONNECT, CONN_REQ>, -; Connect request received
00DC 238 <MSG$_ABORT, LINK_BROKEN>, -; Link broken
00DC 239 <MSG$_DISCON, LINK_BROKEN>, -; Link broken
00DC 240 <MSG$_EXIT, LINK_BROKEN>, -; Link broken
00DC 241 <MSG$_PATHLOST, LINK_BROKEN>, -; Link broken
00DC 242 <MSG$_PROTOCOL, LINK_BROKEN>, -; Link broken
00DC 243 <MSG$_THIRDPARTY, LINK_BROKEN>, -; Link broken
00DC 244 <MSG$_NETSHUT, NET_SHUTDOWN>, -; Network shutting down
00DC 245 >
05 00F8 246 RSB ; Ignore other messages
00F9 247 MBX_FAILURE:
00F9 248 BUG_CHECK ACPMBFAIL,FATAL

```

```

00FD 250
00FD 251 :++
00FD 252 :
00FD 253 : CONFIRM_AST - This is the AST routine for connect accepts.
00FD 254 : Issue a read over the link.
00FD 255 :
00FD 256 :
00FD 257 :--
00FD 258 :
50 5B 04 AC OFFC 00FD 259 CONFIRM_AST: .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
0000'DF4B 7D 00FF 260 .MOVL 4(AP),R11 ; Get the device index
04 50 E8 0103 261 .MOVQ @W^REM$GL_RIOSBVECC[R11],R0 ; Get the IOSB
0109 262 .BLBS R0,5$ ; If LBS I/O was OK
010C 263 :
010C 264 : I/O failed
010C 265 :
FEF1' 30 010C 266 .BSBW REM$KILL_UCB ; Clean up the channel
04 010F 267 .RET ; Done
0110 268 5$:
55 0000'DF4B D0 0110 269 .MOVL @W^REM$GL_UCBVECC[R11],R5 ; Get the UCB address
64 A5 10 8A 0116 270 .BICB #UCB$M_ONLINE,UCB$W_STS(R5) ; Waiting for CONFIGURE
FEE3' 30 011A 271 .BSBW REM$SEND_CONFIG ; Send one, too
5A 02 90 011D 272 .MOVB #REM$C_LNK_READ,R10 ; Set up for a link read
FEDD' 30 0120 273 .BSBW REM$ENQUEUE ; Queue the work element
04 0123 274 .RET ; Dismiss the AST
0124 275 :
0124 276 :
0124 277 :
0124 278 :++
0124 279 :
0124 280 : CONN_REQ - Connection request received - figure out who and why.
0124 281 : Allocate a channel and a UCB, set up the data bases,
0124 282 : and confirm the link, if everything is copasetic,
0124 283 : else reject the logical link.
0124 284 :
0124 285 : INPUTS:
0124 286 :
0124 287 : R9 - points at counted NCB
0124 288 :
0124 289 :--
0124 290 :
5A 0005'CF 9E 0124 291 CONN_REQ:
8A 89 9A 0124 292 .MOVAB W^NCBDESC,R10 ; Point to descriptor
6A 59 D0 0129 293 .MOVZBL (R9)+,(R10)+ ; Store length
5A 04 C2 012C 294 .MOVL R9,(R10) ; Pointer to data
59 00' D0 012F 295 .SUBL2 #4,R10 ; Point to descriptor
58 C5 AF 9E 0132 296 .MOVL S^#IOS$_ACCESS,R9 ; Set the initial function code
0135 297 .MOVAB B^CONFIRM_AST,R8 ; For confirm, take an AST
0139 298 :
0139 299 : Try to set up a database for this channel
0139 300 :
FEC4' 30 0139 301 .BSBW REM$SET_UP_IN ; Do it
13 50 E8 013C 302 .BLBS R0,200$ ; If LBS OK
013F 303 :
013F 304 : Can't take this link - reject it
013F 305 :
013F 306 100$:

```

```

5B 0000'CF 9A 013F 307 MOVZBL W^REM$GB_MAXLINKS,R11 ; Set up index of reject 'device'
57 0000'DF B0 0144 308 MOVW @W^REM$GL_REJ_CHAN,R7 ; Get reject channel
59 0000'8F A8 0149 309 BISW #IOSM_ABORT,R9 ; Fix up the function code
      58 D4 014E 310 CLRL R8 ; No AST
      1D 11 0150 311 BRB 225$
55 0000'DF4B D0 0152 312 200$: MOVL @W^REM$GL_UCBVEC[R11],R5 ; Get UCB address
      00D6 C5 B4 0158 313 CLRW UCB$W_RTT_LINK(R5) ; Initialize cell
04 BA 6A 2F 3A 015C 314 LOCC #^A'/^',(R10),@4(R10) ; Find separator in NCB
      0C 13 0161 315 BEQL 225$ ; If EQL, separator not found
00D6 C5 01 A1 B0 0163 316 MOVW 1(R1),UCB$W_RTT_LINK(R5) ; Store logical link number in RT UCB
00D8 C5 15 A1 90 0169 317 MOVW 21(R1),UCB$B_RTT_OBJ(R5) ; Store object type requested
      016F 318 225$: MOVB
55 0000'DF4B 7E 016F 319 225$: MOVAQ @W^REM$GL_RIOSBVEC[R11],R5 ; Get the IOSB address
      56 D4 0175 320 CLRL R6 ; No P1
      FE86' 30 0177 321 BSBW REM$DO_QIO ; Perform the requisite I/O
      03 50 E8 017A 322 BLBS R0,250$ ; If LBS OK
      FE80' 30 017D 323 BSBW REM$KILL_UCB ; The I/O failed - clean up
      0180 324 250$: RSB
      05 0180 325 RSB ; Done
      0181 326
      0181 327
      0181 328
      0181 329
      0181 330 :++
      0181 331 :
      0181 332 : REM$XPORT_CHAN - Get a channel to the network and set up various vectors
      0181 333 :
      0181 334 : INPUTS:
      0181 335 :
      0181 336 : None
      0181 337 :
      0181 338 : OUTPUTS:
      0181 339 :
      0181 340 : R0 - lbs => things are OK; lbc => NG
      0181 341 : R11 - device index
      0181 342 : Channel no. and unit stored
      0181 343 :
      0181 344 :--
      0181 345 :
      0181 346 REM$XPORT_CHAN::
5B 0000'CF 9A 0181 347 MOVZBL W^REM$GB_MAXLINKS,R11 ; Init index
      0000'DF4B D5 0186 348 10$: TSTL @W^REM$GL_UCBVEC[R11] ; Free slot?
      06 13 018B 349 BEQL 20$ ; If EQL yes
      F6 5B F5 018D 350 SOBGTR R11,10$ ; Loop
      50 D4 0190 351 CLRL R0 ; Return error
      05 0192 352 RSB ; Done
      0193 353 20$:
      0193 354 $ASSIGN_S DEVNAM= W^NETDESC- ; "NET:" refers to NETACP
      0193 355 -CHAN= @W^REM$GL_CHANVEC[R11]- ; Store channel #
      0193 356 MBXNAM= W^MBXDESC ; Specify associated mailbox
      1F 50 E9 01A9 357 BLBC R0,100$ ; If LBC assignment failed
      01AC 358 $GETCHN_S CHAN= @W^REM$GL_CHANVEC[R11],- ; Discover the unit no. of ne
      01AC 359 PRIBUF= W^DIBDESC ; Address of char buf descriptor
      00B3'CF B0 01C3 360 MOVW W^DIBBUF+DIB$W_UNIT,- ; Store associated unit
      0000'DF4B 01C7 361 @W^REM$GL_MBX_UNITS[R11]
      01CB 362 100$:
      01CB 363

```

```

05 01CB 364 RSB ; That is all
    01CC 365
    01CC 366
    01CC 367 :++
    01CC 368 :
    01CC 369 : LINK_BROKEN - The logical link was broken
    01CC 370 :
    01CC 371 : INPUTS:
    01CC 372 :
    01CC 373 : R11 - device index
    01CC 374 :
    01CC 375 :--
    01CC 376
    01CC 377 LINK_BROKEN:
    01CC 378 :
    01CC 379 : We ignore mailbox error codes which signal links are breaking because
    01CC 380 : it is not feasible or necessary to synchronize actions taken here with those
    01CC 381 : taken in rttdriver and elsewhere. There is always io outstanding to the
    01CC 382 : link, either here during protocol init or in rttdriver. This io will
    01CC 383 : complete with errors when the link breaks and appropriate action will be
    01CC 384 : taken.
    01CC 385 :
05 01CC 386 RSB
    01CD 387
    01CD 388 :++
    01CD 389 :
    01CD 390 : NET_SHUTDOWN - Network is shutting down.. Go away, since there is no
    01CD 391 : way to keep "declared name" or "declared obj" info around.
    01CD 392 :
    01CD 393 : Side effects:
    01CD 394 :
    01CD 395 : Mount count is cleared - equivalent to running STOPREM
    01CD 396 :
    01CD 397 :--
    01CD 398
    01CD 399 NET_SHUTDOWN:
52 0000'CF D0 01CD 400 MOVL W*REM$GL Q HEAD,R2 ; Get the AQB address
    0B A2 94 01D2 401 CLR B AQB$B_MNTCNT(R2) ; Mark for dismount
    05 01D5 402 RSB ; Done

```

```

01D6 404
01D6 405 :++
01D6 406 : FUNCTIONAL DESCRIPTION:
01D6 407 :
01D6 408 : BIN_TO_ASC - THIS ROUTINE CONVERTS A BINARY NUMBER TO AN
01D6 409 : ASCII STRING WITH ZERO SUPPRESSION.
01D6 410 : CODE IS DUE TO BILL BROWN (JBC_BIN2ASC).
01D6 411 :
01D6 412 : CALLING SEQUENCE:
01D6 413 :
01D6 414 : BSB BIN_TO_ASC
01D6 415 :
01D6 416 : INPUT PARAMETERS:
01D6 417 :
01D6 418 : R0 = NUMBER TO CONVERT
01D6 419 : R1 = SCRATCH
01D6 420 : R3 = ADDRESS OF PLACE TO STORE FIRST BYTE
01D6 421 :
01D6 422 : OUTPUT PARAMETERS:
01D6 423 :
01D6 424 : NUMBER IS CONVERTED AND STORED
01D6 425 : R3 IS INCREMENTED PAST LAST BYTE STORED
01D6 426 :
01D6 427 : IMPLICIT OUTPUTS:
01D6 428 :
01D6 429 : R0 AND R1 ARE RETURNED ZERO
01D6 430 :
01D6 431 :--
01D6 432 :
01D6 433 BIN_TO_ASC:
01D6 434 CLRL R1
7E 50 50 0A D4 01D6 435 20$: EDIV #10,R0,R0,-(SP)
6E 30 C0 01D8 436 ADDL #^A/0/, (SP)
50 D5 01E0 437 TSTL R0
02 13 01E2 438 BEQL 30$
F2 10 01E4 439 BSBB 20$
83 8E F6 01E6 440 30$: CVTLB (SP)+,(R3)+
05 05 01E9 441 RSB
01EA 442
01EA 443 .END

```

```

: ENTER WITH JSB/BSB
: ZERO HI 1/2 OF QUAD WORD
: DIVIDE BY DECIMAL RADIX
: CONVERT TO ASCII
: ANYTHING LEFT TO CONVERT
: BR IF NO MORE SIGNIFICANT DIGITS
: RECURSE THRU ASCII CONVERT
: INSERT ASCII UNIT NUMBER
: UNWIND RECURSIVE LOOP

```

REXPORT
Symbol table

H 12
- PROVIDE THE ACP TRANSPORT MECHANISM

16-SEP-1984 02:13:14 VAX/VMS Macro V04-00
5-SEP-1984 02:54:18 [REM.SRC]REXPORTD.MAR;1

Page 11
(11)

```

$ST1          = 00000000
AQB$B_MNTCNT  = 0000000B
BIN_TO_ASC    000001D6 R    04
BUG$ ACPMBFAIL ***** X    04
CONFIRM_AST   000000FD R    04
CONN_REQ      00000124 R    04
DIB$K_LENGTH  = 00000074
DIB$W_UNIT    = 0000000C
DIBBUF        000000A7 R    03
DIBDESC       00000027 R    02
IOSM_ABORT    ***** X    04
IOS_ACCESS    ***** X    04
IOS_ACPCONTROL ***** X    04
IOS_READVBLK  ***** X    04
LINK_BROKEN   000001CC R    04
MBXDESC       00000000 R    02
MBX_AST       0000009E R    04
MBX_FAILURE   000000F9 R    04
MBX_MSG_LTH   = 00000096
MBX_Q_IOSB    00000005 R    03
MBX_T_BUFFER  0000000D R    03
MBX_T_DATA    00000011 R    03
MSG$ ABORT     = 00000030
MSG$ CONNECT  = 00000032
MSG$ DISCON   = 00000033
MSG$ EXIT     = 00000034
MSG$ NETSHUT  = 0000003B
MSG$ PATHLOST = 00000036
MSG$ PROTOCOL = 00000037
MSG$ THIRDPARTY = 00000039
NCBDESC       00000005 R    03
NETDESC       00000012 R    02
NET_SHUTDOWN  000001CD R    04
NFB           00000000 R    03
NFB$C DECLOBJ = 00000016
NFBDESC       0000001F R    02
NFB_OBJ       00000001 R    03
REMSC_CURECO  = 00000001
REMSC_CURVRS  = 00000001
REMSC_LNK_READ = 00000002
REMSC_MAXDEVS = 0000000A
REMSC_MAXLINKS = 00000010
REMSC_MAXUNITS = 00000010
REMSC_MBX_READ = 00000001
REMSC_ST_ATTRIB = 00000002
REMSC_ST_CONFIG = 00000001
REMSDQ_QIO    ***** X    04
REMSDQ_QUEUE  ***** X    04
REMSGB_MAXLINKS ***** X    04
REMSGL_CHANVEC ***** X    04
REMSGL_MBX_UNITS ***** X    04
REMSGL_Q_HEAD ***** X    04
REMSGL_REJ_CHAN ***** X    04
REMSGL_RIOSBVEC ***** X    04
REMSGL_UCBVEC ***** X    04
REMSGW_MBX_CHAN ***** X    04
REMSKILL_UCB ***** X    04

```

```

REMSMBX_READ  00000072 RG    04
REMSSEND_CONFIG ***** X    04
REMSSET_OP_IN ***** X    04
REMSXPORT_CHAN 00000181 RG    04
REMSXPORT_START 00000000 RG    04
RT_OBJ        = 00000017
RT_OBJ2       = 0000002A
SS$ ABORT      ***** X    04
SS$ CANCEL     ***** X    04
SYS$ASSIGN     ***** GX    04
SYS$CREMBX     ***** GX    04
SYS$GETCHN     ***** GX    04
SYS$QIO        ***** GX    04
SYS$QIOW       ***** GX    04
UCB$B_RTT_OBJ  = 000000D8
UCB$K_RTT_LEN  = 00000138
UCB$K_RTT_LENGTH = 00000138
UCB$L_DEVDEPND2 = 00000048
UCB$L_RTT_CTRLC = 00000094
UCB$L_RTT_CTRLY = 00000090
UCB$L_RTT_DEVDEPND2 = 00000048
UCB$L_TL_BANDQUE = 0000009C
UCB$L_TL_CTRLC  = 00000094
UCB$L_TL_CTRLY  = 00000090
UCB$L_TL_OUTBAND = 00000098
UCB$M_ONLINE   = 00000010
UCB$W_RTT_LINK = 000000D6
UCB$W_STS      = 00000064

```

EX
Mo
--
RM
IN
SY
CJ
MA
SY
RM

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
REM_XP_PURE	0000002F (47.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC BYTE
REM_XP_IMPURE	0000011B (283.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
REM_XP_CODE	000001EA (490.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.08	00:00:00.33
Command processing	107	00:00:00.63	00:00:02.35
Pass 1	422	00:00:16.31	00:00:34.77
Symbol table sort	0	00:00:02.40	00:00:04.37
Pass 2	92	00:00:02.68	00:00:05.19
Symbol table output	11	00:00:00.12	00:00:00.58
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	666	00:00:22.25	00:00:47.62

The working set limit was 1500 pages.
82688 bytes (162 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1601 non-local and 24 local symbols.
443 source lines were read in Pass 1, producing 20 object records in Pass 2.
42 pages of virtual memory were used to define 40 macros.

! Macro library statistics !

Macro library name	Macros defined
-\$255\$DUA28:[REM.OBJ]REM.MLB;1	3
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	14
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	14
TOTALS (all libraries)	31

1839 GETS were required to define 31 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:REXPORTD/OBJ=OBJ\$:REXPORTD MSRC\$:REXPORTD/UPDATE=(ENH\$:REXPORTD)+EXECML\$/LIB+LIB\$:REM/LIB

0312 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

