


```

SSSSSSSS YY YY SSSSSSS BBBB BBBB RRRRRRR KK KK TTTTTTTTT HH HH RRRRRRR
SSSSSSSS YY YY SSSSSSS BBBB BBBB RRRRRRR KK KK TTTTTTTTT HH HH RRRRRRR
SS YY YY SS SS SS BB BB RR RR KK KK TT TT HH HH RR RR
SS YY YY SS SS SS BB BB RR RR KK KK TT TT HH HH RR RR
SSSSSS YY YY SSSSSS SSSSSS BBBB BBBB RRRRRRR KKKKKK TT TT HHHHHHHHH RRRRRRR
SSSSSS YY YY SSSSSS SSSSSS BBBB BBBB RRRRRRR KKKKKK TT TT HHHHHHHHH RRRRRRR
SS YY YY SS SS SS BB BB RR RR KK KK TT TT HH HH RR RR
SS YY YY SS SS SS BB BB RR RR KK KK TT TT HH HH RR RR
SSSSSS YY YY SSSSSSS BBBB BBBB RR RR KK KK TT TT HH HH RR RR
SSSSSS YY YY SSSSSSS BBBB BBBB RR RR KK KK TT TT HH HH RR RR

```

```

LL LL SSSSSSS
LL LL SSSSSSS
LL II SS
LL II SS
LL II SS
LL II SSSSSS
LL II SSSSSS
LL II SS
LL II SS
LL II SS
LLLLLLLLLL IIIIII SSSSSSS
LLLLLLLLLL IIIIII SSSSSSS

```

(1)	115
(3)	220
(4)	465
(5)	644
(6)	768
(7)	1006
(8)	1082
(9)	1149
(10)	1236

DECLARATIONS
EXESBRKTHRU - Break though write
DO_WRITE - Queue a single write request
GET_SENDTO - Handle SENDTO and SENDTYPE inputs
GET_NEXT_TERMINAL - return next terminal
FIND_NEXT_TERM - Search I/O database
QIO_DONE - process qio completion
CHECK_COMPLETE - Check completion criterion
QIO_TIMEOUT - process qio timeout

```
0000 1 .TITLE SYSBRKTHR - Write breakthru to terminals
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 ++
0000 29
0000 30 FACILITY:
0000 31
0000 32 SYS
0000 33
0000 34 INCLUDES:
0000 35 $BRKTHRU system service
0000 36 $BRDCST system service
0000 37
0000 38 ABSTRACT:
0000 39
0000 40 Write breakthru message to specified terminals and mailboxes.
0000 41
0000 42 ENVIRONMENT:
0000 43
0000 44 Kernel Mode. IPL 0 and 2.
0000 45
0000 46 --
0000 47
0000 48 AUTHOR: Jake VanNoy, CREATION DATE: 3-Feb-1983
0000 49
0000 50 MODIFIED BY:
0000 51
0000 52 V03-011 JLV0392 Jake VanNoy 26-JUL-1984
0000 53 Make check for TRM and SPL at HAVE_UCB.
0000 54 Do not write message to mailbox if class disabled.
0000 55
0000 56 V03-010 JLV0347 Jake VanNoy 8-APR-1984
0000 57 Skip terminal if NET is set. Fix problem in
```

```
0000 58 : check for broadcast to same username.
0000 59 : Copy DEVNAME to SENDNAME so that cluster broadcast
0000 60 : to device will work. Change MOVC of device name
0000 61 : fields to MOVQ's.
0000 62 :
0000 63 : V03-009 JLV0339 Jake VanNoy 9-MAR-1984
0000 64 : Skip terminal if PASSALL is set. Fix mailbox message
0000 65 : to have just DDC part of device name. Force timeout
0000 66 : of a cluster breakthru request to 15 seconds on all
0000 67 : nodes except local. Fix bug that used BRKSL_FLAGS as
0000 68 : scratch.
0000 69 :
0000 70 : V03-008 ACG0385 Andrew C. Goldstein, 28-Dec-1983 15:27
0000 71 : Change UAF$$_USERNAME use to JIB$$_USERNAME, due to
0000 72 : pending UAF format changes
0000 73 :
0000 74 : V03-007 JLV0308 Jake VanNoy 22-SEP-1983
0000 75 : Complete work started in JLV0307. Fix check against
0000 76 : username in GET_SENDTO. Change parameter in call
0000 77 : to IOC$CVT_DEVNAM, since the interface to that routine
0000 78 : has changed.
0000 79 :
0000 80 : V03-006 JLV0307 Jake VanNoy 7-SEP-1983
0000 81 : Fix enhanced privilege bug. Wait until after cluster
0000 82 : broadcast to deallocate BRK. Fix bug in defaulting of
0000 83 : carriage control in $BRDCST. Add use of EXE$$SIGTORET
0000 84 : in $BRDCST.
0000 85 :
0000 86 : V03-005 JLV0302 Jake VanNoy 22-AUG-1983
0000 87 : Add MOVCS to zero entire BRK structure up to where text
0000 88 : is placed. This allowed removing separate CLR$ instructions
0000 89 : in initialization. Save register around MOVC in GET_SENDTO.
0000 90 : Change exit path for SSS_NOOPER error code.
0000 91 :
0000 92 : V03-004 JLV0300 Jake VanNoy 30-JUL-1983
0000 93 : Add OPER priv checks. Allow $BRKTHRU to same username
0000 94 : without priv. Initialize mailbox prefix code. Remove
0000 95 : BRK$ symbols from here and move them to LIB. This
0000 96 : allows cluster broadcast code to use BRK structure.
0000 97 : Add IOSM_CANCTRL0 to QIO. Make use of IOC$CVT_DEVNAM.
0000 98 :
0000 99 : V03-003 LJK0213 Lawrence J. Kenah 23-Jun-1983
0000 100 : Unlock data base before calling GET_NEXT_TERMINAL to make
0000 101 : sure that $GETJPI is not called at IPL 2.
0000 102 :
0000 103 : V03-002 JLV0269 Jake VanNoy 27-MAY-1983
0000 104 : Fix bugs in SET_PRIV routine. Add code to use REQID.
0000 105 : Add code to call EXE$CSP_BRKTHRU, the cluster broadcast
0000 106 : routine.
0000 107 :
0000 108 : V03-001 JLV0245 Jake VanNoy 29-APR-1983
0000 109 : First pass cleanup. Include code for EXE$BRDCST here,
0000 110 : this obsoletes the old SYSBRDCST module.
0000 111 :
0000 112 : **
0000 113 :
0000 114 :
```

```

0000 115 .SBTTL DECLARATIONS
0000 116 :
0000 117 : INCLUDE FILES:
0000 118 :
0000 119 SBRKDEF : Define BRKTHR interface symbols
0000 120 SBRKTDEF : Define BRK block
0000 121 SCCBDEF : Define channel control block
0000 122 SddbDEF : Define device data block
0000 123 SDEVDEF : Define device symbols
0000 124 SDVIDEF : Define GETDVI symbols
0000 125 $IODEF : Define I/O request symbols
0000 126 $IPLDEF : Define IPL fields
0000 127 $JIBDEF : Define Job Information Block
0000 128 $JPIDEF : Define GETJPI symbols
0000 129 $PCBDEF : Define process control block
0000 130 $PHDDEF : Define process header
0000 131 $PRVDEF : Define privilege names
0000 132 $PSLDEF : Define PSL fields
0000 133 $SSDEF : Define status codes
0000 134 $TTDEF : Define tt devdepend symbols
0000 135 $TT2DEF : Define tt devdepend2 symbols
0000 136 $TTYUCBDEF : terminal ucb extensions
0000 137 $UAFDEF : Define user authorization symbols
0000 138 $UCBDEF : Define UCB
0000 139 :
0000 140 : MACROS:
0000 141 :
0000 142 :
0000 143 :
0000 144 : EQUATED SYMBOLS:
0000 145 :
0000 146 :
00000004 0000 147 EFN = 4
00000008 0000 148 MSGBUF = 8
0000000C 0000 149 SENDTO = 12
00000010 0000 150 SENDTYPE = 16
00000014 0000 151 IOSB = 20
00000018 0000 152 CARCON = 24
0000001C 0000 153 FLAGS = 28
00000020 0000 154 REQID = 32
00000024 0000 155 TIMEOUT = 36
00000028 0000 156 ASTADR = 40
0000002C 0000 157 ASTPRM = 44
0000 158 :
0000001F 0000 159 BRK_C_JPIEFN = 31 : system efn
0000001F 0000 160 BRK_C_TIMEFN = 31
0000001F 0000 161 BRK_C_QIOEFN = 31
0000001F 0000 162 BRK_C_DVIEFN = 31
0000001F 0000 163 BRK_C_BRDCSTEFN = 31
00000004 0000 164 BRK_C_MINTIME = 4 : minimum time in seconds
00000004 0000 165 BRK_C_SIMULCAST = 4 : simultaneous QIO's
00000018 0000 166 BRK_C_MAXLINES = 24 : maximum number of lines allowed to clear in screen write
0000000F 0000 167 BRK_C_CLUTIMEOUT = 15 : forced timeout for cluster broadcast
0000 168 :
20000000 0000 169 PRVSM_BYPASS = 1@PRV$V_BYPASS : define mask
80000000 0000 170 PRVSM_SHARE = 1@PRV$V_SHARE : define mask
0000 171 :

```

```
0000 172 ; following assumes for MOVQ's of name buffer's
0000 173
0000 174 ASSUME DDB$$_NAME      EQ 16
0000 175 ASSUME BRK$$_DEVNAME  EQ 16
0000 176 ASSUME BRK$$_SENDNAME EQ 16
0000 177 ASSUME BRK$$_TRMNAME  EQ 16
0000 178
```

```

0000 180 :
0000 181 : Local storage offsets for temporary stack allocation
0000 182 :
0000 183 :
0000 184 :
0000 185 : getjpi stack items
0000 186 :
0000 187 $DEFINI STK
0000 188
0000 189 $DEF STKSW_USERSIZ .BLKW
0002 190 $DEF STKSW_USERJPI .BLKW
0004 191 $DEF STKSL_USERNAME .BLKL
0008 192 $DEF STK$_USERLENR .BLKL
000C 193
000C 194 $DEF STKSW_TERMSIZ .BLKW
000E 195 $DEF STKSW_TERMJPI .BLKW
0010 196 $DEF STKSL_TERMNAME .BLKL
0014 197 $DEF STKSL_TERMLENR .BLKL
0018 198
0018 199 $DEF STKSL_ENDLIST .BLKL
001C 200
001C 201 $DEF STKSW_USERLEN .BLKW
001E 202 $DEF STKST_USERNAME .BLKB JIBSS_USERNAME
002A 203 $DEF STKSW_TERMLEN .BLKW
002C 204
002C 205 $DEF STKSC_LEN
002C 206
002C 207 $DEFEND STK
0000 208 :
0000 209 : OWN STORAGE:
0000 210 :
0000 211
00000000 212 .PSECT Y$EXEPAGED
0000 213
4B 30 5B 1B 41 31 5B 1B 0000 214 erase_pat: .ascii /E1A[OK/
0008 215 assume .-erase_pat EQ 8 ; so quadword access can be done
0008 216
55 21 5B 1B 37 1B 00000010'010E0000' 0008 217 screen_ctrstr: .ascid /7E!UB;1H[K!AD!AD8/
41 21 44 41 21 4B 5B 1B 48 31 3B 42 0016
38 1B 44 0022
0025 218

```



```

0025 220 .SBTTL EXESBRKTHRU - Break though write
0025 221
0025 222 :++
0025 223 :
0025 224 : FUNCTIONAL DESCRIPTION:
0025 225 :
0025 226 :
0025 227 : CALLING SEQUENCE:
0025 228 : NONE
0025 229 :
0025 230 : INPUT PARAMETERS:
0025 231 :
0025 232 : R4 - PCB
0025 233 : AP - argument list
0025 234 :
0025 235 : IMPLICIT INPUTS:
0025 236 : NONE
0025 237 :
0025 238 : OUTPUT PARAMETERS:
0025 239 : NONE
0025 240 :
0025 241 : IMPLICIT OUTPUTS:
0025 242 : NONE
0025 243 :
0025 244 : COMPLETION CODES:
0025 245 : NONE
0025 246 :
0025 247 : SIDE EFFECTS:
0025 248 : NONE
0025 249 :
0025 250 :--
0025 251
OFFC 0025 252 .ENTRY EXESBRKTHRU,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
0027 253
0027 254 :
0027 255 : Check parameters and do initialization needed
0027 256 :
56 D4 0027 257 CLRL R6 ; no buffer yet
0029 258 :
0029 259 : Clear Event Flag
0029 260 :
53 04 AC 9A 0029 261 MOVZBL EFN(AP),R3 ; Fetch EFN
00000000'EF 16 002D 262 JSB SCH$CLREF ; Clear
4F 50 E9 0033 263 BLBC R0,20$ ; Exit on error
0036 264 :
0036 265 : Verify IOSB and clear it
0036 266 :
5B 14 AC D0 0036 267 MOVL IOSB(AP),R11 ; Get address of IOSB
0B 13 003A 268 BEQL 10$ ; Branch if none
009F 31 003C 269 IFWRT #8,(R11),5$ ; Branch if ok
6B 7C 0042 270 BRW ACCVIO_EXIT ; Error if not writeable
51 08 AC D0 0045 271 CLRQ (R11) ; Clear
00000000'GF 16 0047 272 5$:
31 50 E9 0047 273 10$:
0047 273 MOVL MSGBUF(AP),R1 ; Message buffer descriptor
0048 274 JSB G^EXESPROBER_DSC ; Probe descriptor
0051 275 BLBC R0,20$ ; branch if error
0054 276 :

```

```

0054 277 ; R1 and R2 have length and address, calculate size of buffer
0054 278 ; needed for storage.
0054 279
51 51 3C 0054 280 MOVZWL R1,R1 ; clear top word
59 51 7D 0057 281 MOVQ R1,R9 ; save both
53 8E 8F 9A 005A 282 MOVZBL #BRK$C_LENGTH,R3 ; Size of basic block
53 51 C0 005E 283 ADDL R1,R3 ; For normal data
58 000000D0 8F C1 0061 284 ADDL3 #16+(8*BRK_C_MAXLINES),R1,R8 ; screen overhead and message
53 58 C0 0069 285 ADDL R8,R3 ; For screen data
53 03 C0 006C 286 ADDL #3,R3 ; round of to longword by adding and...
53 03 CA 006F 287 BICL #3,R3 ; clearing bits
57 53 D0 0072 288 MOVL R3,R7 ; Save this length
50 04 C5 0075 289 MULL3 #BRK_C_SIMULCAST,- ;
53 50 C0 0077 290 #BRK2$C_LENGTH,R0 ; Size of context area
0079 291 ADDL R0,R3 ; add to length
007C 292
007C 293 ; Compute pages and allocate region
007C 294
51 53 D0 007C 295 MOVL R3,R1 ; Number of bytes
00000000'GF 16 007F 296 JSB G^EXESALOP1IMAG ; Allocate memory
5F 50 E9 0085 297 20$: BLBC R0,ERROR_EXIT ; exit on error
0088 298
0088 299 ; Copy remaining paramters into allocated region
0088 300
56 52 D0 0088 301 MOVL R2,R6 ; Copy Address of block
00 6E 00 2C 008D 303 PUSHR #*M<R1,R2,R3,R4> ; Save
62 008E 8F 0091 304 MOVCS #0,(SP),#0,- ;
0095 305 POPR #*M<R1,R2,R3,R4> ; Restore
0097 306
08 A6 51 B0 0097 307 MOVW R1,BRK$W_SIZE(R6) ; And size
60 A6 6647 9E 009B 308 MOVAB (R6)[R7],BRK$Q_QIOCTX(R6) ; Qio context start address
68 A6 58 D0 00A0 309 MOVL R8,BRK$S_SCRMSGLEN(R6) ; init
1C A6 54 D0 00A4 310 MOVL R4,BRK$S_PCB(R6) ; Save PCB
20 A6 5B D0 00A8 311 MOVL R11,BRK$S_IOSB(R6) ; Set address
00AC 312
00AC 313 ; Copy main message buffer
00AC 314
008C C6 59 B0 00AC 315 MOVW R9,BRK$W_MSCLLEN(R6) ; Save length
6A 59 28 00B1 316 MOVCS R9,(R10),- ;
008E C6 00B4 317 BRK$T_MSGBUF(R6) ; Copy message buffer
6C A6 53 D0 00B7 318 MOVL R3,BRK$S_SCRMSG(R6) ; next byte is where screen message starts
00BB 319
00BB 320 ; Copy send type and "send to:" string (if required)
00BB 321
027B 30 00BB 322 BSBW GET_SENDTO ; handle SENDTO, SENDTYPE
26 50 E9 00BE 323 BLBC R0,ERROR_EXIT ; check status
00C1 324
00C1 325 ; Set up time quadword if timeout requested
00C1 326
50 24 AC D0 00C1 327 MOVL TIMEOUT(AP),R0 ; Timeout value
12 13 00C5 328 BEQL 240$ ; branch if none specified
50 04 D1 00C7 329 CML #BRK_C_MINTIME,R0 ; Compare to minimum number of seconds
13 14 00CA 330 BGTR BADPARAM_EXIT ; Exit if too small
50 50 CE 00CC 331 MNEGL R0,R0 ; Get negative value
00 50 00989680 8F 7A 00CF 332 EMUL #10*1000*1000,R0,#0,- ; Times ten million ticks per second
2C A6 00D7 333

```

```

0F B0 00D9 334 240$: MOVW #BRK_C CLUTIMEOUT,-
4E A6 00DB 335 BRK$Q SECONDS(R6) ; set default timeout for cluster
10 11 00DD 336 BRB ALL_OR ; And continue
00DF 337
00DF 338
00DF 339 ; An error has occured in initial processing...
00DF 340
00DF 341 BADPARAM_EXIT:
50 14 3C 00DF 342 MOVZWL #SS$ BADPARAM,R0 ; set status
03 11 00E2 343 BRB ERROR_EXIT ; exit
00E4 344 ACCVIO_EXIT:
50 0C 3C 00E4 345 MOVZWL #SS$ ACCVIO,R0 ; Set error
00E7 346 ERROR_EXIT:
56 D5 00E7 347 TSTL R6 ; Buffer to delete?
03 13 00E9 348 BEQL 10$ ; Branch if not
056E 30 00EB 349 BSBW RETURN_MEMORY ; return memory
00EE 350 10$:
04 00EE 351 RET ; exit
00EF 352
00EF 353
00EF 354 ; Copy remaining parameters...
00EF 355
00EF 356 ALL_OK:
50 A0000000 8F D0 00EF 357 MOVL #<PRVSM BYPASS!PRVSM_SHARE>,R0 ; privileges required
00F6 358 ASSUME PHD$Q_PRIVMSK EQ 0 ; for indirection
66 54 1C A6 D0 00F6 359 MOVL BRK$Q_PCB(R6),R4 ; Set PCB address
50 6C B4 CB 00FA 360 BICL3 @PCB$Q_PHD(R4),R0,BRK$Q_PRIVS(R6) ; Clear those already set
00FF 361
00FF 362 ASSUME BRK$W_EFN+2 EQ BRK$B_STS ; assumes so next instruction
00FF 363 ASSUME BRK$W_EFN+3 EQ BRK$B_PRVMODE ; can set efn and zero sts and prvm0
64 A6 04 AC 3C 00FF 364 MOVZWL EFN(AP),BRK$W_EFN(R6) ; Copy event flag number
50 20 AC D0 0104 365 MOVL REQID(AP),R0 ; Requestor ID
50 3F D1 0108 366 CML #63,R0 ; Check legal (0-63 legal)
50 D2 1F 010B 367 BLSSU BADPARAM_EXIT ; exit if not
50 A6 50 D0 010D 368 MOVL R0,BRK$Q_REQID(R6) ; Save Requestor ID
38 A6 1C AC D0 0111 369 MOVL FLAGS(AP),BRK$Q_FLAGS(R6) ; Flags
34 A6 18 AC D0 0116 370 MOVL CARCON(AP),BRK$Q_CARCON(R6) ; Set carriage control
24 A6 28 AC D0 011B 371 MOVL ASTADR(AP),BRK$Q_ASTADR(R6) ; Ast routine
28 A6 2C AC D0 0120 372 MOVL ASTPRM(AP),BRK$Q_ASTPRM(R6) ; Ast routine parameter
0125 373
0125 374 ; Other misc. initialization
0125 375
0125 376 ASSUME BRK$W_STATUS+2 EQ BRK$W_SUCCESSCNT
0125 377 ASSUME BRK$W_STATUS+4 EQ BRK$W_TIMEOUTCNT
0125 378 ASSUME BRK$W_STATUS+6 EQ BRK$W_REFUSED CNT
70 A6 01 9B 0125 379 MOVZBW #SS$ NORMAL,BRK$W_STATUS(R6) ; Assume final status
78 A6 0000'8F B0 0129 380 MOVW #MSG$_TRMBRDCST,BRK$W_TRMMSG(R6); set mailbox prefix code
012F 381
012F 382 ; read PSL and save previous mode
012F 383
012F 384 MOVPSL R0 ; fetch PSL
02 50 DC 012F 385 EXTZV #PSL$V_PRVMOD,#PSL$S_PRVMOD,-
50 50 EF 0131 385 R0,R0 ; extract previous mode
67 A6 50 90 0136 387 MOVB R0,BRK$B_PRVMODE(R6) ; save
013A 388
013A 389 ; Set up search contexts
013A 390

```

```

54 A6 01 CE 013A 391 MNEGL #1,BRK$! PIDCTX(R6) ; wild card pid
013E 392 ASSUME BRK$!_UCBCTX+4 EQ BRK$!_DDBCTX ; assume alignment
013E 393
013E 394 ; Format screen message (if SCREEN requested)
013E 395
57 38 A6 DO 013E 396 MOVL BRK$!_FLAGS(R6),R7 ; Flags parameter
4D 57 08 E1 0142 397 BBC #BRK$!_SCREEN,R7,100$ ; Skip if not requested
50 57 9A 0146 398 MOVZBL R7,R0 ; Lines to clear
50 18 D1 0149 399 CMPL #BRK C_MAXLINES,R0 ; Greater than max?
91 1F 014C 400 BLSSU BADPARAM_EXIT ; Branch if yes
51 50 DO 014E 401 MOVL R0,R1 ; copy
52 51 08 C5 0151 402 MULL3 #8,R1,R2 ; bytes of erase pattern
0155 403
0155 404 ; Set up repeating erase line pattern on stack
0155 405
7E FEA7 CF 7D 0155 406 10$: MOVQ W^ERASE_PAT,-(SP) ; copy erase pattern
F8 50 F5 015A 407 SOBGTR R0,10$ ; one for each line
53 5E DO 015D 408 MOVL SP,R3 ; address of erase pattern
04 57 09 E1 0160 409 BBC #BRK$!_BOTTOM,R7,20$ ; Branch if message on top of screen
51 84 8F 9A 0164 410 MOVZBL #132,RT ; Set 'bottom' (note 132 >> 24)
0168 411 20$:
54 008C C6 3C 0168 412 MOVZWL BRK$!_MSGLEN(R6),R4 ; Size
55 008E C6 9E 016D 413 MOVAB BRK$!_MSGBUF(R6),R5 ; address of data
0172 414 $FAO_S CTRSTR = SCREEN_CTRSTR,-
0172 415 OUTLEN = BRK$!_SCRMSGLEN(R6),-
0172 416 OUTBUF = BRK$!_SCRMSGLEN(R6),-
0172 417 P1 = R1,- ; position top/bottom
0172 418 P2 = R2,- ; lines to erase * 8
0172 419 P3 = R3,- ; erase pattern address
0172 420 P4 = R4,- ; size of msgbuf
0172 421 P5 = R5 ; msgbuf address
03 50 E8 018D 422 BLBS R0,100$
FF54 31 0190 423 BRW ERROR_EXIT ; blew it
0193 424 100$:
0193 425
0193 426 ; Start initial QIO's up. AST's are disabled first so that a
0193 427 ; CPU limit exceeded ast cannot fire between assigning the
0193 428 ; channel and setting the CCBSM_IMGTMP flag. Something that would cause
0193 429 ; image exit to occur before the IMGTMP flag was set cannot be allowed.
0193 430 ; Disabling AST makes synchronization of CHECK_COMPLETE easier as well.
0193 431
0193 432 $SETAST_S ENBFLG = #0 ; Disable AST's
019C 433
019C 434 ; (At this point, R6 points to BRK structure, all others are scratch)
019C 435
57 60 A6 DO 019C 436 MOVL BRK$!_QIOCTX(R6),R7 ; QIO context area
58 04 3C 01A0 437 MOVZWL #BRK_C_SIMULCAST,R8 ; Number to do at one time
01A3 438 300$:
67 56 DO 01A3 439 MOVL R6,BRK2$!_COMMON(R7) ; Point back to common region
4F 10 01A6 440 BSBB DO_WRITE ; Do the write
07 50 E9 01A8 441 BLBC R0,350$ ; exit on error
57 0E A7 9E 01AB 442 MOVAB BRK2$!_LENGTH(R7),R7 ; Add size to qio context
F1 58 F5 01AF 443 SOBGTR R8,300$ ; Continue
01B2 444 350$:
50 DD 01B2 445 PUSHL R0 ; Save status
01B4 446
01B4 447 ; Before returning to user, see if there is a cluster to send to

```

			01B4	448					
		E1	01B4	449		BBC	#BRK\$V CLUSTER,-		
OE 38	A6		01B6	450			BRK\$L FLAGS(R6),360\$; Branch if "cluster" not requested
			01B9	451		IFNOCLSTR	360\$; or if not in cluster
00000000	'GF	16	01C1	452		JSB	G^EXE\$CSP_BRKTHRU		; send message
			01C7	453	360\$:				
	044E	30	01C7	454		BSBW	CHECK COMPLETE		; done? Deallocate BRK if so
			01CA	455		\$SETAST_S	ENBFLG = #1		; Enable AST's
	50	8ED0	01D3	456		POPL	_RO		; Restore status
50	2894	8F	B1	01D6	457	CMPW	#SS\$_NOOPER,RO		; no OPER priv?
		03	12	01DB	458	BNEQ	365\$-		; continue if not
	FF07	31	01DD	459		BRW	ERROR_EXIT		; take error exit
			01E0	460	365\$:				
50	01	9A	01E0	461		MOVZBL	#SS\$_NORMAL,RO		; Set success for everything else
		04	01E3	462	370\$:	RET			; Return to user
			01E4	463					

```

01E4 465 .SBTTL DO_WRITE - Queue a single write request
01E4 466 :++
01E4 467 :
01E4 468 : FUNCTIONAL DESCRIPTION:
01E4 469 :
01E4 470 :
01E4 471 : CALLING SEQUENCE:
01E4 472 : BSBW DO_WRITE
01E4 473 :
01E4 474 : INPUT PARAMETERS:
01E4 475 :
01E4 476 : R6 - BRK
01E4 477 : R7 - QIO context area
01E4 478 :
01E4 479 : IMPLICIT INPUTS:
01E4 480 : NONE
01E4 481 :
01E4 482 : OUTPUT PARAMETERS:
01E4 483 : NONE
01E4 484 :
01E4 485 : IMPLICIT OUTPUTS:
01E4 486 : NONE
01E4 487 :
01E4 488 : COMPLETION CODES:
01E4 489 : R0 - status
01E4 490 :
01E4 491 : SSS_NORMAL - all ok or error set in STATUS
01E4 492 : SSS_NOMOREPROC - done with all QIO's
01E4 493 :
01E4 494 : SIDE EFFECTS:
01E4 495 :
01E4 496 : Destroys R1,R2,R3,R4,R5
01E4 497 :
01E4 498 :--
01E4 499 :
01E4 500 UNLOCK_DB:
01E4 501 BBCC #BRK$V LOCKED,-
01E6 502 BRK$B_STS(R6),10$ ; clear locked flag
54 1C A6 DO 01E9 503 MOVL BRK$L_PCB(R6),R4 ; PCB
00000000'GF 16 01ED 504 JSB G^SCH$IOUNLOCK ; unlock
05 01F3 505 SETIPL #0 ; lower IPL
01F6 506 10$: RSB ; Return
01F7 507
01F7 508 DO_WRITE:
01F7 509
01F7 510 1C$:
EB 10 01F7 511 BSBB UNLOCK_DB ; Unlock data base
01FA 30 01F9 512 BSBW GET_NEXT_TERMINAL ; Get next terminal
01FC 513
01FC 514 ; returns with I/O database locked at IPL 2
01FC 515
E5 50 E9 01FC 516 BLBC R0,UNLOCK_DB ; branch if done (no more processes)
01FF 517
01FF 518 ; Test for broadcast to mailbox
01FF 519
55 58 A6 DO 01FF 520 MOVL BRK$L_UCBCTX(R6),R5 ; fetch UCB address
04 E1 0203 521 BBC #TT2$V_BRDCSTMBX,-

```

23	48	A5		0205	522		UCB\$L_DEVDEPND2(R5),40\$; Branch if not allowed
		55	DD	0208	523		R5	; Save ucb address
55	60	A5	DD	020A	524		UCB\$L_AMB(R5),R5	; Get address of associated mailbox
		18	13	020E	525		30\$; Branch if none
				0210	526			
				0210	527			; Send broadcast to assoicated mailbox
				0210	528			
53	008C	C6	3C	0210	529		MOVZWL BRK\$W_MSGLEN(R6),R3	; Get length of message
	53	16	C0	0215	530		ADDL2 #<BRK\$T_MSGBUF-BRK\$W_TRMMSG>,R3	; Add mailbox prefix overhead
54	78	A6	9E	0218	531		MOVAB BRK\$W_TRMMSG(R6),R4	; Set address of mailbox message
00000000		GF	16	021C	532		JSB G*EXE\$WRTMAILBOX	; Send message
	03	50	E9	0222	533		BLBC R0,30\$; branch if error sending to mailbox
	72	A6	B6	0225	534		INCW BRK\$W_SUCCESSCNT(R6)	; One more successful completion
				0228	535	30\$:		
			55	8ED0	0228	536	POPL R5	; Restore ucb address
				022B	537	40\$:		
00020001		8F	D3	022B	538		BITL #<TT\$M_NOBRDCST!TT\$M_PASSALL>,-	
	44	A5		0231	539		UCB\$L_DEVDEPEND(R5)	; test for NOBROADCAST or PASSALL
		C2	12	0233	540		BNEQ 10\$; skip if either set
		AD	10	0235	541		BSBB UNLOCK_DB	; unlock data base
				0237	542			
				0237	543			; Assign channel (if possible)
				0237	544			
			66	D5	0237	545	TSTL BRK\$Q_PRIVS(R6)	; assumes no privs in high longword
			OF	13	0239	546	BEQL 42\$; privs required non-null
				023B	547		\$SETPRV_S -	
				023B	548		ENBFLG = #1,-	; Enable privs
				023B	549		PRVADR = BRK\$Q_PRIVS(R6)	; Privs to set
				024A	550	42\$:		
	52	7E	7E	024A	551		MOVAQ -(SP),R2	; Allocate descriptor on stack
62	0C	A6	9A	024D	552		MOVZBL BRK\$T_DEVNAME(R6),(R2)	; Length
04	A2	0D	A6	9E	0251	553	MOVAB BRK\$T_DEVNAME+1(R6),4(R2)	; address
				0256	554			
				0256	555		\$ASSIGN_S -	
				0256	556		DEVNAM = (R2),-	; device name
				0256	557		CHAN = BRK2\$W_CHAN(R7)	; channel
				0264	558		ADDL #8,SP	; pop descriptor
	5E	08	C0	0264	558		BLBS R0,50\$; branch if ok
	19	50	E8	0267	559		INCW BRK\$W_REFUSED CNT(R6)	; Refused
	76	A6	B6	026A	560		MOVW R0,BRK\$W_STATUS(R6)	; record status
70	A6	50	B0	026D	561	45\$:	\$SETPRV_S -	
				0271	562		ENBFLG = #0,-	; Disable privs
				0271	563		PRVADR = BRK\$Q_PRIVS(R6)	; Privs to disable
				0271	564		BRW 10\$; Try another terminal
			FF74	31	0280	565		
				0283	566			
				0283	567			; modify the CCB so that the channel will be run down at image exit
				0283	568			
				0283	569	50\$:		
				0283	570		\$SETPRV_S -	
				0283	571		ENBFLG = #0,-	; Disable privs
				0283	572		PRVADR = BRK\$Q_PRIVS(R6)	; Privs to reset
				0292	573			
	50	0C	A7	3C	0292	574	MOVZWL BRK2\$W_CHAN(R7),R0	; Channel number
	50	50	CE	0296	575		MNEGL R0,R0	; Get negative
50	00000000	FF40	9E	0299	576		MOVAB @C1L\$GL_CCBASE[R0],R0	; Get CCB address
		02	88	02A1	577		BISB #CCB\$M_IMGTMP,-	
		08	A0	02A3	578		CCB\$B_STS(R0)	; Set image temporary channel

```

02A5 579
02A5 580
02A5 581
51 008E C6 9E 02A5 582 MOVAB BRK$T_MSGBUF(R6),R1 ; assume standard message
52 008C C6 3C 02AA 583 MOVZWL BRK$W_MSGLEN(R6),R2 ; and length
53 34 A6 D0 02AF 584 MOVL BRK$L_CARCON(R6),R3 ; and carriage control
54 2270 8F 3C 02B3 585 MOVZWL #<IOS$WRITEVBLK!-
02B8 586 IOSM_REFRESH!-
02B8 587 IOSM_BREAKTHRU!-
02B8 588 IOSM_CANCTRLO>,R4 ; I/O function code
11 38 A6 E1 02B8 589 BBC #BRK$V_SCREEN,- ; Branch if screen not requested
11 38 A6 1D E1 02BA 590 BRK$L_FLAGS(R6),70$
51 0C 48 A5 D0 02BF 592 UCBSL_DEVDEPND2(R5),70$ ; or not dec crt
51 6C A6 D0 02C2 593 MOVL BRK$L_SCRMSG(R6),R1 ; screen message
52 68 A6 3C 02C6 594 MOVZWL BRK$L_SCRMSGLEN(R6),R2 ; and length
53 53 D4 02CA 595 CLRL R3 ; no carriage control
05 11 02CC 596 BRB 75$ ; force no refresh for screen write
02CE 597 70$:
05 05 38 OA E1 02CE 598 BBC #BRK$V_NOREFRESH,-
54 2000 8F AA 02D0 599 BRK$L_FLAGS(R6),77$ ; Branch if not NO REFRESH
02D3 600 75$: BICW #IOSM_REFRESH,R4 ; Clear refresh flag
02D8 601 77$:
02D8 602
02D8 603
02D8 604
02D8 605
02D8 606
02D8 607
02D8 608
02D8 609
02D8 610
02D8 611
02D8 612
02D8 613
27 50 E9 02FD 614 BLBC R0,200$ ; error from QIO?
0A A6 B6 0300 615 INCW BRK$W_OUTCNT(R6) ; Increment outstanding count
0303 616
0303 617
0303 618
2C A6 7D 0303 619 MOVQ BRK$Q_TIMEOUT(R6),- ; (Test quad)
2C A6 0306 620 BRK$Q_TIMEOUT(R6) ; Time out requested?
19 13 0308 621 BEQL 80$ ; Branch if not
030A 622
030A 623
030A 624
030A 625
030A 626
030A 627
031C 628
04 50 E8 031C 629 BLBS R0,80$ ; branch if ok
70 A6 50 B0 031F 630 MOVW R0,BRK$W_STATUS(R6) ; Set final status
0323 631 80$:
50 01 9A 0323 632 MOVZBL #SS$_NORMAL,R0 ; exit
0326 633 100$:
05 0326 634 RSB
0327 635 ;

```



```
0327 636 ; Error during QIO
0327 637 .
0327 638 200$:
70 A6 50 80 0327 639 MOVW R0, BRK$W_STATUS(R6) ; Set final status
FEBE 31 0328 640 $DASSGN_S (CHAN = BRK2$W_CHAN(R7)) ; Deassign channel
0336 641 BRW -10$ ; Try again with this QIO context
0339 642
```

```

0339 644 .SBTTL GET_SENDTO - Handle SENDTO and SENDTYPE inputs
0339 645 :++
0339 646 :
0339 647 : FUNCTIONAL DESCRIPTION:
0339 648 :
0339 649 : Handle the SENDTYPE and SENDTO parameters and set up BRK.
0339 650 : Privilege is checked for all but BRK$C DEVICE writes.
0339 651 : Writes to same username are allowed without privilege.
0339 652 :
0339 653 : CALLING SEQUENCE:
0339 654 :
0339 655 : BSBW GET_SENDTO
0339 656 :
0339 657 : INPUT PARAMETERS:
0339 658 :
0339 659 : R6 - BRK
0339 660 : SENDTYPE(AP) - sendtype parameter
0339 661 : SENDTO(AP) - sendto parameter
0339 662 :
0339 663 : IMPLICIT INPUTS:
0339 664 : NONE
0339 665 :
0339 666 : OUTPUT PARAMETERS:
0339 667 : NONE
0339 668 :
0339 669 : IMPLICIT OUTPUTS.
0339 670 : NONE
0339 671 :
0339 672 : COMPLETION CODES:
0339 673 :
0339 674 : R0 - success or failure
0339 675 :
0339 676 : SIDE EFFECTS:
0339 677 :
0339 678 : R1-R5,R7 are destroyed.
0339 679 :--
0339 680
0339 681 GET_SENDTO:
0339 682
0339 683 MOVL SENDTYPE(AP),R7 ; fetch Send type
57 10 AC D0 0339 684 CMPL #BRK$C_MAXSENDTYPE,R7 ; Compare to maximum
57 04 D1 0339 685 BLSSU 5$ ; branch if error
12 1F 0340 686
0339 687 MOVW R7,BRK$W_SENDTYPE(R6) ; Save low order word
4C A6 57 B0 0342 688 CASE R7,- ; Case on send type
0346 689 <5$,- ; Invalid
0346 690 10$,- ; send to device name
0346 691 10$,- ; send to username
0346 692 150$,- ; send to all users
0346 693 150$>,- ; send to all terminals
0346 694 TYPE = W ; word context
50 14 3C 0354 695 5$: MOVZWL #SS$ _BADPARAM,R0 ; Set status
05 0357 696 7$: RSB
0358 697 :
0358 698 : single device or username requested
0358 699 :
51 0C AC D0 0358 700 10$: MOVL SENDTO(AP),R1 ; Get "send to" address

```

```

00000000'GF 16 035C 701 JSB G^EXE$PROBER_DSC ; test for read
      F2 50 E9 0362 702 BLBC R0,7$ ; exit on error
      51 51 3C 0365 703 MOVZWL R1,R1 ; zero high word
      EA 13 0368 704 BEQL 5$ ; Must be non-zero
      036A 705
      57 01 91 036A 706 CMPB #BRK$C_DEVICE,R7 ; device
      28 13 036D 707 BEQL 40$ ; Branch if yes
      036F 708
      036F 709 ; Must be Username
      036F 710
      51 0C B1 036F 711 CMPW #JIB$$_USERNAME,R1 ; max user name length
      E0 1F 0372 712 BLSSU 5$ ; error if so
      3C A6 51 90 0374 713 MOVVB R1,BRK$T_SENDNAME(R6) ; simply copy username ascic string
      51 DD 0378 714 PUSHL R1 ; Save Length
      62 51 28 037A 715 MOVCS R1,(R2),-
      3D A6 51 BED0 037D 716 BRK$T_SENDNAME+1(R6) ; and copy string
      54 1C A6 DO 037F 717 POPL R1 ; Restore Length
      54 0080 C4 DO 0382 718 MOVL BRK$L_PCB(R6),R4 ; Fetch PCB address
      0386 719 MOVL PCB$L_JIB(R4),R4 ; Fetch JIB
      038B 720
      038B 721 ; JIB$T_USERNAME is a 12 byte field, with NO BYTE COUNT!
      038B 722
      20 0C 2D 038B 723 CMPC5 #JIB$$_USERNAME,-
      3D A6 0C A4 038D 724 JIB$T_USERNAME(R4),#^A/ /,-
      51 51 0390 725 R1,BRK$T_SENDNAME+1(R6) ; compare strings, fill with blanks
      51 12 0393 726 BNEQ 150$ ; branch if not equal
      4B 11 0395 727 BRB 50$ ; names are same, no priv required
      0397 728
      0397 729 ; Device name, do a GETDVI to translate logical name
      0397 730
      0397 731 40$:
      54 5E DO 0397 732 MOVL SP,R4 ; Save SP
      55 7E DE 039A 733 MOVAL -(SP),R5 ; allocate scratch longword
      7E D4 039D 734 CLRL -(SP) ; end of list
      55 DD 039F 735 PUSHL R5 ; just a longword for device name length
      OD A6 9F 03A1 736 PUSHAB BRK$T_DEVNAME+1(R6) ; copy directly into device name area
      0020000F 8F DD 03A4 737 PUSHL #<DVI$DEVNAM@16>!-
      03AA 738 <BRK$$DEVNAME-1> ; size and getdvi code
      53 5E DO 03AA 739 MOVL SP,R3 ; save
      52 DD 03AD 740 PUSHL R2 ; address (device descriptor)
      51 DD 03AF 741 PUSHL R1 ; length
      51 5E DO 03B1 742 MOVL SP,R1 ; save
      03B4 743 $GETDVIW S -
      03B4 744 EFN = #BRK C DVIEFN,- ; event flag number
      03B4 745 DEVNAM = (R1),- ; get device name (and wait)
      03B4 746 ITMLST = (R3) ; item list
      OC A6 65 90 03CA 747 MOVVB (R5),BRK$T_DEVNAME(R6) ; Copy length
      SE 54 DO 03CE 748 MOVL R4,SP ; Restore SP
      OC A6 7D 03D1 749 MOVQ BRK$T_DEVNAME(R6),-
      3C A6 03D4 750 BRK$T_SENDNAME(R6) ; copy in case of cluster broadcast
      14 A6 7D 03D6 751 MOVQ BRK$T_DEVNAME+8(R6),-
      44 A6 03D9 752 BRK$T_SENDNAME+8(R6) ; copy in case of cluster broadcast
      07 50 E9 03DB 753 BLBC R0,110$ ; check status
      04 88 03DE 754 BISB #BRK$M_CHKPRIV,-
      66 A6 03E0 755 BRK$B_STS(R6) ; Set "check priv later" bit
      03E2 756 50$:
      50 01 3C 03E2 757 MOVZWL #SS$_NORMAL,R0 ; set ok

```

```
05 03E5 758 110$: RSB  
03E6 759  
03E6 760 : Check for OPER priv before allowing request  
03E6 761 :  
54 1C A6 D0 03E6 762 150$: MOVL BRK$PCB(R6),R4 : Fetch PCB address  
03EA 763 IFPRIV OPER,50$ : If priv ok, continue  
50 2894 8F 3C 03F0 764 MOVZWL #SS$_NOOPER,R0 : Set status  
05 03F5 765 RSB : exit  
03F6 766
```



```

048E 882 ; Username match found, scan device name for unit number
048E 883 ;
19 11 048E 884 BRB HAVE_NAME ; exit
0490 885 ;
0490 886 ; Send to all terminals/users
0490 887 ;
00DE 30 0490 888 ALL_TERMS:
00EB 30 0493 889 BSBW LOCKDB ; lock database
30 50 E8 0496 890 BSBW FIND_NEXT_TERM ; Find next terminal
04 11 0499 891 BLBS RO,HAVE_UCB ; Continue if OK
049B 892 BRB TERM_DONE ; Return proper status
049B 893 ;
70 A6 50 B0 049B 894 NEXT_TERM_ERROR:
049B 895 MOVW RO,BRK$W_STATUS(R6) ; Set final status
049F 896 ;
50 09AB 8F 3C 049F 897 TERM_DONE:
049F 898 MOVZWL #SS$ _NOMOREPROC,RO ; no more processes to send to
04A4 899 ;
66 A6 02 88 04A4 900 NO_MORE_TERM:
05 04A8 901 BISB #BRK$M_DONE,BRK$B_STS(R6) ; set done
04A9 902 RSB
04A9 903 ;
00C5 30 04A9 904 HAVE_NAME:
04A9 905 ;
04A9 906 BSBW LOCKDB ; lock database
04AC 907 ;
04AC 908 ; Map name into UCB address of this terminal
04AC 909 ;
7F 0D A6 9F 04AC 910 PUSHAB BRK$T_DEVNAME+1(R6) ; address of device name
0C A6 9A 04AF 911 MOVZBL BRK$T_DEVNAME(R6),-(SP) ; Length
51 SE D0 04B3 912 MOVL SP,R1 ; Address of descriptor
54 1C A6 D0 04B6 913 MOVL BRK$SL_PCB(R6),R4 ; Set PCB address
04BA 914 ;
00000000'GF 16 04BA 915 JSB G^IOCS$SEARCHDEV ; find the UCB (puts addr in R1)
SE 08 C0 04C0 916 ADDL #8,SP ; pop descriptor
D5 50 E9 04C3 917 BLBC RO,NEXT_TERM_ERROR ; error
55 51 D0 04C6 918 MOVL R1,R5 ; UCB address
04C9 919 ;
04C9 920 HAVE_UCB:
04C9 921 ;
04C9 922 ; Check availability, access and privilege
04C9 923 ;
28 38 02 E1 04C9 924 BBC #DEV$V TRM,-
A5 04CB 925 UCBSL_DEVCHAR(R5),3$ ; skip if not terminal
12 E1 04CE 926 BBC #DEV$V AVL,-
23 38 A5 04D0 927 UCBSL_DEVCHAR(R5),3$ ; skip terminal if not available
2040 8F B3 04D3 928 BITW #<DEV$M NET!DEV$M_SPL>,-
38 A5 04D7 929 UCBSL_DEVCHAR(R5) ; skip terminal if DECnet device
18 12 04D9 930 BNEQ 3$ ; or spooled
01 E0 04DB 931 BBS #DEV$V DET,-
16 3C A5 04DD 932 UCBSL_DEVCHAR2(R5),3$ ; skip terminal if detached
50 A6 E0 04E0 933 BBS BRK$SL_REQID(R6),-
OF 00AB C5 04E3 934 UCBSL_TL_BRKTHRU(R5),3$ ; Or specific class disabled
04 E0 04E7 935 BBS #TT2$V BRDCSTMBX,-
OD 48 A5 04E9 936 UCBSL_DEVDEPN2(R5),5$ ; must try this term if BRDCSTMBX
00020001 8F D3 04EC 937 BITL #<TT$M NOBRDCST!TT$M_PASSALL>,-
44 A5 04F2 938 UCBSL_DEVDEPEND(R5) ; test for NOBROADCAST or PASSALL

```

SYS
Pse

PSE

Pha

In1
Com
Pas
Syn
Pas
Syn
Pse
Crc
Ass

The
155
The
131
53

Mac

302
The
MAC

```

03 13 04F4 939 BEQL 5$ ; try terminal if neither set
      04F6 940
      04F6 941 ; For some reason, this device is not acceptable
      04F6 942
004F 31 04F6 943 3$: BRW 40$ ; skip to next terminal
      04F9 944
      04F9 945 5$: BBC #BRK$V_CHKPRIV,-
2E 66 A6 02 E1 04FB 946 ; BRK$B_STS(R6),30$ ; Branch if priv check not required
      04FE 947
      04FE 948 ; Search up process tree to see if owner
      04FE 949
51 1C A6 D0 04FE 950 MOVL BRK$L_PCB(R6),R1 ; PCB address
52 2C A5 D0 0502 951 MOVL UCBS$L_PID(R5),R2 ; Owner PID
52 60 A1 D1 0506 952 10$: CML PCBS$L_PID(R1),R2 ; compare PIDs
      20 13 050A 953 BEQL 30$ ; branch if OK
51 1C A1 3C 050C 954 MOVZWL PCBS$L_OWNER(R1),R1 ; Get index of owner
      0A 13 0510 955 BEQL 20$ ; If equal then none, must have priv
51 00000000'FF41 DO 0512 956 MOVL @L^SCH$GL_PCBVEC[R1],R1 ; Get Owner PCB address
      EA 11 051A 957 BRB 10$ ; Loop
      051C 958 20$:
54 1C A6 D0 051C 959 MOVL BRK$L_PCB(R6),R4 ; PCB address
      0520 960 IFPRIV OPER,30$ ; If privilege, ok to send message
50 2894 8F 3C 0526 961 MOVZWL #SS$_NOOPER,R0 ; set error
      05 052B 962 RSB ; exit
      052C 963
      052C 964 ; set up name and unit number
      C52C 965
      052C 966 30$:
      052C 967 PUSHL R7 ; Save R7
57 50 0F 9A 052E 968 MOVZBL #BRK$$ DEVNAME-1,R0 ; Size of buffer
57 0C A6 9E 0531 969 MOVAB BRK$T_DEVNAME(R6),R7 ; Address of buffer
51 01 A7 9E 0535 970 MOVAB 1(R7),R1 ; Address past byte count
      54 01 CE 0539 971 MNEGL #1,R4 ; Standard device name
00000000'GF 16 053C 972 JSB G^IOC$CVT_DEVNAM ; convert to regular device name
      57 8E D0 0542 973 POPL R7 ; Restore R7
      09 50 E8 0545 974 BLBS R0,50$ ; skip this device if error
      0548 975
      0548 976 ; This terminal failed, reset and loop
      0548 977
      0548 978 40$:
      FC99 30 0548 979 BSBW UNLOCK_DB ; unlock database
      76 A6 B6 054B 980 INCW BRK$W_REFUSED(CNT(R6)) ; Increment
      FE A5 31 054E 981 BRW GET_NEXT_TERMINAL ; Loop
      0551 982 50$:
      0C A6 51 90 0551 983 MOVB R1,BRK$T_DEVNAME(R6) ; Length of string
      58 A6 55 D0 0555 984 MOVL R5,BRK$L_UCBCTX(R6) ; save UCB address
      0559 985
      0559 986 ; set up TRMNAME for mailbox message
      0559 987
      54 A5 B0 0559 988 MOVW UCBS$W_UNIT(R5),-
      7A A6 B0 055C 989 BRK$W_TRMUNIT(R6) ; unit number
50 28 A5 D0 055E 990 MOVL UCBS$L_DDB(R5),R0 ; Fetch DDB
      14 A0 7D 0562 991 MOVQ DDB$T_NAME(R0),-
      7C A6 7D 0565 992 BRK$T_TRMNAME(R6) ; set TRMNAME (first half)
      1C A0 7D 0567 993 MOVQ DDB$T_NAME+8(R0),-
      0084 C6 7A 056A 994 BRK$T_TRMNAME+8(R6) ; set TRMNAME (second half)
      50 01 9A 056D 995 MOVZBL #SS$_NORMAL,R0 ; set success

```


	05	0570	996		RSB				
		0571	997						
		0571	998	LOCKDB:					
	E2	0571	999		BBSS	#BRK\$V_LOCKED,-			
00		0573	1000			BRK\$B_STS(R6),10\$; set locked flag	
OA 66		0576	1001		MOVL	BRK\$L_PCB(R6),R4		; Set PCB address	
54 1C	D0	057A	1002		JSB	G^SCH\$IOLOCKR		; lock I/O database for read access	
0000000C	16	0580	1003	1.s:	RSB				
'GF	05	0581	1004						

```

0581 1006 .SBTTL FIND_NEXT_TERM - Search I/O database
0581 1007 :++
0581 1008 :
0581 1009 : FUNCTIONAL DESCRIPTION:
0581 1010 :
0581 1011 : Given the UCB context of the last terminal, find the next
0581 1012 : terminal that qualifies. Terminal must be online.
0581 1013 :
0581 1014 : If looking for all terminals, an unowned terminal is skipped
0581 1015 : if autobauding.
0581 1016 :
0581 1017 : CALLING SEQUENCE:
0581 1018 :
0581 1019 : BSBW FIND_NEXT_TERM
0581 1020 :
0581 1021 : INPUT PARAMETERS:
0581 1022 :
0581 1023 : R6 - BRK
0581 1024 :
0581 1025 : IMPLICIT INPUTS:
0581 1026 : NONE
0581 1027 :
0581 1028 : OUTPUT PARAMETERS:
0581 1029 :
0581 1030 : R5 - points to UCB
0581 1031 :
0581 1032 : COMPLETION CODES:
0581 1033 :
0581 1034 : R0 = 1, R5 is UCB
0581 1035 : R0 = 0, no more terminals
0581 1036 :
0581 1037 : All other registers preserved.
0581 1038 :
0581 1039 : SIDE EFFECTS:
0581 1040 : NONE
0581 1041 :
0581 1042 :--
0581 1043 :
0581 1044 : FIND_NEXT_TERM:
0581 1045 :
0581 1046 : PUSHR #M<R10,R11> ; Save
5A 0C00 8F BB 0585 1047 : MOVQ BRK$$_UCBCTX(R6),R10 ; ucb and ddb pair
5A 58 A6 7D
0589 1048 :
0589 1049 : BEQL 20$ ; *** TEMP
058B 1050 : CLRL R0 ; *** TEMP
30 AA FFFFFFFF 8F D1 058D 1051 : CMPL #-1,UCB$_LINK(R10) ; *** TEMP until SCAN_IODB enhanced
2F 13 0595 1052 : BEQL 40$ ; *** TEMP to handle missing UCBs
0597 1053 20$:
0597 1054 : JSB G^IOC$SCAN_IODB ; Fetch next UCB
00000000'GF 16 059D 1055 : BLBC R0,40$ ; branch if done
26 50 E9
05A0 1056 :
05A0 1057 : Have valid UCB, see if it's a terminal
05A0 1058 :
05A0 1059 : BBC #DEVS$_TRM,- ;
F2 38 AA E1 05A2 1060 : UCBS$_DEVCHAR(R10),20$ ; Get next if not terminal
04 E1 05A5 1061 : BBC #UCBS$_ONLINE,- ;
ED 64 AA 05A7 1062 : UCBS$_STS(R10),20$ ; next ucb if offline

```

```

5C AA B5 05AA 1063      TSTW  UCBSW_REFC(R10)      ; terminal allocated?
   10 12 05AD 1064      BNEQ  30$                 ; yes, do write
   04 B1 05AF 1065      CMPW  #BRK$C_ALLTERMS,-   ;
4C A6   05B1 1066      BRK$W_SENDTYPE(R6)      ; for all terminals?
   E2 12 05B3 1067      BNEQ  20$                 ; no, try next
   01 E1 05B5 1068      BBC   #TT2$V_AUTOBAUD,-   ;
05 48 AA   05B7 1069      UCBSL_DEVDEPND2(R10),30$ ; branch if not autobaud
   76 A6 B6 05BA 1070      INCW  BRK$W_REFUSEDcnt(R6) ; Refused due to autobaud
   D8 11 05BD 1071      BRB   20$                 ; try again
   05BF 1072
55 5A D0 05BF 1073 30$:  MOVL  R10,R5                 ; Set output
58 A6 5A 7D 05C2 1074      MOVQ  R10,BRK$L_UCBCTX(R6) ; save ucb and ddb pair
   05C6 1075
0C00 8F BA 05C6 1076 40$:  POPR  #^M<R10,R11>       ; Restore
   05 05CA 1077          RSB   ; Return (assumes R0 unmodified from
   05CB 1078          ; call above)
   05CB 1079
   05CB 1080

```

```

05CB 1082      .SBTTL QIO_DONE - process qio completion
05CB 1083
05CB 1084      :++
05CB 1085      :
05CB 1086      : FUNCTIONAL DESCRIPTION:
05CB 1087      :
05CB 1088      :     Completion AST routine for QIO to terminal.
05CB 1089      :
05CB 1090      : CALLING SEQUENCE:
05CB 1091      :
05CB 1092      :     CALLG (as an AST)
05CB 1093      :
05CB 1094      : INPUT PARAMETERS:
05CB 1095      :
05CB 1096      :     4(AP) - Address of per QIO context within BRK
05CB 1097      :
05CB 1098      : IMPLICIT INPUTS:
05CB 1099      :     NONE
05CB 1100      :
05CB 1101      : OUTPUT PARAMETERS:
05CB 1102      :     NONE
05CB 1103      :
05CB 1104      : IMPLICIT OUTPUTS:
05CB 1105      :     NONE
05CB 1106      :
05CB 1107      : COMPLETION CODES:
05CB 1108      :     NONE
05CB 1109      :
05CB 1110      : SIDE EFFECTS:
05CB 1111      :
05CB 1112      :     May result in another QIO being performed or
05CB 1113      :     completion of service.
05CB 1114      :
05CB 1115      :--
05CB 1116
OFFC 05CB 1117 QIO_DONE:      .WORD  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
05CD 1118
57  04 AC  D0 05CD 1119      MOVL  4(AP),R7          ; QIO context
56  67 D0 05D1 1120      MOVL  BRK2$COMMON(R7),R6 ; BRK common area
2C  A6 7D 05D4 1121
2C  A6 05D4 1122      MOVQ  BRK$Q_TIMEOUT(R6),-
0B  13 05D7 1123      BRK$Q_TIMEOUT(R6)      ; Time out specified?
05D9 1124      BEQL  20$             ; branch if no
05DB 1125      $CANTIM_S REQIDT = R7 ; Cancel timer
05E6 1126 20$:
05E6 1127      $DASSGN_S CHAN = BRK2$W_CHAN(R7) ; Deassign channel
05F1 1128      :
05F1 1129      : check IOSB
05F1 1130
50  04 A7 3C 05F1 1131      MOVZWL BRK2$Q_IOSB(R7),R0 ; Fetch status
11  50 E8 05F5 1132      BLBS  R0,30$-          ; branch if no error
50  0830 8F B1 05F8 1133      CMPW  #SS$_CANCEL,R0   ; Make sure it was cancel (from timecut)
0D  13 05FD 1134      BEQL  40$             ;
50  2C B1 05FF 1135      CMPW  #SS$_ABORT,R0   ; Make sure it was cancel (from timeout)
08  13 0602 1136      BEQL  40$             ;
76  A6 B6 0604 1137      INCW  BRK$W_REFUSED(CNT(R6)) ; One more non-successful completion
03  11 0607 1138      BRB   40$             ; continue

```

72	A6	B6	0609	1139	30\$:				
			0609	1140		INCW	BRK\$W_SUCCESSCNT(R6)	:	One more successful completion
			060C	1141	40\$:				
0A	A6	B7	060C	1142		DECW	BRK\$W_OUTCNT(R6)	:	One less outstanding
FB	E5	D	060F	1143		BSBW	DO_WRITE	:	Do next write with this context
02	50	E8	0612	1144		BLBS	RO,100\$:	branch if success
			0615	1145					
01		10	0615	1146		BSBB	CHECK_COMPLETE	:	check for completion
		04	0617	1147	100\$:	RET		:	exit ast

```

0618 1149 .SBTTL CHECK_COMPLETE - Check completion criterion
0618 1150 :++
0618 1151 :
0618 1152 : FUNCTIONAL DESCRIPTION:
0618 1153 :
0618 1154 : See if service is done with all it's duties and
0618 1155 : complete if so.
0618 1156 :
0618 1157 : CALLING SEQUENCE:
0618 1158 :
0618 1159 : BSBW CHECK_COMPLETE
0618 1160 :
0618 1161 : INPUT PARAMETERS:
0618 1162 :
0618 1163 : R6 - BRK
0618 1164 :
0618 1165 : IMPLICIT INPUTS:
0618 1166 : NONE
0618 1167 :
0618 1168 : OUTPUT PARAMETERS:
0618 1169 : NONE
0618 1170 :
0618 1171 : IMPLICIT OUTPUTS:
0618 1172 : NONE
0618 1173 :
0618 1174 : COMPLETION CODES:
0618 1175 : NONE
0618 1176 :
0618 1177 : SIDE EFFECTS:
0618 1178 :
0618 1179 : R0, R1 destroyed
0618 1180 :
0618 1181 :--
0618 1182 :
0618 1183 CHECK_COMPLETE:
0A A6 B5 0618 1184 TSTW BRK$W_OUTCNT(R6) ; I/O still outstanding?
01 01 13 0618 1185 BEQL 10$ ; branch if done
05 05 05 0618 1186 RSB ; otherwise, exit
061E 1187 ;
061E 1188 ; Return status and complete service
061E 1189 ;
061E 1190 10$:
51 20 A6 D0 061E 1191 MOVL BRK$L_IOSB(R6),R1 ; return IOSB
13 13 0622 1192 BEQL 30$ ; Branch if none
0B 70 A6 E9 0624 1193 BLBC BRK$W_STATUS(R6),20$ ; Branch if other error occurred
72 A6 B5 0628 1194 TSTW BRK$W_SUCCESSCNT(R6) ; any messages sent?
06 12 062B 1195 BNEQ 20$ ; branch if yes
0084 8F B0 062D 1196 MOVW #$$$ DEVOFFLINE, -
70 A6 0631 1197 BRK$W_STATUS(R6) ; set device off line
61 70 A6 7D 0633 1198 20$: MOVQ BRK$W_STATUS(R6),(R1) ; Return status and counts
0637 1199 ;
0637 1200 ; Deliver AST if necessary
0637 1201 ;
0637 1202 70$:
51 24 A6 D0 0637 1203 MOVL BRK$L_ASTADR(R6),R1 ; Fetch address
12 13 063B 1204 BEQL 40$ ; Branch if no AST
50 67 A6 9A 063D 1205 MOVZBL BRK$B_PRVMODE(R6),R0 ; Set previous mode

```



```

066F 1236          .SBTTL QIO_TIMEOUT - process qio timeout
066F 1237
066F 1238 :++
066F 1239 :
066F 1240 : FUNCTIONAL DESCRIPTION:
066F 1241 :
066F 1242 :
066F 1243 : CALLING SEQUENCE:
066F 1244 :     NONE
066F 1245 :
066F 1246 : INPUT PARAMETERS:
066F 1247 :
066F 1248 :     4(AP) - QIO context address
066F 1249 :
066F 1250 : IMPLICIT INPUTS:
066F 1251 :     NONE
066F 1252 :
066F 1253 : OUTPUT PARAMETERS:
066F 1254 :     NONE
066F 1255 :
066F 1256 : IMPLICIT OUTPUTS:
066F 1257 :     NONE
066F 1258 :
066F 1259 : COMPLETION CODES:
066F 1260 :     NONE
066F 1261 :
066F 1262 : SIDE EFFECTS:
066F 1263 :     NONE
066F 1264 :
066F 1265 :--
0040 066F 1266 QIO_TIMEOUT:      .WORD   ^M<R6>
50   04 AC   D0 0671 1268
56   60   D0 0671 1269      MOVL   4(AP),R0          ; fetch context
74   A6   B6 0675 1270      MOVL   BRK2$COMMON(R0),R6 ; fetch common area address
0678 1271      INCW   BRK$W_TIMEOUTCNT(R6) ; increment time out count ???
067B 1272      $CANCEL_S BRK2$W_CHAN(R0) ; Cancel I/O, wait for qio_done ast
04   0686 1273      RET
0687 1274

```

PSE

: B
\$AB

Pha

Ini
Com
Pas
Sym
Pas
Sy.
Pse
Cro
Ass

The
110
The
241
78

Mac

-\$2
-\$2
TOT

239

The

MAC


```

007C 0687 1276
      0687 1277
      0689 1278
6D 00000000'GF 9E 0689 1279
      0690 1280
      51 04 AC D0 0690 1281
      0694 1282
      0694 1283
      0694 1284
      53 04 9A 0694 1285
      52 08 AC D0 0697 1286
      0A 13 0698 1287
      53 03 9A 069D 1288
      62 D5 06A0 1289
      03 13 06A2 1290
      53 01 9A 06A4 1291
      06A7 1292 20$:
      54 D4 06A7 1293
      55 20 9A 06A9 1294
      6C 04 D1 06AC 1295
      04 12 06AF 1296
      54 0C AC 7D 06B1 1297
      06E5 1298 30$:
      56 7E 7E 06B5 1299
      06B8 1300
      06B8 1301
      06B8 1302
      06B8 1303
      06B8 1304
      06B8 1305
      06B8 1306
      06B8 1307
      06B8 1308
      03 50 E9 06D3 1309
      50 66 3C 06D6 1310
      06D9 1311 60$:
      50 00002894 8F D1 06D9 1312
      03 12 06E0 1313
      50 24 3C 06E2 1314
      04 06E5 1315 70$:
      06E6 1316
      06E6 1317 .END

.ENTRY EXE$BRDCST, ^M<R2,R3,R4,R5,R6> ; OLD SYS$BRDCST...
MOVAB G^EXE$SIGTORET,(FP) ; Set condition handler
MOVL 4(AP),R1 ; Get message address
;
; Figure out send type
;
MOVZBL #BRK$C_ALLTERMS,R3 ; Assume all terminals
MOVL 8(AP),R2 ; Fetch descriptor address
BEQL 20$ ; Branch if all terminals
MOVZBL #BRK$C_ALLUSERS,R3 ; Assume all users
TSTL (R2) ; Check length
BEQL 20$ ; Branch if zero
MOVZBL #BRK$C_DEVICE,R3 ; Must be terminal name
;
CLRL R4 ; Clear R4 - no flags
MOVZBL #^A/ /,R5 ; Default carcon if only 2 parameters
CML #4,(AP) ; More parameters?
BNEQ 30$ ; Branch if no
MOVQ 12(AP),R4 ; Flags and carcon
;
MOVAQ -(SP),R6 ; allocate IOSB on stack
$BRKTHRU S - ; Call breakthru and wait
EFN = #BRK_C_BRDCSTEFN,-
MSGBUF = (R1),-
SENDTO = (R2),-
SNDTYP = R3,-
FLAGS = R4,-
CARCON = R5,-
TIMOUT = #10,- ; *** SYSGEN PARAMETER ???
IOSB = (R6)
BLBC R0,60$ ; Branch if error
MOVZWL (R6),R0 ; Use IOSB status
;
CML #SS$_NOOPER,R0 ; new status?
BNEQU 70$ ; nope, exit
MOVZWL #SS$_NOPRIV,R0 ; set status
RET ; EXIT

```

```

$ST1 = 00000000
$ST2 = 00000008
ACCVIO_EXIT = 000000E4 R 02
ALL_OK = 000000EF R R 02
ALL_TERMS = 00000490 R 02
ASTADR = 00000028
ASTPRM = 0000002C
BADPARAM_EXIT = 000000DF R 02
BRKSB_PMODE = 00000067
BRKSB_STS = 00000066
BRKSC_ALLTERMS = 00000004
BRKSC_ALLUSERS = 00000003
BRKSC_DEVICE = 00000001
BRKSC_LENGTH = 0000008E
BRKSC_MAXSENDTYPE = 00000004
BRKSL_ASTADR = 00000024
BRKSL_ASTPRM = 00000028
BRKSL_CARCON = 00000034
BRKSL_DDBCTX = 0000005C
BRKSL_FLAGS = 00000038
BRKSL_IOSB = 00000020
BRKSL_PCB = 0000001C
BRKSL_PIDCTX = 00000054
BRKSL_QIOCTX = 00000060
BRKSL_REQID = 00000050
BRKSL_SCRMSG = 0000006C
BRKSL_SCRMSGLEN = 00000068
BRKSL_UCBCTX = 00000058
BRKSM_CHKPRIV = 00000004
BRKSM_DONE = 00000002
BRKSO_PRIVS = 00000000
BRKSO_TIMEOUT = 0000002C
BRKSS_DEVNAME = 00000010
BRKSS_SENDNAME = 00000010
BRKSS_TRMNAME = 00000010
BRKST_DEVNAME = 0000000C
BRKST_MSGBUF = 0000008E
BRKST_SENDNAME = 0000003C
BRKST_TRMNAME = 0000007C
BRKSV_BOTTOM = 00000009
BRKSV_CHKPRIV = 00000002
BRKSV_CLUSTER = 0000000B
BRKSV_DONE = 00000001
BRKSV_LOCKED = 00000000
BRKSV_NOREFRESH = 0000000A
BRKSV_SCREEN = 00000008
BRKSW_EFN = 00000064
BRKSW_MSGLEN = 0000008C
BRKSW_OUTCNT = 0000000A
BRKSW_REFUSED CNT = 00000076
BRKSW_SECONDS = 0000004E
BRKSW_SENDTYPE = 0000004C
BRKSW_SIZE = 00000008
BRKSW_STATUS = 00000070
BRKSW_SUCCESSCNT = 00000072
BRKSW_TIMEOUTCNT = 00000074
BRKSW_TRMSG = 00000078

```

```

BRKSW_TRMUNIT = 0000007A
BRK2SC_LENGTH = 0000000E
BRK2SL_COMMON = 00000000
BRK2SQ_IOSB = 00000004
BRK2SW_CHAN = 0000000C
BRK_C_BRDCSTEFN = 0000001F
BRK_C_CLUTIMEOUT = 0000000F
BRK_C_DVIEFN = 0000001F
BRK_C_JPIEFN = 0000001F
BRK_C_MAXLINES = 00000018
BRK_C_MINTIME = 00000004
BRK_C_QIOEFN = 0000001F
BRK_C_SIMULCAST = 00000004
BRK_C_TIMEFN = 0000001F
CARCON = 00000018
CCBSB_STS = 00000008
CCBSM_IMGMTMP = 00000002
CHECK_COMPLETE = 00000618 R 02
CLUSGC_CLUB = ***** X 02
CTLSGL_CCBASE = ***** X 02
DDBSS_NAME = 00000010
DDBST_NAME = 00000014
DEVSM_NET = 00002000
DEVSM_SPL = 00000040
DEVSV_AVL = 00000012
DEVSV_DET = 00000001
DEVSV_TRM = 00000002
DO_WRITE = 000001F7 R 02
DVIS_DEVNAM = 00000020
EFN = 00000004
ERASE_PAT = 00000000 R 02
ERROR_EXIT = 000000E7 R 02
EXESACOP1MAG = ***** X 02
EXESBRDCST = 00000687 RG 02
EXESBRKTHRU = 00000025 RG 02
EXESCSP_BRKTHRU = ***** X 02
EXESDEAF1 = ***** X 02
EXESPROBER_DSC = ***** X 02
EXESSIGTORET = ***** X 02
EXESWRMAILBOX = ***** X 02
FIND_NEXT_TERM = 00000581 R 02
FLAGS = 0000001C
GET_NEXT_TERMINAL = 000003F6 R 02
GET_SENDTO = 00000339 R R 02
HAVE_NAME = 000004A9 R R 02
HAVE_UCB = 000004C9 R 02
IOSM_BREAKTHRU = 00000200
IOSM_CANCTRLO = 00000040
IOSM_REFRESH = 00002000
IOS_WRITEVBLK = 00000030
IOCSVT_DEVNAM = ***** X 02
IOCSSCAN_IODB = ***** X 02
IOCSSSEARCHDEV = ***** X 02
IOSB = 00000014
JIBSS_USERNAME = 0000000C
JIBST_USERNAME = 0000000C
JPIS_TERMINAL = 0000031D

```

! 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	00000020 (44.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
YSEXEPAGED	000006E6 (1766.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.07	00:00:01.77
Command processing	112	00:00:00.50	00:00:04.76
Pass 1	623	00:00:27.10	00:01:22.88
Symbol table sort	0	00:00:04.50	00:00:12.69
Pass 2	220	00:00:05.39	00:00:20.48
Symbol table output	24	00:00:00.21	00:00:00.42
Psect synopsis output	2	00:00:00.03	00:00:00.22
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1012	00:00:37.82	00:02:03.24

The working set limit was 2100 pages.
155190 bytes (304 pages) of virtual memory were used to buffer the intermediate code.
There were 150 pages of symbol table space allocated to hold 2771 non-local and 66 local symbols.
1317 source lines were read in Pass 1, producing 24 object records in Pass 2.
53 pages of virtual memory were used to define 51 macros.

! Macro library statistics !

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	15
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	32
TOTALS (all libraries)	47

3023 GETS were required to define 47 macros.

There were no errors, warnings or information messages.

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



