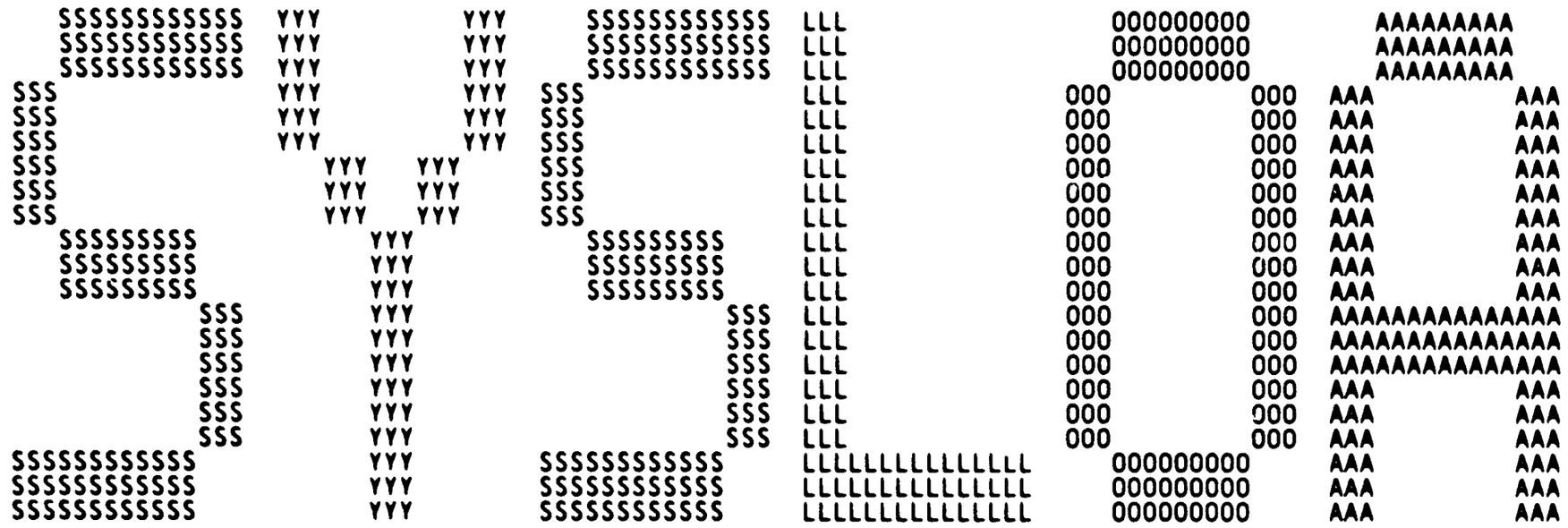
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA

```
  SSSSSSSS    CCCCCCCC    SSSSSSSS  LL           000000      AAAAAA
  SSSSSSSS    CCCCCCCC    SSSSSSSS  LL           000000      AAAAAA
SS            CC          SS        LL         00      00  AA      AA
SS            CC          SS        LL         00      00  AA      AA
SS            CC          SS        LL         00      00  AA      AA
SS            CC          SS        LL         00      00  AA      AA
  SSSSSS      CC            SSSSSS  LL         00      00  AA      AA
  SSSSSS      CC            SSSSSS  LL         00      00  AA      AA
        SS    CC                  SS  LL       00      00  AAAAAAAAAA
        SS    CC                  SS  LL       00      00  AAAAAAAAAA
        SS    CC                  SS  LL       00      00  AA      AA      ....
        SS    CC                  SS  LL       00      00  AA      AA      ....
SSSSSSSS      CCCCCCCC  SSSSSSSS    LLLLLLLLLL   000000    AA      AA      ....
SSSSSSSS      CCCCCCCC  SSSSSSSS    LLLLLLLLLL   000000    AA      AA      ....


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
```

(1) 142 DECLARATIONS
(1) 292 INITIALIZATION CODE
(1) 496 SCS$ACCEPT
(1) 546 SCS$ALLOC_CDT
(1) 629 SCS$ALLOC_RSPID
(1) 682 SCS$CONFIG_PTH
(1) 814 SCS$CONFIG_SYS
(1) 924 SCS$CONNECT
(1) 1065 SCS$DEALL_CDT
(1) 1119 SCS$DEALL_RSPID - Dellocate a response id
(1) 1120 SCS$RECYL_RSPID - Recycle a response id
(1) 1208 SCS$DIRECTORY
(1) 1339 SCS$DISCONNECT
(1) 1379 SCS$ENTER
(1) 1438 SCS$FIND_RDTE - Find RDTE for RSPID
(1) 1486 SCS$LISTEN
(1) 1544 SCS$LOCLOOKUP
(1) 1597 SCS$REMOVE
(1) 1658 SCS$RESUMEWAITR
(1) 1733 SCS$LKP_RDTCDRP
(1) 1830 SCS$LKP_RDTWAIT
(1) 1924 SCS$LKP_MSGWAIT - Scan message wait queues
(2) 2048 SCS$NEW_SB - New/Reused System Block Available for Polling
(3) 2110 SCS$POLL_PROC - Declare a process name to the poller
(4) 2181 SCS$POLL_MODE - Enable/Disable polling for a process
(5) 2258 START_POLL - Start poll of requested processes
(6) 2441 SCS$DIR_LOOKUP - look up process names on remote node
(11) 2683 SCS$POLL_MBX - Declare Polling Notification Mailbox
(12) 2820 SCS$CANCEL_MBX - Cancel Polling Notification Mailbox
(13) 2867 SCS$SHUTDOWN - Shutdown all SCS virtual circuits

```
00000001  0000      1          PRMSW=1                         ; SET SWITCH TO GENERATE PARAMETER DESCRIPTO
                    0000      1          .IF     NDF,PRMSW
                    0000      2          .TITLE  SCSVEC - System Communications Service Vectors
                    0000      3          .IFF
                    0000      4          .TITLE  SCSLOA - System Communications Service Loadcode
                    0000      5          .ENDC
                    0000      6          .IDENT  'V04-000'
                    0000      7  ;
                    0000      8  ;*****************************************************************************
                    0000      9  ;*                                                                           *
                    0000     10  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
                    0000     11  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
                    0000     12  ;*  ALL RIGHTS RESERVED.                                                     *
                    0000     13  ;*                                                                           *
                    0000     14  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
                    0000     15  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
                    0000     16  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
                    0000     17  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
                    0000     18  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
                    0000     19  ;*  TRANSFERRED.                                                             *
                    0000     20  ;*                                                                           *
                    0000     21  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
                    0000     22  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
                    0000     23  ;*  CORPORATION.                                                             *
                    0000     24  ;*                                                                           *
                    0000     25  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
                    0000     26  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
                    0000     27  ;*                                                                           *
                    0000     28  ;*                                                                           *
                    0000     29  ;*****************************************************************************
                    0000     30  ;
                    0000     31
                    0000     32  ;++
                    0000     33  ; FACILITY:
                    0000     34  ;       Executive data base
                    0000     35  ;
                    0000     36  ; ABSTRACT:
                    0000     37  ;       This module contains the global entry point vectors for System
                    0000     38  ;       Communications Services (SCS).  The targets of these vectors are
                    0000     39  ;       updated to point to the actual SCS image after it is loaded into
                    0000     40  ;       non-paged pool.
                    0000     41  ;
                    0000     42  ; ENVIRONMENT:
                    0000     43  ;
                    0000     44  ; AUTHOR:  KERBEY T. ALTMANN, CREATION DATE:    22-APR-1981
                    0000     45  ;
                    0000     46  ; MODIFIED BY:
                    0000     47  ;
                    0000     48  ;       V03-033 DWT0237         David W. Thiel          10-Aug-1984
                    0000     49  ;               Correct reference to RDT$L_MAXRDIDX to match change
                    0000     50  ;               made in V03-032 below.
                    0000     51  ;
                    0000     52  ;       V03-032 NPK3058         N. Kronenberg           25-Jul-1984
                    0000     53  ;               Fix maximume RDTE index to be index rather than number
                    0000     54  ;               of RDTEs configured.
                    0000     55  ;
                    0000     56  ;       V03-031 NPK3056         N. Kronenberg           23-Jul-1984
```

```
0000     57 ;                 Add capacity in Connection Descriptor List for addition
0000     58 ;                 of CDT's beyond the sysgened number.  Add to SCS$ALLOC_CDT
0000     59 ;                 the ability to allocate more CDTs from pool if the
0000     60 ;                 are no free CDTs linked to the CDL.
0000     61 ;                 Fix error in computation of CDL max index in CDL header
0000     62 ;                 (it was # CDTs not index.)
0000     63 ;
0000     64 ;        V03-030 TCM0001         Trudy C. Matthews       20-Jul-1984
0000     65 ;                 Remove routine IOC$THREADCRB; put it in IOSUBNPAG instead.
0000     66 ;
0000     67 ;        V03-029 KTA3113         Kerbey T. Altmann       20-Mar-1984
0000     68 ;                 Add new entry - SCS$SHUTDOWN to shut down all SCS virtual
0000     69 ;                 circuits.  Add new pointers for MSCP image and list of PDTs.
0000     70 ;
0000     71 ;        V03-028 KTA3102         Kerbey T. Altmann       09-Feb-1984
0000     72 ;                 Prevent RDT/CDT sequence numbers from ever being zero.
0000     73 ;
0000     74 ;        V03-027 TMK0001         Todd M. Katz            05-Jan-1984
0000     75 ;                 Change SCS$DIRECTORY so that it sends a counted sequential
0000     76 ;                 message back to the remote SYSAP with the results of the
0000     77 ;                 directory lookup. SCS$DIRECTORY was sending an un-counted
0000     78 ;                 sequential message, and if the local SCSMAXMSG size was greater
0000     79 ;                 than the remote SCSMAXMSG size, then this was resulting in
0000     80 ;                 the remote port driver receiving a sequential message larger
0000     81 ;                 in size then any of its receive buffers.
0000     82 ;
0000     83 ;        V03-026 KDM0063         Kathleen D. Morse       03-Aug-1983
0000     84 ;                 Replace random number generation usage of PR$_TODR with
0000     85 ;                 usage of EXE$GQ_SYSTIME.
0000     86 ;
0000     87 ;        V03-025 NPK3029         N. Kronenberg           22-Jul-1983
0000     88 ;                 Zero most fields in a freshly allocated CDT (needed
0000     89 ;                 for per connection counters.)
0000     90 ;                 Modify SCS$CONFIG_PTH to take advantage of rearranged
0000     91 ;                 path block.
0000     92 ;
0000     93 ;        V03-024 MSH0010         Maryann Hinden          11-Jul-1983
0000     94 ;                 Fix method of addressing SCS$GA_EXISTS.
0000     95 ;
0000     96 ;        V03-023 MSH0009         Maryann Hinden          27-Jun-1983
0000     97 ;                 Modifications to error handling for SCS$DIRECTORY.
0000     98 ;                 Keep process poller from polling a given system until
0000     99 ;                 current poll is complete.
0000    100 ;
0000    101 ;        V03-022 MSH0008         Maryann Hinden          27-Jun-1983
0000    102 ;                 Use remote process name in DIR_LOOKUP which won't get
0000    103 ;                 deallocated.
0000    104 ;
0000    105 ;        V03-021 MSH0007         Maryann Hinden          23-Jun-1983
0000    106 ;                 Use $PRCPOLDEF.  Check for nonpaged pool not available
0000    107 ;                 in SCS$DIRECTORY.  Add SCS$GA_EXISTS to start CONFIGURE
0000    108 ;                 process.
0000    109 ;
0000    110 ;        V03-020 MSH0006         Maryann Hinden          18-May-1983
0000    111 ;                 Put nodename in process poller mailbox.
0000    112 ;
0000    113 ;        V03-019 ROW0184         Ralph O. Weber          19-APR-1983
```

```
                                J 1
0000    114 ;                       Change order of wait queue searching in SCS$LKP_MSGWAIT so
0000    115 ;                       that message buffer wait queue is scanned first.  This is done
0000    116 ;                       because the way PADRIVER enters CDRPs into to the send credit
0000    117 ;                       and message buffer wait queues makes the message buffer wait
0000    118 ;                       queue is likely to contain older (longer waiting) CDRPs.  The
0000    119 ;                       connection manager needs t is additional level of age ordering
0000    120 ;                       to help provide sequential delivery of messages.
0000    121 ;
0000    122 ;               V03-018 MSH0005         Maryann Hinden          19-Apr-1983
0000    123 ;                       Use only one CDRP per connection.
0000    124 ;
0000    125 ;               V03-017 JWH0217         Jeffrey W. Horn         14-Apr-1983
0000    126 ;                       Change second SLV table to be generated with SLVTAB macro.
0000    127 ;
0000    128 ;               V03-016 MSH0004         Maryann Hinden          08-Apr-1983
0000    129 ;                       Changes to support HSC. Various bug fixes.
0000    130 ;
0000    131 ;               V03-015 JWH0211         Jeffrey W. Horn         13-Apr-1983
0000    132 ;                       Use SLVTAB macro to produce loadable-code prologue.
0000    133 ;
0000    134 ;               V03-014 MSH0003         Maryann Hinden          01-Apr-1983
0000    135 ;                       Move location of MY_PROC.
0000    136 ;
0000    137 ;               V03-013 MSH0002         Maryann Hinden          24-Mar-1983
0000    138 ;
0000    139 ;--
0000    140
```

```
                                K 1
0000    142          .SBTTL  DECLARATIONS
0000    143 ;
0000    144 ; INCLUDE FILES
0000    145 ;
0000    146          .nocross
0000    147          .IF     DF,PRMSW                ; For linkage with SCS
0000    148          $CCBDEF
0000    149          $CDLDEF
0000    150          $CDRPDEF
0000    151          $CDTDEF
0000    152          $CRBDEF
0000    153          $DDBDEF
0000    154          $DDTDEF
0000    155          $DYNDEF
0000    156          $IPLDEF
0000    157          $PBDEF
0000    158          $PBODEF
0000    159          $PDTDEF
0000    160          $PRCPOLDEF
0000    161          $RDDEF
0000    162          $RDTDEF
0000    163          .IFTF                           ; Always need the following definition
0000    164          $SBDEF
0000    165          .IFT
0000    166          $SBODEF
0000    167          $SDIRDEF
0000    168          $SPNBDEF
0000    169          $SPPBDEF
0000    170          $SSDEF
0000    171          $TQEDEF
0000    172          $UCBDEF
0000    173          .ENDC
0000    174          .cross
0000    175
0000    176 ;
0000    177 ; Local macros
0000    178 ;
0000    179          .MACRO  LIST_HEAD,ENTRY
0000    180          .ALIGN  LONG
0000    181 ENTRY:: .LONG   ENTRY
0000    182          .LONG   ENTRY
0000    183          .ENDM
0000    184
0000    185          .MACRO  POINTER,ENTRY
0000    186          .ALIGN  LONG
0000    187 ENTRY:: .LONG   0
0000    188          .ENDM
0000    189
0000    190 ;
0000    191 ; Misc labels
0000    192 ;
0000    193          .IF     NDF,PRMSW               ; For linkage with SYS.EXE
0000    194          .PSECT  $$$$00,LONG
0000    195
0000    196 ;
0000    197 ; Quadword linked list heads
0000    198 ;
```

```
                                                L 1
0000    199
0000    200 SCS$GQ_CONFIG::
0000    201         .LONG   SCS$GA_LOCALSB
0000    202         .LONG   SCS$GA_LOCALSB
0000    203         LIST_HEAD SCS$GQ_DIRECT
0000    204         LIST_HEAD SCS$GQ_POLL           ; List of SPPB's giving process names
0000    205
0000    206 ;
0000    207 ; Longword pointers
0000    208 ;
0000    209
0000    210         POINTER SCS$GL_BDT
0000    211         POINTER SCS$GL_CDL
0000    212         POINTER SCS$GL_RDT
0000    213         POINTER SCS$GL_MCLEN
0000    214         POINTER SCS$GL_MCADR
0000    215         POINTER SCS$GL_MSCP             ; Start of MSCP server process
0000    216         POINTER SCS$GL_PDT              ; Singly link list of PDT's
0000    217
0000    218 ;
0000    219 ; Process poller static data
0000    220 ;
0000    221
0000    222 SCS$GA_DFLTMSK::                        ; Mask of processes to enable
0000    223         .BLKB   SB$S_ENBMSK             ;       in new systems that appear
0000    224 SCS$GW_NEXTBIT::                        ; Next bit available for assignment
0000    225         .WORD   0
0000    226
0000    227 SCS$GA_EXISTS:: .LONG   0               ; Flag to indicate presence of SCS
0000    228
0000    229 DO_RSB:
0000    230         CLRL    R0                      ; Failure status
0000    231         RSB                             ; Ignore call to unloaded routine
0000    232
0000    233         .ALIGN  LONG
0000    234 ;
0000    235 ; Vector list
0000    236 ;
0000    237
0000    238 SCS$AL_LOAVEC::
0000    239         .IFF
00000000    240         .PSECT  $$$000,LONG
0000    241 SCS_START:
0000    242         SLVTAB  END     = SCS_END, -
0000    243                 INITRTN = INIT_BEGIN, -
0000    244                 SUBTYP  = DYN$C_LC_SCS, -
0000    245                 FACILITY= <SCS_Loadable>
0024    246
0024    247         .ENDC
0024    248
0024    249         LOADVEC SCS$ACCEPT
0029    250         LOADVEC SCS$ALLOC_CDT
002E    251         LOADVEC SCS$ALLOC_RSPID
0033    252         LOADVEC SCS$CONFIG_PTH
0038    253         LOADVEC SCS$CONFIG_SYS
003D    254         LOADVEC SCS$CONNECT
0042    255         LOADVEC SCS$DEALL_CDT
```

```
                     0047   256          LOADVEC SCS$DEALL_RSPID
                     004C   257          LOADVEC SCS$DISCONNECT
                     0051   258          LOADVEC SCS$ENTER
                     0056   259          LOADVEC SCS$LISTEN
                     005B   260          LOADVEC SCS$LOCLOOKUP
                     0060   261          LOADVEC SCS$REMOVE
                     0065   262          LOADVEC SCS$RESUMEWAITR
                     006A   263          LOADVEC SCS$UNSTALLUCB
                     006F   264          LOADVEC SCS$LKP_RDTCDRP
                     0074   265          LOADVEC SCS$LKP_RDTWAIT
                     0079   266          LOADVEC SCS$RECYL_RSPID
                     007E   267          LOADVEC SCS$FIND_RDTE
                     0083   268          LOADVEC SCS$LKP_MSGWAIT
                     0088   269          LOADVEC SCS$DIR_LOOKUP          ; Search for processes on remote node
                     008D   270          LOADVEC SCS$NEW_SB,,DO_RSB      ; Called when an SB is created/reused
                     0092   271          LOADVEC SCS$POLL_PROC,,DO_RSB   ; Declare a process name to the poller
                     0097   272          LOADVEC SCS$POLL_MODE           ; Enable/Disable polling for a process
                     009C   273          LOADVEC SCS$POLL_MBX,,DO_RSB    ; Declare a mailbox to receive poller notifi
                     00A1   274          LOADVEC SCS$CANCEL_MBX          ; Cancel notifications to a mailbox
                     00A6   275          LOADVEC SCS$SHUTDOWN,,DO_RSB    ; Shut down all SCS virtual circuits
                     00AB   276
                     00AB   277 ;
                     00AB   278 ; End of list
                     00AB   279 ;
                     00AB   280          .IF     NDF,PRMSW
                     00AB   281          .NLIST                          ; Don't clutter vector listing
                     00AB   282          .IFF                            ; For linking with SCS
                 FFFFFFFF    00AB   283          .LONG   -1                      ; End of vectors
                     00AF   284
                     00AF   285 ;
                     00AF   286 ; The SCA process name poller makes the following assumptions
                     00AF   287 ;
                     00AF   288          ASSUME  IPL$_TIMER  EQ  IPL$_SCS
                     00AF   289          ASSUME  IPL$_SYNCH  EQ  IPL$_SCS
                     00AF   290
```

```
                        00AF   292          .SBTTL  INITIALIZATION CODE
                        00AF   293 ;+
                        00AF   294 ; This code is called once upon the loading of SCS.  It initializes any
                        00AF   295 ; piece of SCS that needs it.  It then deallocated itself to pool.
                        00AF   296 ;-
                        00AF   297
                        00AF   298 INIT_BEGIN:
              5E   0C   C2  00AF   299          SUBL    #12, SP                 ; Get some scratch area
              56   5E   D0  00B2   300          MOVL    SP, R6                  ; Set a pointer
                 007F   30  00B5   301          BSBW    SEQ_NO                  ; Get a sequence number
                 0144   30  00B8   302          BSBW    INIT_RDT                ; Do the Request Desc
                58 50   E9  00BB   303          BLBC    R0, 10$
                        00BE   304 ;        BSBB    SEQ_NO                  ; Get a sequence number
 01 A6        10F3 8F   A0  00BE   305          ADDW2   #^X10F3, 1(R6)          ; Permute the low byte
                 0092   30  00C4   306          BSBW    INIT_CDL                ; Do the Connect Desc
              5E   0C   C0  00C7   307          ADDL    #12, SP                 ; Release the scratch area
                49 50   E9  00CA   308          BLBC    R0, 10$                 ; Check for error
                        00CD   309          LISTEN  MSGADR=W^SCS$DIRECTORY,-
                        00CD   310                  ERRADR=W^LISTENERR,-
                        00CD   311                  LPRNAM=B^DIRNAME,-
                        00CD   312                  PRINFO=B^DIRINFO         ; Put our name in directory-leave
                        00E4   313                                           ; (MUST BE AFTER INIT OF CDT)
                2F 50   E9  00E4   314          BLBC    R0, 10$                 ; Check for error
                 0164   30  00E7   315          BSBW    POLLER_INIT             ; Initialize process name poller
                29 50   E9  00EA   316          BLBC    R0,10$                  ; Branch on error
           50   FF0F CF   DE  00ED   317          MOVAL   SCS_START, R0           ; Set R0--> start of code
00000000'GF     50   D0  00F2   318          MOVL    R0,G^SCS$GA_EXISTS      ; Indicate presence of SCS
                        00F9   319                                           ; (flag to start CONFIGURE process)
           51   02B0'8F   3C  00F9   320          MOVZWL  #<CODE_HEADER-SCS_START>,R1 ; Set the size of just the init code
           FF05 CF   51   A3  00FE   321          SUBW3   R1,SCS_START+8,-        ; Compute size of remaining code
                 02B8'CF        0103   322                  W^CODE_HEADER+8         ;
        53   00000004'GF   9E  0106   323          MOVAB   G^EXE$GL_NONPAGED+4,R3  ; List head
             00000000'GF   16  010D   324          JSB     G^EXE$DEALLOCATE        ; Just the smile on the Cheshire cat...
                  50   01   D0  0113   325          MOVL    #1,R0                   ; Set success
                            05  0116   326 10$:     RSB
                        0117   327
                        0117   328 ;
52 4F 54 43 45 52 49 44 24 53 43 53  0117   329 DIRNAME:.ASCII  /SCS$DIRECTORY   /
                     20 20 20 59     0123
70 20 2C 79 74 69 63 20 74 61 68 57  0127   330 DIRINFO:.ASCII  /What city, plez?/
                     3F 7A 65 6C     0133
                        0137   331 ;
                        0137   332 ; Calculate a random sequence number from the system time
                        0137   333 ;
                        0137   334 SEQ_NO:
        66   00000002'GF   D0  0137   335          MOVL    G^EXE$GQ_TODCBASE+2,(R6); Get middle bits of system time
                  50   66   3C  013E   336          MOVZWL  (R6), R0                ; Get low 16 bits
                51   02 A6   3C  0141   337          MOVZWL  2(R6), R1               ; Get another 16 bits
                   51   50   C4  0145   338          MULL2   R0, R1                  ; Multiply them
                50   01 A6   3C  0148   339          MOVZWL  1(R6), R0               ; Add in a fudge
              66   51   50   C1  014C   340          ADDL3   R0, R1, (R6)            ; Store away for later
                01 A6   B5  0150   341          TSTW    1(R6)                   ; Check for zero
                   03   12  0153   342          BNEQ    10$                     ; Not so, okay
                01 A6   B6  0155   343          INCW    1(R6)                   ; Yes, fix it
                            05  0158   344 10$:     RSB
                        0159   345 ;
                        0159   346 ; Allocate and initialize the Connection Descriptor List
```

```
                        0159   347 ;
                        0159   348 INIT_CDL:
     54   00000000'9F   3C  0159   349          MOVZWL  @#SCS$GW_CDTCNT, R4     ; Pick up number of CDT's to alloc
     000002D4'EF   54   A0  0160   350          ADDW    R4,SCS$GW_CDTMAX        ; Add sysgened # CDT's to # extension
                        0167   351                                          ;  CDT's allowed
          57   00A0 8F   3C  0167   352          MOVZWL  #<<CDT$C_LENGTH+15>&^C15>,R7  ; Get rounded CDT size
          51   54   57   C5  016C   353          MULL3   R7, R4, R1              ; Get total amount of space
     54   00000064'8F   C0  0170   354          ADDL    #SCS$K_CDTEXT,R4        ; Get # CDL pointers to CDTs needed
               55   10   D0  0177   355          MOVL    #CDL$C_LENGTH, R5       ; Fixed portion of CDL
          55   0F A544   DE  017A   356          MOVAL   15(R5)[R4], R5          ; Compute total CDL size with
               55   0F   CA  017F   357          BICL    #15, R5                 ;  appropriate rounding
               51   55   C0  0182   358          ADDL    R5, R1                  ; Compute total pool needed
                  06AE   30  0185   359          BSBW    SCS_ALONONPAGED         ; Allocate it
                 73 50   E9  0188   360          BLBC    R0, 20$                 ; Error - no memory
                        018B   361 ;
                        018B   362 ; R1  =  Size of allocated block
                        018B   363 ; R2 --> Al ocated block
                        018B   364 ; R4  =  #CDT's to allocate now + # extension CDTs allowed
                        018B   365 ; R5  =  Size of CDL (Rounded up)
                        018B   366 ; R7  =  Size of CDT(Rounded up)
                        018B   367 ;
     54   00000064'8F   C2  018B   368          SUBL    #SCS$K_CDTEXT,R4        ; Back off R4 to # CDT's to allocate now
                  3F   BB  0192   369          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save reg
62   51   00   6E   00   2C  0194   370          MOVC5   #0, (SP), #0, R1, (R2)  ; Clear it all out
                  3F   BA  019A   371          POPR    #^M<R0,R1,R2,R3,R4,R5>
   00000000'9F   10 A2   DE  019C   372          MOVAL   CDL$C_LENGTH(R2), -
                        01A4   373                  @#SCS$GL_CDL                ; Set pointer to CDL in system space
          51   6245   9E  01A4   374          MOVAB   (R2)[R5], R1            ; Get pointer to first CDT
            62   54   D0  01A8   375          MOVL    R4, (R2)                ; Set # CDTs initially
                 82   D7  01AB   376          DECL    (R2)+                   ; Convert # CDTs to max initl index
            82   51   D0  01AD   377          MOVL    R1, (R2)+               ; Set first free CDT
            82   55   B0  01B0   378          MOVW    R5, (R2)+               ; Set size of CDL
      82    0160 8F   B0  01B3   379          MOVW    #DYN$C_SCS!-
                        01B8   380                  <DYN$C_SCS_CDL@8>,(R2)+ ; Set type and subtype
                 82   D4  01B8   381          CLRL    (R2)+                   ; Clear reserved area
                 55   D4  01BA   382          CLRL    R5                      ; Initialize index
            82   51   D0  01BC   383 10$:     MOVL    R1, (R2)+               ; Set address
        53     6147   9E  01BF   384          MOVAB   (R1)[R7], R3            ; Get address of next CDT
            61   53   D0  01C3   385          MOVL    R3, CDT$L_LINK R1)      ; Set it into link field
      08 A1    57   B0  01C6   386          MOVW    R7, CDT$W_SIZE(R1)      ; Set size field
0A A1         0260 8F   B0  01CA   387          MOVW    #DYN$C_SCS!-
                        01D0   388                  <DYN$C_SCS_CDT@8>, -
                        01D0   389                   CDT$B_TYPE(R1)         ; Set type and subtype
      18 A1    55   B0  01D0   390          MOVW    R5, CDT$L_LCONID(R1)    ; Set index piece of CONID
   1A A1    01 A6   B0  01D4   391          MOVW    1(R6),CDT$L_LCONID+2(R1); Set sequence piece of CONID
            28 A1   D4  01D9   392          CLRL    CDT$W_STATE(R1)         ; Set state to virgin
   30 A1    30 A1   9E  01DC   393          MOVAB   CDT$L_WAITQFL(R1), -
                        01E1   394                  CDT$L_WAITQFL(R1)       ; Set up initial cond
   34 A1    30 A1   9E  01E1   395          MOVAB   CDT$L_WAITQFL(R1), -
                        01E6   396                  CDT$L_WAITQBL(R1)       ; Set up initial cond
   38 A1    38 A1   9E  01E6   397          MOVAB   CDT$L_CRWAITQFL(R1), -
                        01EB   398                  CDT$L_CRWAITQFL(R1)     ; Set up initial cond
   3C A1    38 A1   9E  01EB   399          MOVAB   CDT$L_CRWAITQFL(R1), -
                        01F0   400                  CDT$L_CRWAITQBL(R1)     ; Set up initial cond
          51   53   D0  01F0   401          MOVL    R3, R1                  ; Get next CDT
      C5  55   54   F2  01F3   402          AOBLSS  R4, R5,10$              ; Loop until all done
            FF60 C1   D4  01F7   403          CLRL    -CDT$C_LENGTH(R1)       ; Zero out final next pointer
```

```
                50   01    D0  01FB   404          MOVL    #1, R0
                           05  01FE   405 20$:     RSB
                               01FF   406 ;
                               01FF   407 ; Allocate and initialize the Request Descriptor Table
                               01FF   408 ;
                               01FF   409 INIT_RDT:
                               01FF   410          ASSUME  RD$C_LENGTH EQ 8
        54   00000000'9F    3C  01FF   411          MOVZWL  @#SCS$GW_RDTCNT, R4     ; Pick up number of RDTE's
                  51   18    D0  0206   412          MOVL    #RDT$C_LENGTH, R1       ; Fixed portion
                 51   6144    7E  0209   413          MOVAQ   (R1)[R4], R1            ; Total space needed
                      0626    30  020D   414          BSBW    SCS_ALONONPAGED         ; Allocate it
                    3A   50    E9  0210   415          BLBC    R0, 20$                 ; No space, error
                53   18 A2    9E  0213   416          MOVAB   RDT$C_LENGTH(R2), R3    ; Set past RDT
        00000000'9F      53    D0  0217   417          MOVL    R3, @#SCS$GL_RDT        ; Set pointer to RDT in system space
                  55     52    D0  021E   418          MOVL    R2, R5                  ; Save it
                  82     55    D0  0221   419          MOVL    R5, (R2)+               ; Set initial condition
                  82     55    D0  0224   420          MOVL    R5, (R2)+               ; Ditto
                  82     51    B0  0227   421          MOVW    R1, (R2)+               ; Set size field
              82   0660   8F    B0  022A   422          MOVW    #DYN$C_SCS!-
                                       022F   423                  <DYN$C_SCS_RDT@8>,(R2)+ ; Set type & subtype
                  82     53    7D  022F   424          MOVQ    R3, (R2)+               ; Set first free RDTE & # RDTEs
                    FC   A2    D7  0232   425          DECL    -4(R2)                  ; Convert # RDTEs to maximum index
                           82    D4  0235   426          CLRL    (R2)+                   ; Clear out reserved field
                53   08   A3    9E  0237   427          MOVAB   RD$C_LENGTH(R3), R3     ; Set R3 point to next RDTE
                  82     83    7E  023B   428 10$:     MOVAQ   (R3)+, (R2)+            ; Set next free RDTE link
                           82    B4  023E   429          CLRW    (R2)+                   ; Set state to FREE
                82   01   A6    B0  0240   430          MOVW    1(R6), (R2)+            ; Set in sequence number
                    F4   54    F5  0244   431          SOBGTR  R4, 10$                 ; Loop u til done
                    F8   A2    D4  0247   432          CLRL    -RD$C_LENGTH(R2)        ; Zero out final next pointer
                  50     01    D0  024A   433          MOVL    #1, R0
                               05  024D   434 20$:     RSB
                                   024E   435 ;
                                   024E   436 ; Initialize the process name poller
                                   024E   437 ;
                                   024E   438 POLLER_INIT:
        53   00000000'GF    DE    024E   439          MOVAL   G^SCS$GQ_CONFIG,R3      ; Address of system block listhead
                  52   63    D0  0255   440          MOVL    (R3),R2                 ; Address of first SB
                  53   52    D1  0258   441 10$:     CMPL    R2,R3                   ; Back to listhead?
                       08    13  025B   442          BEQL    20$                     ; Branch if done
                     06F1    30  025D   443          BSBW    SCS$NEW_SB              ; New system to poll
                  52   62    D0  0260   444          MOVL    SB$L_FLINK(R2),R2       ; Chain to next SB
                       F3    11  0263   445          BRB     10$                     ; Iterate over known systems
                                   0265   446 ;
                                   0265   447 ; Design note:
                                   0265   448 ;
                                   0265   449 ;       The following could be done in less code by having EXE$TIMEOUT make
                                   0265   450 ;       the call.  The tradeoff is that if EXE$TIMEOUT calls the poller,
                                   0265   451 ;       then there is a JSB/RSB overhead every second when not in an SCS
                                   0265   452 ;       environment.  In an SCS environment, however, a TQE and the
                                   0265   453 ;       associated queue management is avoided each second.
                                   0265   454 ;
                  51   30    3C    0265   455 20$:     MOVZWL  #TQE$K_LENGTH,R1        ; Size of a TQE
                     05CB    30  0268   456          BSBW    SCS_ALONONPAGED         ; Allocate a TQE
                    32   50    E9  026B   457          BLBC    R0,30$                  ; Exit on error
              08 A2      51    B0  026E   458          MOVW    R1,TQE$W_SIZE(R2)       ; Store size
              0A A2      0F    90  0272   459          MOVB    #DYN$C_TQE,TQE$B_TYPE(R2)       ; Store type
              0B A2      05    90  0276   460          MOVB    #TQE$C_SSREPT,TQE$B_RQTYPE(R2)  ; Store type of timer queue entry
```

```
                 0C A2    0A05'CF    9E  027A   461          MOVAB   W^START_POLL,TQE$L_FPC(R2)      ; Address of timeout routine
                                     0280   462          ASSUME  TQE$L_FR3+4,EQ,TQE$L_FR4
                       10 A2    7C  0280   463          CLRQ    TQE$L_FR3(R2)             ; Zero R3 and R4
 20 A2    00    01    00989680 8F    7A  0283   464          EMUL    #10*1000*1000,#1,#0, -    ; Delta time = 1 second
                                     028D   465                  TQE$Q_DELTA(R2)
                    55    52    D0  028D   466          MOVL    R2,R5                     ; Address of TQE
                 50    00000000'GF    7D  0290   467          MOVQ    G^EXE$GQ_SYSTIME,R0       ; Immediate timeout
                       00000000'GF    16  0297   468          JSB     G^EXE$INSTIMQ             ; Link into timer queue
                          50    01    D0  029D   469          MOVL    S^#SS$_NORMAL,R0          ; All is well
                                      05  02A0   470 30$:      RSB
                                          02A1   471
                             000002A1     02A1   472          LC=.
                             000002B0     02A1   473          .=<LC+15>&-16                     ; Align on 16 byte boundary
                                          02B0   474 ;
                                          02B0   475 ; Put a SLV prologue on this piece.
                                          02B0   476 ;
                                          02B0   477 CODE_HEADER:
                                          02B0   478          SLVTAB  END     = SCS_END, -
                                          02B0   479                  SUBTYP  = DYN$C_LC_SCS, -
                                          02B0   480                  FACILITY= <SCS_Loadable>
                                          02D4   481
                                          02D4   482 ;
                                          02D4   483 ; Some SCS loadable data:
                                          02D4   484 ;
                                          02D4   485
                             00000064     02D4   486 SCS$K_CDTEXT == 100                       ; Allow # CDT's to be extended
                                          02D4   487                                          ;  by 100 beyond sysgened value
                                          02D4   488
                                          02D4   489 SCS$GW_CDTMAX::
                                          02D4   490
                       0063               02D4   491          .WORD   SCS$K_CDTEXT-1            ; This will be the maximum CDT
                                          02D6   492                                          ;  index ever allowed in the CDL.
                                          02D6   493                                          ;  It = 100-1 + sysgened # CDT's
                                          02D6   494                                          ;  to be filled in during SCS init
```

```
                     02D6   496          .SBTTL  SCS$ACCEPT
                     02D6   497 ;++
                     02D6   498 ;
                     02D6   499 ; FUNCTIONAL DESCRIPTION:
                     02D6   500 ;
                     02D6   501 ;
                     02D6   502 ; CALLING SEQUENCE:
                     02D6   503 ;
                     02D6   504 ;       BSBW    SCS$ACCEPT
                     02D6   505 ;
                     02D6   506 ; INPUTS:
                     02D6   507 ;
                     02D6   508 ;       R3 = Address of listening CDT
                     02D6   509 ;        0(SP) = Input address                   MSGADR
                     02D6   510 ;        4(SP) = [Datagram input address]        DGADR
                     02D6   511 ;        8(SP) = Error address                   ERRADR
                     02D6   512 ;       12(SP) = Initial credit value           INITCR
                     02D6   513 ;       14(SP) = Minimum send credit value      MINSCR
                     02D6   514 ;       16(SP) = Initial DG credit value        INITDG
                     02D6   515 ;       18(SP) = [Block transfer priority]      BLKPRI
                     02D6   516 ;       20(SP) = Address of connect data        CONDAT
                     02D6   517 ;       24(SP) = Address of auxilary structure  AUXSTR
                     02D6   518 ;       28(SP) = [Bad response packet address]  BADRSP
                     02D6   519 ;       32(SP) = Return address
                     02D6   520 ;
                     02D6   521 ; OUTPUTS:
                     02D6   522 ;
                     02D6   523 ;       R0 = Status
                     02D6   524 ;       R2 = Address of listening CDT
                     02D6   525 ;       R3 = Address of CDT allocated
                     02D6   526 ;       R4 = Address of PDT for listening
                     02D6   527 ;       R1,R2 Destroyed
                     02D6   528 ;       R5   Preserved
                     02D6   529 ;
                     02D6   530 ;--
                     02D6   531
                     02D6   532 SCS$ACCEPT:
   52    53    D0    02D6   533          MOVL    R3, R2                          ; Transfer listen CDT ptr
         18    10    02D9   534          BSBB    SCS$ALLOC_CDT                   ; Allocate a CDT
     11  50    E9    02DB   535          BLBC    R0, 10$                         ; None, clean up and leave
                     02DE   536          ASSUME  CDT$L_MSGINPUT+4 EQ CDT$L_DGINPUT
   63    8E    7D    02DE   537          MOVQ    (SP)+,CDT$L_MSGINPUT(R3); Get both MSGINPUT and DGINPUT
     0C  A3 8ED0    02E1   538          POPL    CDT$L_ERRADDR(R3)               ; Set address of error routine
54   10  A2    D0    02E5   539          MOVL    CDT$L_PDT(R2), R4               ; Pick up PDT addr from listen
   51    0C    9A    02E9   540          MOVZBL  #PDT$C_ACCEPT, R1               ; Offset in PDT to use
      0266    31    02EC   541          BRW     PDT_JMP                         ; Finish in port dependent code
                     02EF   542
   5E    20    C0    02EF   543 10$:     ADDL    #8*4, SP                        ; Clean args off stack
               05    02F2   544          RSB
```

```
                     546          .SBTTL  SCS$ALLOC_CDT
                     547 ;++
                     548 ;
                     549 ; FUNCTIONAL DESCRIPTION:
                     550 ;
                     551 ;       Attempt to remove a CDT from the free list.  If no more
                     552 ;       are available, and the CDL has room for an extension CDT,
                     553 ;       then allocate one CDT from pool.  Put new CDT address in CDT,
                     554 ;       step current maximum index in CDL header, init CDT to closed
                     555 ;       state, and give it a local connection ID.
                     556 ;
                     557 ;       In allocated CDT, init wait queues, zero most of the rest of
                     558 ;       the CDT, and return to caller.
                     559 ;
                     560 ; CALLING SEQUENCE:
                     561 ;
                     562 ;       BSBW    SCS$ALLOC_CDT
                     563 ;
                     564 ; INPUTS:
                     565 ;
                     566 ;       NONE
                     567 ;
                     568 ; OUTPUTS:
                     569 ;
                     570 ;       R0 = SS$_NORMAL if success
                     571 ;          = SS$_INSFCDT if no CDT's available
                     572 ;       R1 = CONID
                     573 ;       R3 = Pointer to CDT if successful
                     574 ;          = 0 otherwize
                     575 ;       R2,R4,R5 preserved
                     576 ;
                     577 ;--
                     578
                     579 SCS$ALLOC_CDT:
51   00000000'9F    D0   02F3   580          MOVL    @#SCS$GL_CDL, R1          ; Pick up pointer to list
     53    F4 A1    D0   02FA   581          MOVL    CDL$L_FREECDT(R1), R3     ; Find first free one
           42       12   02FE   582          BNEQ    20$                       ; Branch if got one
   51    00A0 8F    3C   0300   583          MOVZWL  #CDT$C_LENGTH,R1          ; Get size of a CDT
           52       DD   0305   584          PUSHL   R2                        ; Save caller's register
         052C       30   0307   585          BSBW    SCS_ALONONPAGED           ; Allocate one from pool
     53    52       D0   030A   586          MOVL    R2,R3                     ; Copy assumed new CDT addr
           52 8ED0    D0   030D   587          POPL    R2                        ; Restore caller's R2
        5D 50       E9   0310   588          BLBC    R0,NO_CDT                 ; Branch if didn't get CDT from pool
   08 A3    51       B0   0313   589          MOVW    R1,CDT$W_SIZE(R3)         ; Set CDT size,
                     B0   0317   590          MOVW    #DYN$C_SCS!-              ;  structure type,
                     0318   591                      <DYN$C_SCS_CDT@8>,-       ;  and subtype
 0A A3   0260 8F          0318   592                      CDT$B_TYPE(R3)            ;
         28 A3    D4   031D   593          CLRL    CDT$W_STATE(R3)           ; Set CDT state = closed
51   00000000'9F    D0   0320   594          MOVL    @#SCS$GL_CDL,R1           ; Get start of CDT list
     50    61       D0   0327   595          MOVL    (R1),R0                   ; Get 1st CDT address
   50    18 A0      D0   032A   596          MOVL    CDT$L_LCONID(R0),R0       ; Get 1st CDT's local CONID
                     032E   597                                                ;  in order to capture its sequence #
         F0 A1    B6   032E   598          INCW    CDL$W_MAXCONIDX(R1)       ; Step max legal CONID index in CDL
   50    F0 A1    B0   0331   599          MOVW    CDL$W_MAXCONIDX(R1),R0    ; Combine seq # and index
   18 A3    50       D0   0335   600          MOVL    R0,CDT$L_LCONID(R3)       ;  = new CDT's connection ID
      50    50       3C   0339   601          MOVZWL  R0,R0                     ; Isolate index
    6140    53       D0   033C   602          MOVL    R3,(R1)[R0]               ; Save new CDT addr in CDL
```

```
                                04   11  0340   603          BRB     30$                         ; Join common CDT init
                                                0342   604
                            F4  A1  63   D0  0342   605 20$:    MOVL    CDT$L_LINK(R3), -
                                                0346   606             CDL$L_FREECDT(R1)       ; Set pointer to next free CDT
                                                0346   607
                                3C   BB  0346   608 30$:    PUSHR   #^M<R2,R3,R4,R5>            ; Save registers for MOVC
                        00  00  8F   00  2C  0348   609          MOVC5   #0,#0,#0,-              ; Zero CDT from longwd after
                                0084 8F      034D   610                  #<CDT$C_LENGTH-CDT$L_LCONID-4>,-
                                  1C A3      0350   611                  CDT$L_LCONID+4(R3)      ;  local CONID to end
                                3C   BA  0352   612          POPR    #^M<R2,R3,R4,R5>            ; Restore destroyed registers
                            51    18 A3  D0  0354   613          MOVL    CDT$L_LCONID(R3), R1    ; Get conid for return to caller
                        30  A3    30 A3  9E  0358   614          MOVAB   CDT$L_WAITQFL(R3), -
                                                035D   615                  CDT$L_WAITQFL(R3)       ; Set up initial cond
                        34  A3    30 A3  9E  035D   616          MOVAB   CDT$L_WAITQFL(R3), -
                                                0362   617                  CDT$L_WAITQBL(R3)       ; Set up initial cond
                        38  A3    38 A3  9E  0362   618          MOVAB   CDT$L_CRWAITQFL(R3), -
                                                0367   619                  CDT$L_CRWAITQFL(R3)     ; Set up initial cond
                        3C  A3    38 A3  .E  0367   620          MOVAB   CDT$L_CRWAITQFL(R3), -
                                                036C   621                  CDT$L_CRWAITQBL(R3)     ; Set up initial cond
                                50   01  D0  036C   622          MOVL    #1,R0                   ; Set success
                                         05  036F   623          RSB
                                             0370   624
                                             0370   625 NO_CDT:
                             50   21AC 8F   3C  0370   626          MOVZWL  #SS$_INSFCDT,R0         ; Set error status
                                         05  0375   627          RSB                             ;  and return to caller
```

```
                     0376   629          .SBTTL  SCS$ALLOC_RSPID
                     0376   630 ;++
                     0376   631 ;
                     0376   632 ; FUNCTIONAL DESCRIPTION:
                     0376   633 ;
                     0376   634 ;
                     0376   635 ; CALLING SEQUENCE:
                     0376   636 ;
                     0376   637 ;       BSBW    SCS$ALLOC_RSPID
                     0376   638 ;
                     0376   639 ; INPUTS:
                     0376   640 ;
                     0376   641 ;       R5 = Pointer to context block (usually CDRP)
                     0376   642 ;
                     0376   643 ; OUTPUTS:
                     0376   644 ;
                     0376   645 ;       R1 = RSPID
                     0376   646 ;       R2 = Pointer to RDTE
                     0376   647 ;       R0,R3,R4 preserved
                     0376   648 ;
                     0376   649 ;       RSPID stored in CDRP    (* What if not a CDRP input??? *)
                     0376   650 ;
                     0376   651 ; SIDE EFFECTS:
                     0376   652 ;
                     0376   653 ;       If no RSPID available, process is put on wait queue, and control
                     0376   654 ;       returned to caller's caller.
                     0376   655 ;
                     0376   656 ;--
                     0376   657
                     0376   658 SCS$ALLOC_RSPID:
51   00000000'9F    D0  0376   659          MOVL    @#SCS$GL_RDT, R1        ; Pick up list head
     52    F4 A1    D0  037D   660          MOVL    RDT$L_FREERD(R1), R2    ; Find first free one
                1F    13  0381   661          BEQL    10$                     ; None available
     F4 A1    62    D0  0383   662          MOVL    RD$L_LINK(R2), -
                          0387   663                  RDT$L_FREERD(R1)        ; Update free list
   51   52    51    C3  0387   664          SUBL3   R1, R2, R1              ; Form offset
   51   51    0D    78  038B   665          ASHL    #13, R1, R1             ; Turn into slot number
     51   06 A2    B0  038F   666          MOVW    RD$W_SEQNUM(R2), R1     ; Insert sequence number
   51   51    10    9C  0393   667          ROTL    #16, R1, R1             ; Reverse to correct order
     20 A5    51    D0  0397   668          MOVL    R1, CDRP$L_RSPID(R5)    ; Store in contex block
        62    55    D0  039B   669          MOVL    R5, RD$L_CDRP(R2)       ; Fill in context block addr
           04 A2    B6  039E   670          INCW    RD$W_STATE(R2)          ; Set state to IN-USE
                    05  03A1   671          RSB
                        03A2   672
     10 A5    53    7D  03A2   673 10$:     MOVQ    R3, CDRP$L_FR3(R5)      ; Save R3,R4 in context block
           0C A5 8ED0  03A6   674          POPL    CDRP$L_FPC(R5)          ; ... and PC
     EC B1    65    0E  03AA   675          INSQUE  CDRP$L_FQFL(R5), -      ; Insert this context block
                        03AE   676                  @RDT$L_WAITBL(R1)       ;   on RDT wait queue
     50   28 A5    D0  03AE   677          MOVL    CDRP$L_RWCPTR(R5), R0   ; Pick up any counter
                02    13  03B2   678          BEQL    20$                     ; None
                60    B6  03B4   679          INCW    (R0)                    ; Bump the number of resources
                    05  03B6   680 20$:     RSB                             ; Return to caller's caller
```

```
03B7   682          .SBTTL  SCS$CONFIG_PTH
03B7   683 ;++
03B7   684 ;
03B7   685 ; FUNCTIONAL DESCRIPTION:
03B7   686 ;
03B7   687 ;       CONFIG_PTH searches  the  path  blocks  on  all  system
03B7   688 ;       blocks for one with a matching station address and local
03B7   689 ;       port name.  If none is found, then error is returned in
03B7   690 ;       R0.   Otherwise, most of the path block information
03B7   691 ;       is returned in the caller's output array plus the system
03B7   692 ;       ID of the remote system on this  path  and  the  station
03B7   693 ;       address  of  the  next  path  to this remote system.  If
03B7   694 ;       there are no more paths, the remote station ID  returned
03B7   695 ;       is -1.
03B7   696 ;
03B7   697 ;       A fork process interested in discovering all paths to  a
03B7   698 ;       particular  system  first  issues  a  CONFIG_SYS call to
03B7   699 ;       learn the station addr/local port name  of  the 1st  path
03B7   700 ;       to the system. The fork  process  then  calls  CONFIG_PTH
03B7   701 ;       repeatedly  until  a  next  station  address  of  -1  is
03B7   702 ;       returned.
03B7   703 ;
03B7   704 ; CALLING SEQUENCE:
03B7   705 ;
03B7   706 ;       BSBW    CONFIG_PTH
03B7   707 ;
03B7   708 ; INPUTS:
03B7   709 ;
03B7   710 ;       R1 = Address of station address followed by local port name:
03B7   711 ;
03B7   712 ;            +--------+--------+--------+--------+
03B7   713 ;            |      Remote station addr, l.o.    |
03B7   714 ;            +--------+--------+--------+--------+
03B7   715 ;            |    unused       | Rstation, h.o.  |
03B7   716 ;            +--------+--------+--------+--------+
03B7   717 ;            |     Local port name, e.g., PAA0   |
03B7   718 ;            +--------+--------+--------+--------+
03B7   719 ;
03B7   720 ;       R2 = Address of output buffer to return path information
03B7   721 ;               0 if no output desired
03B7   722 ;
03B7   723 ; OUTPUTS:
03B7   724 ;
03B7   725 ;       R0 = SS$_NOSUCHNODE if failure to find specified path
03B7   726 ;          = 1 if success
03B7   727 ;       R1 = Address of path block found (if R0 = success)
03B7   728 ;       R2,R3,R4,R5 preserved
03B7   729 ;
03B7   730 ;--
03B7   731
03B7   732 ;
03B7   733 ; Path block and output array adjacency assumptions:
03B7   734 ;
03B7   735
03B7   736 ASSUME  PB$B_RSTATION+6   EQ  PB$W_STATE
03B7   737 ASSUME  PB$W_STATE+2      EQ  PB$L_RPORT_TYP
03B7   738 ASSUME  PB$L_RPORT_TYP+4 EQ  PB$L_RPORT_REV
```

```
                        03B7   739 ASSUME  PB$L_RPORT_REV+4  EQ  PB$L_RPORT_FCN
                        03B7   740 ASSUME  PB$L_RPORT_FCN+4  EQ  PB$B_RST_PORT
                        03B7   741 ASSUME  PB$B_RST_PORT+1   EQ  PB$B_RSTATE
                        03B7   742 ASSUME  PB$B_RSTATE+1     EQ  PB$W_RETRY
                        03B7   743 ASSUME  PB$W_RETRY+2      EQ  PB$T_LPORT_NAME
                        03B7   744 ASSUME  PB$T_LPORT_NAME+4 EQ  PB$B_CBL_STS
                        03B7   745 ASSUME  PB$B_CBL_STS+1    EQ  PB$B_PO_STS
                        03B7   746 ASSUME  PB$B_PO_STS+1     EQ  PB$B_P1_STS
                        03B7   747
                        03B7   748 ASSUME  PBO$B_RSTATION+6  EQ  PBO$W_STATE
                        03B7   749 ASSUME  PBO$W_STATE+2     EQ  PBO$L_RPORT_TYP
                        03B7   750 ASSUME  PBO$L_RPORT_TYP+4 EQ  PBO$L_RPORT_REV
                        03B7   751 ASSUME  PBO$L_RPORT_REV+4 EQ  PBO$L_RPORT_FCN
                        03B7   752 ASSUME  PBO$L_RPORT_FCN+4 EQ  PBO$B_RST_PORT
                        03B7   753 ASSUME  PBO$B_RST_PORT+1  EQ  PBO$B_RSTATE
                        03B7   754 ASSUME  PBO$B_RSTATE+1    EQ  PBO$W_RETRY
                        03B7   755 ASSUME  PBO$W_RETRY+2     EQ  PBO$T_LPORT_NAME
                        03B7   756 ASSUME  PBO$T_LPORT_NAME+4 EQ PBO$B_CBL_STS
                        03B7   757 ASSUME  PBO$B_CBL_STS+1   EQ  PBO$B_PO_STS
                        03B7   758 ASSUME  PBO$B_PO_STS+1    EQ  PBO$B_P1_STS
                        03B7   759 ASSUME  PBO$B_P1_STS+2    EQ  PBO$B_NXT_RSTAT
                        03B7   760 ASSUME  PBO$B_NXT_RSTAT+8 EQ  PBO$T_NXT_LPORT
                        03B7   761 ASSUME  PBO$T_NXT_LPORT+4 EQ  PBO$B_SYSTEMID
                        03B7   762
                        03B7   763 SCS$CONFIG_PTH:
             1C   BB   03B7   764          PUSHR   #^M<R2,R3,R4>           ; Save registers and output buf. ptr.
 50 00000000'9F   DE   03B9   765          MOVAL   @#SCS$GQ_CONFIG, R0     ; Pick up pointer to database
          52   50   D0   03C0   766          MOVL    R0, R2                  ; Hold starting point
          50   60   D0   03C3   767 10$:     MOVL    (R0), R0                ; Get next block in list
          52   50   D1   03C6   768          CMPL    R0, R2                  ; Back where we started (empty) ?
               61   13   03C9   769          BEQL    50$                     ; Yes, leave now
                        03CB   770 ;
                        03CB   771 ; Got a system block, now search for a path block
                        03CB   772 ;
    53   0C   A0   DE   03CB   773          MOVAL   SB$L_PBFL(R0), R3       ; No, have a sys block - find 1st path
          54   53   D0   03CF   774          MOVL    R3, R4                  ; Hold starting point
          53   63   D0   03D2   775 20$:     MOVL    (R3), R3                ; Get next block in list
          54   53   D1   03D5   776          CMPL    R3, R4                  ; Back where we started (empty) ?
               E9   13   03D8   777          BEQL    10$                     ; Yes, try another system
    0C A3      61   D1   03DA   778          CMPL    (R1), PB$B_RSTATION(R3) ; No, have a pth - check for ID match
               F2   12   03DE   779          BNEQ    20$                     ; No match on first 32 bits, try again
 10 A3   04   A1   B1   03E0   780          CMPW    4(R1), PB$B_RSTATION+4(R3) ; Check high order
               EB   12   03E5   781          BNEQ    20$                     ; No match on next 16 bits, try again
 24 A3   08   A1   D1   03E7   782          CMPL    8(R1), PB$T_LPORT_NAME(R3) ; Check local port name
               E4   12   03EC   783          BNEQ    20$                     ; Branch if no match, try again
                        03EE   784 ;
                        03EE   785 ; Found a path block whose virtual circuit matches - return the info.
                        03EE   786 ;
          52   6E   D0   03EE   787          MOVL    (SP),R2                 ; Recover addr to store info
               34   13   03F1   788          BEQL    40$                     ; None, done store anything
    51   0C   A3   DE   03F3   789          MOVAL   PB$B_RSTATION(R3), R1   ; R1 covers info in PB
          82   81   7D   03F7   790          MOVQ    (R1)+, (R2)+            ; Transfer 8 bytes (RSTATION)
          82   81   7D   03FA   791          MOVQ    (R1)+, (R2)+            ; Transfer 8 bytes (port type/rev)
          82   81   7D   03FD   792          MOVQ    (R1)+, (R2)+            ; Transfer 8 bytes (port fcn/state,retries)
          82   61   7D   0400   793          MOVQ    (R1), (R2)+             ; Transfer 8 bytes (LPORT name/cable sts)
          82   01   CE   0403   794          MNEGL   #1, (R2)+               ; Set reserved address to
          82   01   CE   0406   795          MNEGL   #1, (R2)+               ; to indicate no next path.
```

```
                     82   D4   0409    796           CLRL    (R2)+                           ; Zero out next local port name
                51   63   D0   040B    797           MOVL    (R3), R1                        ; Get next path block
                54   51   D1   040E    798           CMPL    R1, R4                          ; Is there a next?
                     0D   13   0411    799           BEQL    30$                             ; No
          F4 A2    0C   A1   7D   0413    800           MOVQ    PB$B_RSTATION(R1), -12(R2) ; Yes, transfer its station address
                   FA   A2   B4   0418    801           CLRW    -6(R2)                          ; Clear out reserved field
          FC A2    24   A1   D0   041B    802           MOVL    PB$T_LPORT_NAME(R1), -4(R2) ;  and transfer its local port name
             62    18   A0   7D   0420    803 30$:      MOVQ    SB$B_SYSTEMID(R0), (R2) ; Transfer associated system ID
                   06   A2   B4   0424    804           CLRW    6(R2)                           ; Clear out reserved field
                50   01   D0   0427    805 40$:      MOVL    #1, R0                          ; Set for success
                     05   11   042A    806           BRB     60$                             ; Clean up
                                042C    807
           50    028C   8F   3C   042C    808 50$:      MOVZWL  #SS$_NOSUCHNODE, R0             ; Set for failure
                     52 8ED0   0431    809 60$:      POPL    R2                              ; Restore R2
                >1   53   D0   0434    810           MOVL    R3, R1                          ; Transfer path block pointer
                53   8E   7D   0437    811           MOVQ    (SP)+, R3                       ; Restore R3,R4
                     05   043A    812           RSB                                     ; Leave
```

```
043B    814          .SBTTL  SCS$CONFIG_SYS
043B    815 ;++
043B    816 ;
043B    817 ; FUNCTIONAL DESCRIPTION:
043B    818 ;
043B    819 ;       CONFIG_SYS searches the list of System Blocks  until  it
043B    820 ;       finds  one  with  a system ID matching the specified ID.
043B    821 ;       If no match is found,  error  is  returned.   Otherwise,
043B    822 ;       most  of  the SB is returned to the caller together with
043B    823 ;       the ID of the next system in the list.
043B    824 ;
043B    825 ;       A fork process interested in finding out about  all  the
043B    826 ;       systems  in  the  configuration simply issues successive
043B    827 ;       CONFIG_SYS  calls  on  the  next  system  ID  until  the
043B    828 ;       returned  next  system ID = 0.  To get information about
043B    829 ;       the first system, specify system ID of 0.   There  is  a
043B    830 ;       possibility  that  the configuration will change between
043B    831 ;       calls unless the entire configuration scan is conducted
043B    832 ;       at SCS IPL or higher.
043B    833 ;
043B    834 ; CALLING SEQUENCE:
043B    835 ;
043B    836 ;       BSBW    CONFIG_SYS
043B    837 ;
043B    838 ; INPUTS:
043B    839 ;
043B    840 ;       R1 = Address of system id to search for
043B    841 ;       R2 = Address of output buffer to return system information
043B    842 ;               0 if none desired
043B    843 ;
043B    844 ; OUTPUTS:
043B    845 ;
043B    846 ;       R0 = SS$_NOSUCHNODE if system id not found
043B    847 ;       R1 = Address of system block found (if R0 = success)
043B    848 ;       R2,R3,R4,R5  Preserved
043B    849 ;
043B    850 ;--
043B    851
043B    852 ASSUME  SB$B_SYSTEMID+8   EQ  SB$W_MAXDG
043B    853 ASSUME  SB$W_MAXDG+2      EQ  SB$W_MAXMSG
043B    854 ASSUME  SB$W_MAXMSG+2     EQ  SB$T_SWTYPE
043B    855 ASSUME  SB$T_SWTYPE+4     EQ  SB$T_SWVERS
043B    856 ASSUME  SB$T_SWVERS+4     EQ  SB$Q_SWINCARN
043B    857 ASSUME  SB$Q_SWINCARN+8   EQ  SB$T_HWTYPE
043B    858 ASSUME  SB$T_HWTYPE+4     EQ  SB$B_HWVERS
043B    859 ASSUME  SB$B_HWVERS+12    EQ  SB$T_NODENAME
043B    860 ASSUME  SB$T_NODENAME+16  EQ  SB$L_DDB
043B    861
043B    862 ASSUME  SBO$B_SYSTEMID+8  EQ  SBO$W_MAXDG
043B    863 ASSUME  SBO$W_MAXDG+2     EQ  SBO$W_MAXMSG
043B    864 ASSUME  SBO$W_MAXMSG+2    EQ  SBO$T_SWTYPE
043B    865 ASSUME  SBO$T_SWTYPE+4    EQ  SBO$T_SWVERS
043B    866 ASSUME  SBO$T_SWVERS+4    EQ  SBO$Q_SWINCARN
043B    867 ASSUME  SBO$Q_SWINCARN+8  EQ  SBO$T_HWTYPE
043B    868 ASSUME  SBO$T_HWTYPE+4    EQ  SBO$B_HWVERS
043B    869 ASSUME  SBO$B_HWVERS+12   EQ  SBO$T_NODENAME
043B    870 ASSUME  SBO$T_NODENAME+16 EQ  SBO$B_RSTATION1
```

```
                        043B   871 ASSUME  SBO$B_RSTATION1+8  EQ  SBO$T_LPORT1
                        043B   872 ASSUME  SBO$T_LPORT1+4    EQ  SBO$B_NXT_SYSID
                        043B   873
                0000003C 043B   874 SB_COPY_LEN = SB$T_NODENAME+16 - SB$B_SYSTEMID
                        043B   875
                        043B   876 SCS$CONFIG_SYS:
                   0C   BB 043B   877          PUSHR   #^M<R2,R3>                      ; Save reg. and output buffer ptr.
50   00000000'9F   DE 043D   878          MOVAL   @#SCS$GQ_CONFIG, R0             ; Pick up pointer to database
                52   50 D0 0444   879          MOVL    R0, R2                          ; Hold starting point
                50   60 D0 0447   880 10$:     MOVL    (R0), R0                        ; Get next block in list
                52   50 D1 044A   881          CMPL    R0, R2                          ; Back where we started (empty) ?
                     5D 13 044D   882          BEQL    60$                             ; Yes, leave now
                     61 D5 044F   883          TSTL    (R1)                            ; Is this possibly the magic 0 value?
                     05 12 0451   884          BNEQ    20$                             ; No, continue
                   04 A1 B5 0453   885          TSTW    4(R1)                           ; Yes, try the high 16 bits
                     0D 13 0456   886          BEQL    30$                             ; Yes, it is 0 - just get the first one
                18 A0   61 D1 0458   887 20$:     CMPL    (R1), SB$B_SYSTEMID(R0)         ; Nonzero value - check for a match
                     E9 12 045C   888          BNEQ    10$                             ; No match - try again
             1C A0   04 A1 B1 045E   889          CMPW    4(R1), SB$B_SYSTEMID+4(R0)      ; Lo 32 bits match - try high 16 bit
                     E2 12 0463   890          BNEQ    10$                             ; No match - try again
                        0465   891 ;
                        0465   892 ; Found a matching system block - transfer the information
                        0465   893 ;
                   5E   6E D0 0465   894 30$:     MOVL    (SP),R3                         ; Recover pointer to output array
                        3A 13 0468   895          BEQL    50$                             ; None, don't bother to transfer
                        35 BB 046A   896          PUSHR   #^M<R0,R2,R4,R5>                ; Save registers destroyed in MOVC
                        3C 28 046C   897          MOVC3   #SB_COPY_LEN,-                  ; Copy data from SYSTEMID in SB
             63   18 A0      046E   898                  SB$B_SYSTEMID(R0),(R3)          ;  through nodename to output array
                        35 BA 0471   899          POPR    #^M<R0,R2,R4,R5>                ; Restore registers
                   83   01 CE 0473   900          MNEGL   #1,(R3)+                        ; Set 1st station addr to -1
                   83   01 CE 0476   901          MNEGL   #1,(R3)+                        ;  in case there is no path on SB
                        83 D4 0479   902          CLRL    (R3)+                           ; Zero local port name of first path
             51   0C A0 D0 047B   903          MOVL    SB$L_PBFL(R0), R1               ; Get 1st path block
                        12 13 047F   904          BEQL    40$                             ; None there
                   61   51 D1 0481   905          CMPL    R1, (R1)                        ; Is there really one?
                        0D 13 0484   906          BEQL    40$                             ; No, skip on
          F4 A3   0C A1 7D 0486   907          MOVQ    PB$B_RSTATION(R1), -12(R3)      ; Yes, move in station id
                   FA A3 B4 048B   908          CLRW    -6(R3)                          ; Clear out reserved word
          FC A3   24 A1 D0 048E   909          MOVL    PB$T_LPORT_NAME(R1),-4(R3)      ; Move in local port name
                        63 7C 0493   910 40$:     CLRQ    (R3)                            ; Clear out next system id
                   51   60 D0 0495   911          MOVL    SB$L_FLINK(R0), R1              ; Get next system block
                   52   51 D1 0498   912          CMPL    R1, R2                          ; Is there really one?
                        07 13 049B   913          BEQL    50$                             ; No, skip on
                63   18 A1 7D 049D   914          MOVQ    SB$B_SYSTEMID(R1), (R3)         ; Yes, move in system id
                     06 A3 B4 04A1   915          CLRW    6(R3)                           ; Clear out reserved word
                   51   50 D0 04A4   916 50$:     MOVL    R0, R1                          ; Recover system block address
                   50   01 D0 04A7   917          MOVL    #1, R0                          ; Set success
                        05 11 04AA   918          BRB     70$
                        04AC   919
                50   028C 8F 3C 04AC   920 60$:     MOVZWL  #SS$_NOSUCHNODE, R0             ; Set failure
                   52   8E 7D 04B1   921 70$:     MOVQ    (SP)+, R2                       ; Clean stack
                        05 04B4   922          RSB
```

```
                     04B5    924          .SBTTL  SCS$CONNECT
                     04B5    925 ;++
                     04B5    926 ;
                     04B5    927 ; FUNCTIONAL DESCRIPTION:
                     04B5    928 ;
                     04B5    929 ;
                     04B5    930 ; CALLING SEQUENCE:
                     04B5    931 ;
                     04B5    932 ;       BSBW    SCS$CONNECT
                     04B5    933 ;
                     04B5    934 ; INPUTS:
                     04B5    935 ;
                     04B5    936 ;        0(SP) = Input address                          MSGADR
                     04B5    937 ;        4(SP) = [Datagram input address]               DGADR
                     04B5    938 ;        8(SP) = Error address                          ERRADR
                     04B5    939 ;       12(SP) = Address of remote system ID            RSYSID
                     04B5    940 ;       16(SP) = [Address of remote station addr]       RSTADR
                     04B5    941 ;       20(SP) = Address of remote process name         RPRNAM
                     04B5    942 ;       24(SP) = Address of local process name          LPRNAM
                     04B5    943 ;       28(SP) = Initial credit value                   INITCR
                     04B5    944 ;       30(SP) = Minimum send credit value              MINSCR
                     04B5    945 ;       32(SP) = [Initial datagram credit value]        INITDG
                     04B5    946 ;       34(SP) = [Block transfer priority]              BLKPRI
                     04B5    947 ;       36(SP) = [Address of connect data]              CONDAT
                     04B5    948 ;       40(SP) = [Address of auxilary structure]        AUXSTR
                     04B5    949 ;       44(SP) = [Bad response packet address]          BADRSP
                     04B5    950 ;       48(SP) = Return address
                     04B5    951 ;
                     04B5    952 ; OUTPUTS:
                     04B5    953 ;
                     04B5    954 ;       R0 = Status
                     04B5    955 ;       R3 = Address of CDT allocated if success
                     04B5    956 ;          = 0 if error status
                     04B5    957 ;       R4 = Address of PDT for this station address
                     04B5    958 ;       R1,R2 Destroyed
                     04B5    959 ;       R5  Preserved
                     04B5    960 ;
                     04B5    961 ;       CDT fields filled in
                     04B5    962 ;
                     04B5    963 ;--
                     04B5    964
                     04B5    965          .ENABLE lsb
                     04B5    966 SCS$CONNECT:
      FE3B     30    04B5    967          BSBW    SCS$ALLOC_CDT                  ; Allocate a CDT
    03 50      E8    04B8    968          BLBS    R0,5$                          ; Okay
     00AF      31    04BB    969          BRW     140$                           ; None, clean up and leave
                     04BE    970          ASSUME  CDT$L_MSGINPUT+4 EQ CDT$L_DGINPUT
  63   8E      7D    04BE    971 5$:      MOVQ    (SP)+,CDT$L_MSGINPUT(R3)       ; Get both MSGINPUT and DGINPUT
    0C A3   8ED0    04C1    972          POPL    CDT$L_ERRADDR(R3)              ; Set address of error routine
                     04C5    973 ;
                     04C5    974 ; Check if remote station/local port specified.  If so, go connect over
                     04C5    975 ;  that vc.
                     04C5    976 ;
    04 AE      D5    04C5    977          TSTL    4(SP)                          ; RSTADR specified?
       4B      12    04C8    978          BNEQ    70$                            ; Branch if so
                     04CA    979 ;
                     04CA    980 ; Now obtain the remote system id and use it to search the system
```

```
                        04CA   981 ;  blocks for the correct one.  If found, return the PDT for the next
                        04CA   982 ;  path block and update that field.
                        04CA   983 ;
            51 8ED0     04CA   984          POPL    R1                          ; Pick up addr of rem system id
               52   D4  04CD   985          CLRL    R2                          ; Don't want any info block
          FF69   30     04CF   986          BSBW    SCS$CONFIG_SYS              ; Get the pointer to system block
        04 50   E8      04D2   987          BLBS    R0, 10$                     ; Found it, continue
               8E   D5  04D5   988          TSTL    (SP)+                       ; Get rid of rem station addr
               57   11  04D7   989          BRB     90$                         ; Clean up stack and leave
                        04D9   990 ;
                        04D9   991 ; From the system block pick up the path block.
                        04D9   992 ;
               8E   D5  04D9   993 10$:     TSTL    (SP)+                       ; Remove RSTADR parameter
    54   14 A1   D0      04DB   994          MOVL    SB$L_PBCONNX(R1), R4        ; Get the next path block to use
               15   13  04DF   995          BEQL    35$                         ; Branch if no PB's available
    50   0C A1   DE      04E1   996          MOVAL   SB$L_PBFL(R1), R0           ; Hold path block list head
        52   54   D0      04E5   997          MOVL    R4, R2                      ; Save original pointer
    03   12 A4   B1      04E8   998 20$:     CMPW    PB$W_STATE(R4),#PB$C_OPEN ; Is it good?
               16   13  04EC   999          BEQL    50$                         ; Yes
        54   64   D0      04EE  1000 30$:     MOVL    PB$L_FLINK(R4), R4          ; No, try next block
        52   54   D1      04F1  1001          CMPL    R4, R2                      ; Back to start?
               07   12  04F4  1002          BNEQ    40$                         ; No, try another
50   2094 8F   3C      04F6  1003 35$:     MOVZWL  #SS$_UNREACHABLE, R0        ; Yes, no good path
               33   11  04FB  1004          BRB     90$                         ; Error out
                        04FD  1005
        50   54   D1      04FD  1006 40$:     CMPL    R4, R0                      ; Is it list head? (system block)
               EC   13  0500  1007          BEQL    30$                         ; Yes, get another block
               E4   11  0502  1008          BRB     20$                         ; No, try it
                        0504  1009
        52   64   D0      0504  1010 50$:     MOVL    PB$L_FLINK(R4), R2          ; Try to find the next next block
        50   52   D1      0507  1011          CMPL    R2, R0                      ; Is it list head?
               03   12  050A  1012          BNEQ    60$                         ; No, use it
        52   62   D0      050C  1013          MOVL    PB$L_FLINK(R2), R2          ; Yes, get the first in list
    14 A1   52   D0      050F  1014 60$:     MOVL    R2, SB$L_PBCONNX(R1)        ; Update next path block entry
               23   11  0513  1015          BRB     100$
                        0515  1016 ;
                        0515  1017 ; Remote station/local port specified, get the exact path.  Check to see
                        0515  1018 ; that it is open.
                        0515  1019 ;
               8E   D5  0515  1020 70$:     TSTL    (SP)+                       ; Clear remote system ID parameter
            51 8ED0     0517  1021          POPL    R1                          ; Get addr of remote sta/local port
               52   D4  051A  1022          CLRL    R2                          ; No info to move
          FE98   30     051C  1023          BSBW    SCS$CONFIG_PTH              ; Find it
        54   51   D0      051F  1024          MOVL    R1, R4                      ; Transfer to correct register
          0B 50   E9      0522  1025          BLBC    R0, 90$                     ; Error, leave
    03   12 A4   B1      0525  1026 80$:     CMPW    PB$W_STATE(R4),#PB$C_OPEN ; Is it good?
               0D   13  0529  1027          BEQL    100$                        ; Yes
50   2094 8F   3C      052B  1028          MOVZWL  #SS$_UNREACHABLE, R0        ; No, no good path
             0076   30     0530  1029 90$:     BSBW    SCS$DEALL_CDT1              ; Bad id, get rid of the CDT
        5E   08   C0      0533  1030          ADDL    #<2*4>, SP                  ; Get rid of process names
               38   11  0536  1031          BRB     150$                        ; Leave
                        0538  1032
20 A3   0C A4   D0      0538  1033 100$:    MOVL    PB$B_RSTATION(R4), -        ; Move in 32 bits of Remote Sta Id
                        053D  1034                  CDT$B_RSTATION(R3)          ;    into the CDT
24 A3   10 A4   B0      053D  1035          MOVW    PB$B_RSTATION+4(R4), -      ; Another 16 bits
                        0542  1036                  CDT$B_RSTATION+4(R3)
    1C A3   54   D0      0542  1037          MOVL    R4, CDT$L_PB(R3)            ; Store the pointer to Path Block
```

```
                54   2C A4    D0  0546  1038          MOVL    PB$L_PDT(R4), R4        ; Recover PDT entry
                10 A3   54    D0  054A  1039          MOVL    R4, CDT$L_PDT(R3)       ; Store the PDT
                   51   18    9A  054E  1040          MOVZBL  #PDT$L_CONNECT, R1      ; Offset in PDT to jump to
                                    0551  1041          ASSUME  CDT$L_RPROCNAM+4 EQ CDT$L_LPROCNAM
                50 A3   8E    7D  0551  1042          MOVQ    (SP)+,CDT$L_RPROCNAM(R3); Get both local & remote process names
                                    0555  1043 ;
                                    0555  1044 ; Now take care of the credit fields
                                    0555  1045 ;
                                    0555  1046          ASSUME  CDT$W_MINSEND EQ CDT$W_INITLREC+2
                                    0555  1047          ASSUME  CDT$W_DGREC EQ CDT$W_MINSEND+2
                                    0555  1048          ASSUME  CDT$B_PRIORITY EQ CDT$W_DGREC+2
                48 A3   8E    7D  0555  1049 PDT_JMP:  MOVQ    (SP)+,CDT$W_INITLREC(R3); Pick up initial rec, min send, dgrec,pro
                      58 A3 8ED0  0559  1050          POPL    CDT$L_CONDAT(R3)        ; Transfer connection data
                         05   12  055D  1051          BNEQ    110$                    ; Something there
             58 A3    74'AF   9E  055F  1052          MOVAB   B^BLANK,CDT$L_CONDAT(R3); Else give 'em blanks
                                    0564  1053          ASSUME  CDT$L_AUXSTRUC+4 EQ CDT$L_BADRSP
                5C A3   8E    7D  0564  1054 110$:    MOVQ    (SP)+,CDT$L_AUXSTRUC(R3); Pick up aux struc addr & bad resp addr
                   51   54    C0  0568  1055          ADDL2   R4,R1                   ; Pick up address from PDT
                        91    17  056B  1056          JMP     @(R1)+                  ; Finish in port dependent code
                                    056D  1057
                   5E   1C    C0  056D  1058 140$:    ADDL    #7*4, SP                ; Clean args off stack
                   5E   14    C0  0570  1059 150$:    ADDL    #5*4, SP
                                05  0573  1060          RSB
                                    0574  1061
20 20 20 20 20 20 20 20 20 20 20 20 20  0574  1062 BLANK:  .ASCII/                    /
                      20 20 20 20 20    0580
                                    0584  1063          .DISABLE lsb
```

```
                     0584  1065          .SBTTL  SCS$DEALL_CDT
                     0584  1066 ;++
                     0584  1067 ;
                     0584  1068 ; FUNCTIONAL DESCRIPTION:
                     0584  1069 ;
                     0584  1070 ;
                     0584  1071 ; CALLING SEQUENCE:
                     0584  1072 ;
                     0584  1073 ;       BSBW    SCS$DEALL_CDT
                     0584  1074 ;
                     0584  1075 ; INPUTS:
                     0584  1076 ;
                     0584  1077 ;       R3 = Pointer to CDT
                     0584  1078 ;
                     0584  1079 ; OUTPUTS:
                     0584  1080 ;
                     0584  1081 ;       R1 trashed
                     0584  1082 ;       R3 cleared
                     0584  1083 ;       R0,R2,R4,R5  preserved
                     0584  1084 ;
                     0584  1085 ; COMPLETION CODES:
                     0584  1086 ;
                     0584  1087 ;       NONE (always succedes)
                     0584  1088 ;
                     0584  1089 ;--
                     0584  1090
                     0584  1091 SCS$DEALL_CDT:                          ; Unlink from PB first
          52   DD    0584  1092          PUSHL   R2                      ; Save a register
51   1C A3   D0    0586  1093          MOVL    CDT$L_PB(R3), R1        ; Pick up Path block
51   C8 A1   DE    058A  1094          MOVAL   PB$L_CDTLST-CDT$L_CDTLST(R1), R1 ; Point to head of list
52   6C A1   D0    058E  1095 10$:     MOVL    CDT$L_CDTLST(R1), R2    ; Get next block
          12   13    0592  1096          BEQL    30$                     ; There is none
     53   52   D1    0594  1097          CMPL    R2, R3                  ; Are we next block?
          05   13    0597  1098          BEQL    20$                     ; Yes
     51   52   D0    0599  1099          MOVL    R2, R1                  ; No, shift pointer
          F0   11    059C  1100          BRB     10$                     ; And try again
                     059E  1101
6C A1     6C A3    D0    059E  1102 20$:     MOVL    CDT$L_CDTLST(R3), -
                     05A3  1103                      CDT$L_CDTLST(R1)    ; Put our next link in last block
          6C A3    D4    05A3  1104          CLRL    CDT$L_CDTLST(R3)    ; Clear out link
             52 8ED0    05A6  1105 30$:     POPL    R2                      ; Restore a register
                     05A9  1106
                     05A9  1107 SCS$DEALL_CDT1:                         ; Deallocate w/o unlinking
51   00000000'9F    D0    05A9  1108          MOVL    @#SCS$GL_CDL, R1        ; Pick up list head
     63   F4 A1    D0    05B0  1109          MOVL    CDL$L_FREECDT(R1), -
                     05B4  1110                      CDT$L_LINK(R3)      ; Insert in chain
     F4 A1   53    D0    05B4  1111          MOVL    R3, CDL$L_FREECDT(R1)   ; Set this block as first free
          1A A3    B6    05B8  1112 10$:     INCW    CDT$L_LCONID+2(R3)      ; Bump sequence number
             FB    13    05BB  1113          BEQL    10$                     ; Zero not allowed
                     05BD  1114          ASSUME  CDT$W_STATE+2 EQ CDT$W_BLKSTATE
          28 A3    D4    05BD  1115          CLRL    CDT$W_STATE(R3)         ; Set to CLOSED
             53    D4    05C0  1116          CLRL    R3                      ; Eliminate any chance of spurious CDT
                   05    05C2  1117          RSB
```

```
                     05C3  1119          .SBTTL  SCS$DEALL_RSPID - Dellocate a response id
                     05C3  1120          .SBTTL  SCS$RECYL_RSPID - Recycle a response id
                     05C3  1121 ;++
                     05C3  1122 ;
                     05C3  1123 ; SCS$DEALL_RSPID - Dellocate a response id
                     05C3  1124 ; SCS$RECYL_RSPID - Recycle a response id
                     05C3  1125 ;
                     05C3  1126 ; Functional description:
                     05C3  1127 ;
                     05C3  1128 ;       Check validity of the value in the CDRP$L_RSPID field. Increment
                     05C3  1129 ;       the RDTE sequence number.
                     05C3  1130 ;
                     05C3  1131 ;       For deallocate a response id (SCS$DEALL_RSPID):
                     05C3  1132 ;
                     05C3  1133 ;               Dequeue next waiter for an RDTE and start him up.  If no
                     05C3  1134 ;               waiter to dequeue, set the RDTE state to unallocated, link
                     05C3  1135 ;               RDTE as first entry in the free list, and return
                     05C3  1136 ;
                     05C3  1137 ;       For recycle a response id (SCS$RECYL_RSPID):
                     05C3  1138 ;
                     05C3  1139 ;               Update response id in CDRP and return.
                     05C3  1140 ;
                     05C3  1141 ; Calling sequence:
                     05C3  1142 ;
                     05C3  1143 ;       BSBW    SCS$DEALL_RSPID
                     05C3  1144 ;       BSBW    SCS$RECYL_RSPID
                     05C3  1145 ;
                     05C3  1146 ; Inputs:
                     05C3  1147 ;
                     05C3  1148 ;       R5 --> CDRP
                     05C3  1149 ;       SCS$GL_RDT
                     05C3  1150 ;
                     05C3  1151 ; Outputs:
                     05C3  1152 ;
                     05C3  1153 ;       R0,R1,R2  Destroyed; R3,R4,R5  Preserved -- SCS$DEALL_RSPID
                     05C3  1154 ;       R0,R1  Destroyed; R2,R3,R4,R5  Preserved -- SCS$RECYL_RSPID
                     05C3  1155 ;
                     05C3  1156 ;--
                     05C3  1157
                     05C3  1158 SCS$RECYL_RSPID:
          7E    D4   05C3  1159          CLRL    -(SP)                   ; Put SCS$RECYL_RSPID flag on stack.
                     05C5  1160
                     05C5  1161 SCS$DEALL_RSPID:
                     05C5  1162
          55    DD   05C5  1163          PUSHL   R5                      ; Save CDRP address.
55     20 A5    D0   05C7  1164          MOVL    CDRP$L_RSPID(R5), R5    ; Get RSPID (if any).
          4A    13   05CB  1165          BEQL    90$                     ; No RSPID, then bugcheck.
        018F    30   05CD  1166          BSBW    SCS$FIND_RDTE           ; Locate RDTE for the RSPID.
       44 50    E9   05D0  1167          BLBC    R0, 90$                 ; Branch if error in RSPID.
    50    55    D0   05D3  1168          MOVL    R5, R0                  ; Save RDTE address.
       06 A0    B6   05D6  1169 5$:      INCW    RD$W_SEQNUM(R0)         ; Bump RDTE sequence number.
          FB    13   05D9  1170          BEQL    5$                      ; Zero sequence number not allowed.
       04 AE    D5   05DB  1171          TSTL    4(SP)                   ; Was this a SCS$RECYL_RSPID call?
          3B    13   05DE  1172          BEQL    50$                     ; Branch if recycling a response id.
55     E8 B1    0F   05E0  1173          REMQUE  @RDT$L_WAITFL(R1), R5   ; Grab any process waiting on RSPIDs.
          1F    1D   05E4  1174          BVS     10$                     ; Branch if no processes waiting.
                     05E6  1175 ;
```

```
                          05E6  1176 ; There is a stalled process waiting for a RSPID, resume it.
                          05E6  1177 ;
          7E    53    7D  05E6  1178          MOVQ    R3, -(SP)               ; Save context registers.
    52    50    51    C3  05E9  1179          SUBL3   R1, R0, R2              ; Get RDTE offset from RDT base.
                          05ED  1180          ASSUME  RD$C_LENGTH EQ 8
    52    52    0D    78  05ED  1181          ASHL    #13, R2, R2             ; Form index in high order R2.
       52    06 A0    B0  05F1  1182          MOVW    RD$W_SEQNUM(R0), R2     ; Insert sequence number.
 20 A5    52    10    9C  05F5  1183          ROTL    #16, R2, CDRP$L_RSPID(R5) ; Form final RSPID and store in CDRP.
          60    55    D0  05FA  1184          MOVL    R5, RD$L_CDRP(R0)       ; Store his CDRP in the RDTE.
             024B    30  05FD  1185          BSBW    SCS$RESUMEWAITR         ; Resume the waiter
          53    8E    7D  0600  1186          MOVQ    (SP)+, R3               ; Restore saved registers.
                08    11  0603  1187          BRB     20$                     ; Join common completion path.
                          0605  1188
             04 A0    B4  0605  1189 10$:     CLRW    RD$W_STATE(R0)          ; Set state as free.
       60    F4 A1    D0  0608  1190          MOVL    RDT$L_FREERD(R1), -
                          060C  1191                  RD$L_LINK(R0)           ; Insert RDTE in free RDTE chain.
    F4 A1       50    D0  060C  1192          MOVL    R0, RDT$L_FREERD(R1)    ; Make it the first free entry.
                55 8ED0  0610  1193 20$:     POPL    R5                      ; Restore CDRP address.
             20 A5    D4  0613  1194          CLRL    CDRP$L_RSPID(R5)        ; Clear out RSPID.
                05    05  0616  1195          RSB                             ; Return from SCS$DEALL_RSPID.
                          0617  1196
                          0617  1197
                          0617  1198 90$:     BUG_CHECK  INVRSPID,FATAL       ; "Invalid RSPID" bugcheck.
                          061B  1199
                          061B  1200
                          061B  1201                                          ; Complete SCS$RECYL_RSPID call.
                55 8ED0  061B  1202 50$:     POPL    R5                      ; Restore CDRP address.
 22 A5    06 A0    B0  061E  1203          MOVW    RD$W_SEQNUM(R0), -      ; Copy new sequence number to
                          0623  1204                  CDRP$L_RSPID+2(R5)      ; CDRP.
                8E    D5  0623  1205          TSTL    (SP)+                   ; Pop SCS$RECYL_RSPID call flag.
                05    05  0625  1206          RSB                             ; Return from SCS$RECYL_RSPID.
```

```
                     0626  1208          .SBTTL  SCS$DIRECTORY
                     0626  1209 ;+
                     0626  1210 ;
                     0626  1211 ; FUNCTIONAL DESCRIPTION:
                     0626  1212 ;
                     0626  1213 ;
                     0626  1214 ; CALLING SEQUENCE:
                     0626  1215 ;
                     0626  1216 ;       BSBW    SCS$DIRECTORY
                     0626  1217 ;
                     0626  1218 ; INPUTS:
                     0626  1219 ;
                     0626  1220 ;       NONE
                     0626  1221 ;
                     0626  1222 ; OUTPUTS:
                     0626  1223 ;
                     0626  1224 ;       NONE
                     0626  1225 ;
                     0626  1226 ; SIDE EFFECTS:
                     0626  1227 ;
                     0626  1228 ;       NONE
                     0626  1229 ;
                     0626  1230 ;--
                     0626  1231
                     0626  1232 SCS$DIRECTORY:
                     0626  1233          ACCEPT  MSGADR=B^30$,-
                     0626  1234                  ERRADR=W^DIRERR,-       ; If error, disconnect.
                     0626  1235                  INITCR=#1               ; We'll accept anything
          01 50  E9  064A  1236          BLBC    R0,10$                  ; If error, reject
                 05  064D  1237          RSB
                     064E  1238
     53  52    D0  064E  1239 10$:     MOVL    R2,R3                   ; Copy of listening CDT address
                     0651  1240          REJECT
                 05  0654  1241          RSB
                     0655  1242
51    04 A2    9E  0655  1243 30$:     MOVAB   4(R2), R1               ; Pick up procname
   7E    52    7D  0659  1244          MOVQ    R2,-(SP)                ; Save address of message and CDT
      0154    30  065C  1245          BSBW    SCS$LOCLOOKUP           ; Look it up
   62    50    B0  065F  1246          MOVW    R0, (R2)                ; Set status
     0C  50    E9  0662  1247          BLBC    R0,35$                  ; If LBC, unsuccessful lookup
52    14 A2    9E  0665  1248          MOVAB   <16+4>(R2), R2          ; Set to info
82    1C A1    7D  0669  1249          MOVQ    SDIR$B_PROCINF(R1),(R2)+    ; Transfer first 8 bytes
62    24 A1    7D  066D  1250          MOVQ    <SDIR$B_PROCINF+8>(R1),(R2) ; ... and sec 8
   51    2C    3C  0671  1251 35$:     MOVZWL  #CDRP$C_LENGTH, R1      ; Size of block needed
      01BF    30  0674  1252          BSBW    SCS_ALONONPAGED         ; Allocate it
     2E  50    E9  0677  1253          BLBC    R0,40$                  ; Branch if no pool available
   55    52    D0  067A  1254          MOVL    R2, R5                  ; Transfer address
   52    8E    7D  067D  1255          MOVQ    (SP)+, R2               ; Restore registers
08 A5    51    B0  0680  1256          MOVW    R1, CDRP$W_CDRPSIZE(R5) ; Set up block
0A A5    39    9B  0684  1257          MOVZBW  #DYN$C_CDRP, -
                     0688  1258                  CDRP$B_CD_TYPE(R5)      ; Set type
     18 A5    D4  0688  1259          CLRL    CDRP$L_SAVD_RTN(R5)     ; Just to be safe
1C A5    52    D0  068B  1260          MOVL    R2, CDRP$L_MSG_BUF(R5)  ; Set address of buffer
     20 A5    D4  068F  1261          CLRL    CDRP$L_RSPID(R5)        ; Indicate we are turning around
24 A5    53    D0  0692  1262          MOVL    R3, CDRP$L_CDT(R5)      ; Set in pointer
     28 A5    D4  0696  1263          CLRL    CDRP$L_RWCPTR(R5)       ; Just to be safe
                     0699  1264          RECYCL_MSG_BUF                  ; We are going to recycle the buffer
```

```
                51   24   9A  069C  1265          MOVZBL  #36,R1                  ; Maximum size of message
                                 069F  1266          SEND_CNT_MSG_BUF                ; Send message and term thread
                50   55   D0  06A2  1267          MOVL    R5, R0                  ; Set pointer to head of block
                   019D   31  06A5  1268          BRW     SCS_DEALNONPAGD         ; Get rid of the block
                                 06A8  1269
                                 06A8  1270 ;
                                 06A8  1271 ;  We come here if we could not allocate pool for the CDRP
                                 06A8  1272 ;
                52   8E   7D  06A8  1273 40$:     MOVQ    (SP)+,R2                ; Pop stack
                                 06AB  1274          DEALLOC_MSG_BUF_REG             ; Deallocate message buf
                                 06AE  1275          DISCONNECT
                           05    06B4  1276          RSB
                                 06B5  1277
                                 06B5  1278 ;
                                 06B5  1279 ;  Clean up if there is an error while responding to a directory message
                                 06B5  1280 ;
                                 06B5  1281 ;  On entry:
                                 06B5  1282 ;       R0      - error status
                                 06B5  1283 ;       R3      - address of CDT
                                 06B5  1284 ;       R4      - address of PDT
                                 06B5  1285 ;
                                 06B5  1286 DIRERR:
                                 06B5  1287          SCAN_MSGBUF_WAIT -              ; Find CDRP'S
                                 06B5  1288                  ACTION = DIR_CLEANUP
                                 06C2  1289          DISCONNECT                      ; Break connection
                           05    06C8  1290          RSB
                                 06C9  1291
                                 06C9  1292 ;
                                 06C9  1293 ;  For each CDRP which is found, remove it from queue, if needed, deallocate
                                 06C9  1294 ;  associated message buffer, if any, and return CDRP to pool.
                                 06C9  1295 ;
                                 06C9  1296 ;  On entry:
                                 06C9  1297 ;       R3      - address of CDT
                                 06C9  1298 ;       R4      - address of PDT
                                 06C9  1299 ;       R5      - address of CDRP
                                 06C9  1300 ;
                                 06C9  1301 DIR_CLEANUP:
                                 06C9  1302
                                 06C9  1303 ;
                                 06C9  1304 ;  Remove CDRP from queue, if needed
                                 06C9  1305 ;
                      52   DD   06C9  1306          PUSHL   R2
                   04 A5   D5   06CB  1307          TSTL    CDRP$L_FQBL(R5)         ; Is there a queue pointer?
                      11   13   06CE  1308          BEQL    5$                      ; If EQL, none
                55    04 B5   D1   06D0  1309          CMPL    @CDRP$L_FQBL(R5),R5     ; Is CDRP in a queue?
                      0B   12   06D4  1310          BNEQ    5$                      ; If NEQ, no
                   55   65   0F   06D6  1311          REMQUE  (R5),R5                 ; Dequeue CDRP
                52    28 A5   D0   06D9  1312          MOVL    CDRP$L_RWCPTR(R5),R2    ; Get wait count pointer
                      02   13   06DD  1313          BEQL    5$                      ; Branch if none
                      62   B7   06DF  1314          DECW    (R2)                    ; Adjust
                                 06E1  1315
                                 06E1  1316 ;
                                 06E1  1317 ;  Deallocate associated message buffer, if any
                                 06E1  1318 ;
                      52 8ED0   06E1  1319 5$:      POPL    R2                      ; Restore reg
                   1C A5   D5   06E4  1320          TSTL    CDRP$L_MSG_BUF(R5)      ; Do we have any message buffers?
                      03   13   06E7  1321          BEQL    10$                     ; If EQL, no
```

```
                     06E9  1322          DEALLOC_MSG_BUF                 ; Get rid of it
                     06EC  1323
                     06EC  1324 ;
                     06EC  1325 ;  Return CDRP to non-paged pool
                     06EC  1326 ;
          50    55   D0  06EC  1327 10$:     MOVL    R5,R0                   ; Address of CDRP in R0
00000000'GF    16   06EF  1328          JSB     G^COM$DRVDEALMEM        ; Give back to pool
                05   06F5  1329          RSB                             ; Cleanup done
                     06F6  1330
                     06F6  1331 ;
                     06F6  1332 ;  Clean up for errors on the listen
                     06F6  1333 ;
                     06F6  1334 LISTENERR:
                     06F6  1335          DISCONNECT                      ; Nothing to clean up -- disconnect
                05   06FC  1336          RSB                             ; Return from error routine.
                     06FD  1337
```

```
                     06FD  1339          .SBTTL  SCS$DISCONNECT
                     06FD  1340 ;++
                     06FD  1341 ;
                     06FD  1342 ; FUNCTIONAL DESCRIPTION:
                     06FD  1343 ;
                     06FD  1344 ;
                     06FD  1345 ; INPUTS:
                     06FD  1346 ;
                     06FD  1347 ;         R3 -->  CDT
                     06FD  1348 ;         R4 -->  PDT
                     06FD  1349 ;
                     06FD  1350 ; IMPLICIT INPUTS:
                     06FD  1351 ;
                     06FD  1352 ;         CDT$W_STATE
                     06FD  1353 ;
                     06FD  1354 ; OUTPUTS:
                     06FD  1355 ;
                     06FD  1356 ;         NONE
                     06FD  1357 ;
                     06FD  1358 ; SIDE EFFECTS:
                     06FD  1359 ;
                     06FD  1360 ;         If state was LISTEN, then directory entry removed and CDT deallocated.
                     06FD  1361 ;
                     06FD  1362 ;--
                     06FD  1363
                     06FD  1364 SCS$DISCONNECT:
                     06FD  1365          ASSUME  CDT$C_CLOSED EQ 0
          28 A3   B5 06FD  1366          TSTW    CDT$W_STATE(R3)          ; Is state CLOSED?
             06   12 0700  1367          BNEQ    10$                      ; No
50   06A9 8F   3C 0702  1368          MOVZWL  #SS$_ALRDYCLOSED,R0      ; Yes, issue qualified success
                05 0707  1369          RSB
                   0708  1370
01   28 A3   B1 0708  1371 10$:     CMPW    CDT$W_STATE(R3), -       ; Is state LISTEN?
                0C 070C  1372                  #CDT$C_LISTEN
             03   12 070C  1373          BNEQ    20$                      ; No
           00F5   31 070E  1374          BRW     SCS$REMOVE               ; Yes, remove the entry and exit
                   0711  1375
54   10 A3   D0 0711  1376 20$:     MOVL    CDT$L_PDT(R3),R4         ; Pick up the PDT address
          28 B4   17 0715  1377          JMP     @PDT$C_DCONNECT(R4)      ; Let port driver handle other states
```

```
                     0718  1379          .SBTTL  SCS$ENTER
                     0718  1380 ;++
                     0718  1381 ;
                     0718  1382 ; FUNCTIONAL DESCRIPTION:
                     0718  1383 ;
                     0718  1384 ;
                     0718  1385 ; CALLING SEQUENCE:
                     0718  1386 ;
                     0718  1387 ;       BSBW    SCS$ENTER
                     0718  1388 ;
                     0718  1389 ; INPUTS:
                     0718  1390 ;
                     0718  1391 ;       R0 = Address of process name (16 byte string)
                     0718  1392 ;       R1 = Address of process info (16 byte string)
                     0718  1393 ;       R2 = CONID
                     0718  1394 ;
                     0718  1395 ;       SCS$GQ_DIRECT
                     0718  1396 ;
                     0718  1397 ; OUTPUTS:
                     0718  1398 ;
                     0718  1399 ;       R0 = SS$_INSFMEM if no pool for directory block
                     0718  1400 ;          = SS$_NORMAL if success
                     0718  1401 ;       R1,R2  destroyed
                     0718  1402 ;       R3,R4,R5  preserved
                     0718  1403 ;
                     0718  1404 ; SIDE EFFECTS:
                     0718  1405 ;
                     0718  1406 ;       NONE
                     0718  1407 ;
                     0718  1408 ;--
                     0718  1409
                     0718  1410 SCS$ENTER:
        0F   BB      0718  1411          PUSHR   #^M<R0,R1,R2,R3>        ; Save registers and arguments.
    51  30   D0      071A  1412          MOVL    #SDIR$C_LENGTH, R1      ; Size of a directory entry
      0116   30      071D  1413          BSBW    SCS_ALONONPAGED         ; Allocate it
     37 50   E9      0720  1414          BLBC    R0, 40$                 ; None, error out
  53   08 A2   9E    0723  1415          MOVAB   SDIR$W_SIZE(R2), R3     ; Set pointer to size field
    83   51   B0      0727  1416          MOVW    R1, (R3)+               ; Set size field
83  0360 8F   B0      072A  1417          MOVW    #DYN$C_SCS!<DYN$C_SCS_DIRa8>, (R3)+ ; Size type and subtype
        51 8ED0      072F  1418          POPL    R1                      ; Get pointer to process name
    83   81   7D      0732  1419          MOVQ    (R1)+, (R3)+            ; Move 8 bytes
    83   61   7D      0735  1420          MOVQ    (R1), (R3)+             ; Move 8 bytes
        51 8ED0      0738  1421          POPL    R1                      ; Get pointer to process info
        08   13      073B  1422          BEQL    10$                     ; None present
    83   81   7D      073D  1423          MOVQ    (R1)+, (R3)+            ; Move 8 bytes
    83   61   7D      0740  1424          MOVQ    (R1), (R3)+             ; Move 8 bytes
        04   11      0743  1425          BRB     20$
                     0745  1426
        83   7C      0745  1427 10$:     CLRQ    (R3)+                   ; Clear it out
        83   7C      0747  1428          CLRQ    (R3)+                   ; Ditto
        63 8ED0      0749  1429 20$:     POPL    (R3)                    ; Set in CONID
53  00000004'9F   D0      074C  1430          MOVL    @#SCS$GQ_DIRECT+4, R3   ; End of queue
    63   62   0E      0753  1431          INSQUE  (R2), (R3)              ; Insert at end
        53 8ED0      0756  1432 30$:     POPL    R3                      ; Restore register
           05      0759  1433          RSB
                     075A  1434
    5E   0C   C0      075A  1435 40$:     ADDL    #3+4, SP                ; Clean up stack
```

```
                F7   11  075D  1436          BRB     30$                     ; ... and out
```

```
                                075F  1438          .SBTTL  SCS$FIND_RDTE - Find RDTE for RSPID
                                075F  1439 ;++
                                075F  1440 ;
                                075F  1441 ; SCS$FIND_RDTE - Find the RDTE for a given RSPID
                                075F  1442 ;
                                075F  1443 ; Functional description:
                                075F  1444 ;
                                075F  1445 ;       The RDT ID index is extracted from the given response id, RSPID, and
                                075F  1446 ;       validated.  The index is then used to locate the RDTE in the RDT.  A
                                075F  1447 ;       sequence number check is then performed and a check is made to insure
                                075F  1448 ;       that the RDTE is busy.
                                075F  1449 ;
                                075F  1450 ; Inputs:
                                075F  1451 ;
                                075F  1452 ;       R5      a response id, RSPID
                                075F  1453 ;
                                075F  1454 ;       IPL     IPL$_SCS
                                075F  1455 ;
                                075F  1456 ; Outputs:
                                075F  1457 ;
                                075F  1458 ;       R0      1 ==> Lookup successful
                                075F  1459 ;               0 ==> invalid response ID
                                075F  1460 ;       R1      RDT base address (This is not advertized to SYSAPs but is
                                075F  1461 ;                                 required by SCS$DEALL_RSPID.)
                                075F  1462 ;       R2-R4   preserved
                                075F  1463 ;       R5      RDTE address (if R0 = 1)
                                075F  1464 ;-
                                075F  1465
                                075F  1466 SCS$FIND_RDTE:
                                075F  1467
51   00000000'9F    D0          075F  1468          MOVL    @#SCS$GL_RDT, R1        ; Get base address of RDT.
     50    55      3C          0766  1469          MOVZWL  R5, R0                  ; Get RDT index from RSPID.
   F8 A1   50      D1          0769  1470          CMPL    R0, RDT$L_MAXRDIDX(R1)  ; Is index too big?
           18      1A          076D  1471          BGTRU   90$                     ; Branch if too big.
                               076F  1472          ASSUME  RD$C_LENGTH EQ 8
         6140      7F          076F  1473          PUSHAQ  (R1)[R0]                ; Get pointer to RDTE.
50   55   10      9C          0772  1474          ROTL    #16, R5, R0             ; Get response id sequence number.
             55 8ED0          0776  1475          POPL    R5                      ; Prepair to return RDTE pointer.
  50   06 A5      B1          0779  1476          CMPW    RD$W_SEQNUM(R5), R0     ; Do the sequence numbers match?
           08      12          077D  1477          BNEQ    90$                     ; Branch if no match.
                               077F  1478          ASSUME  RD$V_BUSY EQ 0
   04 04 A5      E9          077F  1479          BLBC    RD$W_STATE(R5), 90$     ; Branch if RDTE is not busy.
    50   01      D0          0783  1480          MOVL    #1, R0                  ; Set success status.
           05      05          0786  1481          RSB                             ; Return to caller.
                               0787  1482
           50      D4          0787  1483 90$:     CLRL    R0                      ; Set error status.
           05      05          0789  1484          RSB                             ; Return to caller.
```

```
                     078A  1486          .SBTTL  SCS$LISTEN
                     078A  1487 ;++
                     078A  1488 ;
                     078A  1489 ; FUNCTIONAL DESCRIPTION:
                     078A  1490 ;
                     078A  1491 ;
                     078A  1492 ; CALLING SEQUENCE:
                     078A  1493 ;
                     078A  1494 ;       BSBW    SCS$LISTEN
                     078A  1495 ;
                     078A  1496 ; INPUTS:
                     078A  1497 ;
                     078A  1498 ;       00(SP) = Address to call with CONNECT_REQ msg          MSGADR
                     078A  1499 ;       04(SP) = Address to call with error in VC              ERRADR
                     078A  1500 ;       08(SP) = Address of listening process name             LPRNAM
                     078A  1501 ;       0C(SP) = Address of listening process information      PRINFO
                     078A  1502 ;       10(SP) = Return address
                     078A  1503 ;
                     078A  1504 ; OUTPUTS:
                     078A  1505 ;
                     078A  1506 ;       R0 = SS$_INSFCDT if could not allocate CDT
                     078A  1507 ;          = SS$_INSFMEM if could not allocate pool for directory entry
                     078A  1508 ;          = SS$_NORMAL if success
                     078A  1509 ;       R1,R2 destroyed
                     078A  1510 ;       R3 = Pointer to CDT if success
                     078A  1511 ;          = 0 if failure
                     078A  1512 ;       R4,R5 preserved
                     078A  1513 ;
                     078A  1514 ;       CDT$L_INPUT     Address to call process with CONNECT_REQ msgs
                     078A  1515 ;       CDT$L_LCONID    Local connection id for CDT in R3
                     078A  1516 ;       CDT$W_STATE     LISTENING
                     078A  1517 ;
                     078A  1518 ; SIDE EFFECTS:
                     078A  1519 ;
                     078A  1520 ;       Directory entry made if successful, CDT state set to LISTEN
                     078A  1521 ;
                     078A  1522 ;--
                     078A  1523
                     078A  1524 SCS$LISTEN:
     FB66    30      078A  1525          BSBW    SCS$ALLOC_CDT           ; Grab a CDT
   18 50     E9      078D  1526          BLBC    R0, 10$                 ; None, error out
52   51     D0      0790  1527          MOVL    R1, R2                  ; Shift CONID around
     63 8ED0         0793  1528          POPL    CDT$L_MSGINPUT(R3)      ; Fill in with address
  0C A3 8ED0         0796  1529          POPL    CDT$L_ERRADDR(R3)       ; Ditto for errors
50   8E     7D      079A  1530          MOVQ    (SP)+, R0               ; Pick up arguments
   FF78     30      079D  1531          BSBW    SCS$ENTER               ; Make a directory entry
  09 50     E9      07A0  1532          BLBC    R0, 20$                 ; None, error out
28 A3   01   B0      07A3  1533          MOVW    #CDT$C_LISTEN, -        ; Set state to LISTENing
                     07A7  1534                  CDT$W_STATE(R3)
             05      07A7  1535          RSB
                     07A8  1536
5E   10   C0         07A8  1537 10$:     ADDL    #4*4, SP                ; Clean up arguments
             05      07AB  1538          RSB
                     07AC  1539
   FDFA     30      07AC  1540 20$:     BSBW    SCS$DEALL_CDT1          ; Get rid of the CDT
5E   0C   C0         07AF  1541          ADDL    #3*4, SP                ; Clean up arguments
             05      07B2  1542          RSB
```

```
                                07B3  1544          .SBTTL  SCS$LOCLOOKUP
                                07B3  1545 ;++
                                07B3  1546 ;
                                07B3  1547 ; FUNCTIONAL DESCRIPTION:
                                07B3  1548 ;
                                07B3  1549 ;
                                07B3  1550 ; INPUTS:
                                07B3  1551 ;
                                07B3  1552 ;       R1 = Addr of local process name to look up
                                07B3  1553 ;
                                07B3  1554 ; OUTPUTS:
                                07B3  1555 ;
                                07B3  1556 ;       R0 = SS$_NOSUCHOBJ if listener could not be found
                                07B3  1557 ;          = SS$_NORMAL if listener found
                                07B3  1558 ;       R1 = Addr of entry if found
                                07B3  1559 ;       R2 = Preserved
                                07B3  1560 ;       R3 = Addr of listening CDT for local process, if success
                                07B3  1561 ;          = 0 if failure
                                07B3  1562 ;       R4,R5 Preserved
                                07B3  1563 ;
                                07B3  1564 ;--
                                07B3  1565
                                07B3  1566 SCS$LOCLOOKUP:
                          55    DD  07B3  1567          PUSHL   R5                      ; Save a register
                       55 51    D0  07B5  1568          MOVL    R1, R5                  ; Copy addr of proc name
                          54    DD  07B8  1569          PUSHL   R4                      ; Save another reg
                          52    DD  07BA  1570          PUSHL   R2                      ; ... and another
                54 00000000'9F    DE  07BC  1571          MOVAL   @#SCS$GQ_DIRECT, R4     ; Get head of directory list
                          54    DD  07C3  1572          PUSHL   R4                      ; Save a copy
                       54 64    D0  07C5  1573 10$:     MOVL    (R4), R4                ; Next entry
                       6E 54    D1  07C8  1574          CMPL    R4, (SP)                ; End of the line?
                          30    13  07CB  1575          BEQL    30$                     ; Yes, leave - not found
             0C A4     65 10    29  07CD  1576          CMPC3   #16, (R5), -
                                07D2  1577                  SDIR$B_PROCNAM(R4)      ; Names match?
                             F1 12  07D2  1578          BNEQ    10$                     ; No, try again
                       55 2C A4 D0  07D4  1579          MOVL    SDIR$L_CONID(R4), R5    ; Yes, pick up CONID
                             23 13  07D8  1580          BEQL    30$                     ; Zero is error
                          52 55 3C  07DA  1581          MOVZWL  R5, R2                  ; Isolate index
                53 00000000'9F D0  07DD  1582          MOVL    @#SCS$GL_CDL, R3        ; Pick up pointer to CDL
                      53 6342 D0  07E4  1583          MOVL    (R3)[R2], R3            ; Obtain pointer to selected CDT
                   18 A3 55 D1  07E8  1584          CMPL    R5, CDT$L_LCONID(R3)    ; CONID's match?
                             0F 12  07EC  1585          BNEQ    30$                     ; No, failure
                          50 01 D0  07EE  1586          MOVL    #1, R0                  ; Yes, success
                          51 54 D0  07F1  1587          MOVL    R4, R1                  ; Return pointer to entry
                             8E D5  07F4  1588 20$:     TSTL    (SP)+                   ; Get rid of temp
                          52 8ED0  07F6  1589          POPL    R2                      ; Recover R2
                          54 8E 7D  07F9  1590          MOVQ    (SP)+,R4                ; Restore registers
                             05 07FC  1591          RSB
                                07FD  1592
                50 20A4 8F 3C  07FD  1593 30$:     MOVZWL  #SS$_NOSUCHOBJ, R0      ; Could not find listener *** TEMP ****
                             53 D4  0802  1594          CLRL    R3                      ; No chance of spurious CDT
                             EE 11  0804  1595          BRB     20$                     ;  and out
```

```
                     0806  1597          .SBTTL  SCS$REMOVE
                     0806  1598 ;++
                     0806  1599 ;
                     0806  1600 ; FUNCTIONAL DESCRIPTION:
                     0806  1601 ;
                     0806  1602 ;
                     0806  1603 ; CALLING SEQUENCE:
                     0806  1604 ;
                     0806  1605 ;       BSBW    SCS$REMOVE
                     0806  1606 ;
                     0806  1607 ; INPUTS:
                     0806  1608 ;
                     0806  1609 ;       R3 = CDT address
                     0806  1610 ;
                     0806  1611 ; OUTPUTS:
                     0806  1612 ;
                     0806  1613 ;       R0 = SS$_NOSUCHOBJ if could not find entry
                     0806  1614 ;          = SS$_NORMAL if successful
                     0806  1615 ;       R1,R2 destroyed
                     0806  1616 ;       R3 = 0 if successful
                     0806  1617 ;          = CDT if failure
                     0806  1618 ;       R4,R5 preserved
                     0806  1619 ;
                     0806  1620 ; SIDE EFFECTS:
                     0806  1621 ;
                     0806  1622 ;       Directory entry removed from directory.
                     0806  1623 ;
                     0806  1624 ;--
                     0806  1625
                     0806  1626 SCS$REMOVE:
     50   20A4 8F    3C  0806  1627          MOVZWL  #SS$_NOSUCHOBJ, R0      ; Assume failure
52    00000000'9F    DE  080B  1628          MOVAL   @#SCS$GQ_DIRECT, R2     ; Get pointer to directory head
          51   52    D0  0812  1629          MOVL    R2, R1                  ; Copy
          52   62    D0  0815  1630 10$:     MOVL    (R2), R2                ; Get next entry
          52   51    D1  0818  1631          CMPL    R1, R2                  ; End of the line?
               18    13  081B  1632          BEQL    20$                     ; Yes, leave can't find entry
   2C A2    18 A3    D1  081D  1633          CMPL    CDT$L_LCONID(R3), -
                         0822  1634                  SDIR$L_CONID(R2)        ; Match CONID?
               F1    12  0822  1635          BNEQ    10$                     ; No, try next one
          50   62    0F  0824  1636          REMQUE  (R2), R0                ; Yes, remove it
               53    DD  0827  1637          PUSHL   R3                      ; Save CDT addr
              0019   30  0529  1638          BSBW    SCS_DEALNONPAGD         ; Release the pool space
               53 8ED0  082C  1639          POPL    R3                      ; Restore CDT
          50   01    D0  082F  1640          MOVL    #1, R0                  ; Set for success
              FD74   31  0832  1641          BRW     SCS$DEALL_CDT1          ; Get rid of CDT and return
                         0835  1642
                     05  0835  1643 20$:     RSB                             ; Leave
```

```
                                0836  1645 ;
                                0836  1646 ; Misc routines
                                0836  1647 ;
                                0836  1648
                                0836  1649 SCS_ALONONPAGED:
00000000'9F    16               0836  1650          JSB     @#EXE$ALONONPAGED       ; Allocate the pool
      05 50    E8               083C  1651          BLBS    R0, 10$                 ; Skip out if success
50    0124 8F  3C               083F  1652          MOVZWL  #SS$_INSFMEM, R0        ; Set a reconizible error code
               05               0844  1653 10$:     RSB                             ; Leave
                                0845  1654
                                0845  1655 SCS_DEALNONPAGD:
00000000'9F    17               0845  1656          JMP     @#COM$DRVDEALMEM        ; Deallocated
```

```
                     084B  1658          .SBTTL  SCS$RESUMEWAITR
                     084B  1659 ;++
                     084B  1660 ;
                     084B  1661 ; FUNCTIONAL DESCRIPTION:
                     084B  1662 ;
                     084B  1663 ;       Global SCS routine to resume a thread that had been
                     084B  1664 ;       waiting on a resource wait queue.  This routine resumes the thread
                     084B  1665 ;       and upon return from the thread, if the CDRP associated with the
                     084B  1666 ;       resumed thread had a RWAITCNT, then we attempt to start any stalled
                     084B  1667 ;       IRP's that may have been queued to the UCB while we were waiting.
                     084B  1668 ;
                     084B  1669 ; INPUTS:
                     084B  1670 ;
                     084B  1671 ;       R5 => CDRP of thread to resume
                     084B  1672 ;
                     084B  1673 ; OUTPUTS:
                     084B  1674 ;
                     084B  1675 ;       Thread resumed and IRP's started up when appropriate.
                     084B  1676 ;
                     084B  1677 ;       Note all registers R0 - R5 may be modified.
                     084B  1678 ;
                     084B  1679 ;--
                     084B  1680
                     084B  1681 SCS$RESUMEWAITR:                        ; Here we will start driver thread that
                     084B  1682                                         ;  was waiting for resource.
                     084B  1683
                     084B  1684 ;
                     084B  1685 ; We divide the CDRP thread to be restarted into two classes:  Those which
                     084B  1686 ;       do NOT have a RWAITCNT and those which do.  The code path for those
                     084B  1687 ;       without is very simple and appears directly below.  The more complex
                     084B  1688 ;       case of CDRP's with a RWAITCNT begins at label THREAD_HAS_RWAITCNT.
                     084B  1689 ;
                     084B  1690
53   28 A5    D0     084B  1691          MOVL    CDRP$L_RWCPTR(R5),R3    ; R3 => UCB$L_RWAITCNT or R3 = 0.
     08       12     084F  1692          BNEQ    THREAD_HAS_RWAITCNT     ; NEQ implies R3 => UCB$L_RWAITCNT.
53   10 A5    7D     0851  1693          MOVQ    CDRP$L_FR3(R5),R3       ; Restore register context to thread.
     0C B5    16     0855  1694          JSB     @CDRP$L_FPC(R5)         ; Call back driver thread.
              05     0858  1695          RSB                             ; Return to caller.
                     0859  1696
                     0859  1697 THREAD_HAS_RWAITCNT:                    ; Here we have R3 =>  UCB$L_RWAITCNT.
     63       B7     0859  1698          DECW    (R3)                    ; One less CDRP waiting for resources.
     BC A5    DD     085B  1699          PUSHL   CDRP$L_UCB(R5)          ; Save this CDRP'S UCB address on stack.
53   10 A5    7D     085E  1700          MOVQ    CDRP$L_FR3(R5),R3       ; Restore context of waiting IRP.
     0C B5    16     0862  1701          JSB     @CDRP$L_FPC(R5)         ; Call back driver thread.
        55 8ED0     0865  1702          POPL    R5                      ; R5 => UCB.
                     0868  1703
                     0868  1704 ;
                     0868  1705 ; Here we have experienced return from a resumed thread that maintains a
                     0868  1706 ;       UCB$W_RWAITCNT.  Now we see if we can start up any IRP's that may have
                     0868  1707 ;       backed on this UCB.  These would be new IRP's that had not yet gotten
                     0868  1708 ;       into STARTIO.
                     0868  1709 ;
                     0868  1710 ; NOTE - This entrypoint SCS$UNSTALLUCB may be called independently to
                     0868  1711 ;       unstall UCB's that may have had non-zero RWAITCNT's for other
                     0868  1712 ;       reasons.
                     0868  1713 ;
                     0868  1714
```

```
                          0868  1715 SCS$UNSTALLUCB:
          56 A5    B5     0868  1716 10$:     TSTW    UCB$W_RWAITCNT(R5)      ; Can we startup backed up IRP's?
             1A    12     086B  1717          BNEQ    20$                     ; NEQ implies NO, so branch around.
       53    4C B5 0F     086D  1718          REMQUE  @UCB$L_IOQFL(R5),R3     ; Remove 1st (if any) IRP from queue.
             14    1D     0871  1719          BVS     20$                     ; VS implies no backed up IRP's.
             55    DD     0873  1720          PUSHL   R5                      ; Remember UCB address on stack.
                   AA     0875  1721          BICW    #UCB$M_CANCEL!-
                          0876  1722           UCB$M_TIMOUT,-
64 A5    0048 8F          0876  1723          UCB$W_STS(R5)                   ; Clear cancel and time out
                          087B  1724
                          087B  1725          ASSUME  DDT$L_START     EQ      0
    51   0088 D5    D0    087B  1726          MOVL    @UCB$L_DDT(R5),R1       ; R1 => Start io routine.
             61    16     0880  1727          JSB     (R1)                    ;START I/O OPERATION
             55 8ED0      0882  1728          POPL    R5                      ; Restore R5 => UCB of interest.
             E1    11     0885  1729          BRB     10$                     ; Loop back to start more backed up IRP's.
                          0887  1730
                   05     0887  1731 20$:     RSB
```

```
0888  1733          .SBTTL  SCS$LKP_RDTCDRP
0888  1734
0888  1735 ;++
0888  1736 ;
0888  1737 ; FUNCTIONAL DESCRIPTION:
0888  1738 ;
0888  1739 ;       Search RDT for CDRPs with CDT address matching the one in R3
0888  1740 ;       upon entry.  For each CDRP found, call user-supplied action
0888  1741 ;       routine.
0888  1742 ;
0888  1743 ; CALLING SEQUENCE:
0888  1744 ;
0888  1745 ;       JSB     SCS$LKP_RDTCDRP
0888  1746 ;
0888  1747 ; INPUTS:
0888  1748 ;
0888  1749 ;       R0 -->  Addr of action routine
0888  1750 ;       R3 -->  Addr of CDT
0888  1751 ;
0888  1752 ; OUTPUTS:
0888  1753 ;
0888  1754 ;       R0 -->  status code
0888  1755 ;               success ==> RDT contains no CDRPs with matching CDT
0888  1756 ;               failure ==> RDT contains one or more CDRPs with matching CDT
0888  1757 ;       R1      not altered or preserved by this routine; may be used as a
0888  1758 ;               communication mechanism between action routine and caller of
0888  1759 ;               this routine
0888  1760 ;       R2 - R5 preserved
0888  1761 ;
0888  1762 ; ACTION ROUTINE INPUTS:
0888  1763 ;
0888  1764 ;       R1 -->  unchanged from value set by this routine's caller
0888  1765 ;       R3 -->  Addr of CDT
0888  1766 ;       R4 -->  unchanged from value set by this routine's caller
0888  1767 ;       R5 -->  Addr of CDRP
0888  1768 ;
0888  1769 ; ACTION ROUTINE OUTPUTS:
0888  1770 ;
0888  1771 ;       R0      scratch
0888  1772 ;       R1      value to be returned to this routine's caller
0888  1773 ;       R2      scratch
0888  1774 ;       R3      Addr of CDT; MUST be preserved
0888  1775 ;       R4 - R5 scratch
0888  1776 ;
0888  1777 ;       N.B. More likely than not R4 is a PDT address which also must be
0888  1778 ;       preserved.  However, this routine makes no such assumption.
0888  1779 ;
0888  1780 ; STACK USAGE:
0888  1781 ;
0888  1782 ;       This routine manipulates the stack in a rather odd mannor.  Therefore,
0888  1783 ;       the following map of the stack, as it is at the time of the JSB to the
0888  1784 ;       action routine, is provided to describe how the stack is used.
0888  1785 ;
0888  1786 ;       !-----------------------------------------------------------!
0888  1787 ;       !                       Saved index R0                      ! :(SP)
0888  1788 ;       !-----------------------------------------------------------!
0888  1789 ;       !                       Saved list base R2                  !
```

```
                     0888  1790 ;       |-------------------------------------------------------------|
                     0888  1791 ;       | r0                Action routine address                    |
                     0888  1792 ;       |-------------------------------------------------------------|
                     0888  1793 ;       | r1                "found-one" flag                      R0  |
                     0888  1794 ;       |-------------------------------------------------------------|
                     0888  1795 ;       | r2                Saved register 2                      R2  |
                     0888  1796 ;       |-------------------------------------------------------------|
                     0888  1797 ;       | r4                Saved register 4                      R4  |
                     0888  1798 ;       |-------------------------------------------------------------|
                     0888  1799 ;       | r5                Saved register 5                      R5  |
                     0888  1800 ;       |-------------------------------------------------------------|
                     0888  1801 ;         ^                                                           ^
                     0888  1802 ;         |-as saved by PUSHR                  as restored by POPR-|
                     0888  1803 ;
                     0888  1804 ;--
                     0888  1805
                     0888  1806 SCS$LKP_RDTCDRP:
                37   BB  0888  1807          PUSHR   #^M<R0,R1,R2,R4,R5>     ; Save registers, make place for saved
      04 AE      01   D0  088A  1808          MOVL    #1, 4(SP)               ; "found-one" flag, & init. none found.
50  00000000'9F      3C  088E  1809          MOVZWL  @#SCS$GW_RDTCNT, R0     ; Pick up number of entries.
52  00000000'9F      D0  0895  1810          MOVL    @#SCS$GL_RDT, R2        ; Pick up pointer to list.
            52     DD   089C  1811          PUSHL   R2                      ; Save that on the stack.
                     089E  1812          ASSUME  RD$C_LENGTH EQ 8
55    F8 A240      7E   089E  1813 10$:     MOVAQ   B^-8(R2)[R0], R5        ; Pick up next entry.
      04 A5        B5   08A3  1814          TSTW    RD$W_STATE(R5)          ; Is it in use?
            19     13   08A6  1815          BEQL    20$                     ; No, try another.
      55    65     D0   08A8  1816          MOVL    RD$L_CDRP(R5), R5       ; Pick up CDRP pointer.
            14     13   08AB  1817          BEQL    20$                     ; None, try another.
   53    24 A5     D1   08AD  1818          CMPL    CDRP$L_CDT(R5), R3      ; CDT's match?
            0E     12   08B1  1819          BNEQ    20$                     ; No, try some other.
      08 AE        D4   08B3  1820          CLRL    8(SP)                   ; Raise "found-one" flag.
            50     DD   08B6  1821          PUSHL   R0                      ; Save list index.
      08 BE        16   08B8  1822          JSB     @8(SP)                  ; Call action routine.
            50 8ED0  08BB  1823          POPL    R0                      ; Restore list index.
   52     6E     D0   08BE  1824          MOVL    (SP), R2                ; Restore list base address.
      DA 50        F5   08C1  1825 20$:     SOBGTR  R0, 10$                 ; Loop until done.
5E    08 AE       9E   08C4  1826          MOVAB   8(SP), SP               ; Cleanup the stack.
            35     BA   08C8  1827          POPR    #^M<R0,R2,R4,R5>        ; Restore saved "found-one" flag and
                   05   08CA  1828          RSB                             ; other preserved registers.
```

```
                                .SBTTL  SCS$LKP_RDTWAIT

                        ;++
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       Search RSPID wait queue for next CDRP with CDRP$L_CDT that
                        ;       matches CDT specified in R3.  For each CDRP found, call user-
                        ;       supplied action routine.
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       JSB     SCS$LKP_RDTWAIT
                        ;
                        ; INPUTS:
                        ;
                        ;       R0 -->  Addr of action routine
                        ;       R3 -->  Addr of CDT
                        ;
                        ; OUTPUTS:
                        ;
                        ;       R0 -->  status code
                        ;               success ==> RDT contains no CDRPs with matching CDT
                        ;               failure ==> RDT contains one or more CDRPs with matching CDT
                        ;       R1      not altered or preserved by this routine; may be used as a
                        ;               communication mechanism between action routine and caller of
                        ;               this routine
                        ;       R2 - R5 preserved
                        ;
                        ; ACTION ROUTINE INPUTS:
                        ;
                        ;       R1 -->  unchanged from value set by this routine's caller
                        ;       R3 -->  Addr of CDT
                        ;       R4 -->  unchanged from value set by this routine's caller
                        ;       R5 -->  Addr of CDRP
                        ;
                        ; ACTION ROUTINE OUTPUTS:
                        ;
                        ;       R0      scratch
                        ;       R1      value to be returned to this routine's caller
                        ;       R2      scratch
                        ;       R3      Addr of CDT; MUST be preserved
                        ;       R4 - R5 scratch
                        ;
                        ;       N.B. More likely than not R4 is a PDT address which also must be
                        ;       preserved.  However, this routine makes no such assumption.
                        ;
                        ;  STACK USAGE:
                        ;       This routine manipulates the stack in a rather odd mannor.  Therefore,
                        ;       the following map of the stack, as it is at the time of the JSB to the
                        ;       action routine, is provided to describe how the stack is used.
                        ;
                        ;       !-----------------------------------------------------------!
                        ;       !                Saved next entry address R5                ! :(SP)
                        ;       !-----------------------------------------------------------!
                        ;       !                 Saved end of list address                 !
                        ;       !-----------------------------------------------------------!
```

```
                                08CB  1887 ;       |  r0                  Action routine address                    |
                                08CB  1888 ;       |------------------------------------------------------------|
                                08CB  1889 ;       |  r1                  "found-one" flag                    R0    |
                                08CB  1890 ;       |------------------------------------------------------------|
                                08CB  1891 ;       |  r2                  Saved register 2                    R2    |
                                08CB  1892 ;       |------------------------------------------------------------|
                                08CB  1893 ;       |  r4                  Saved register 4                    R4    |
                                08CB  1894 ;       |------------------------------------------------------------|
                                08CB  1895 ;       |  r5                  Saved register 5                    R5    |
                                08CB  1896 ;       |------------------------------------------------------------|
                                08CB  1897 ;         ^                                                         ^
                                08CB  1898 ;         |-as saved by PUSHR                  as restored by POPR-|
                                08CB  1899 ;
                                08CB  1900 ;--
                                08CB  1901
                                08CB  1902 SCS$LKP_RDTWAIT:
                      37    BB  08CB  1903          PUSHR   #^M<R0,R1,R2,R4,R5>     ; Save registers, make place for saved
                   04 AE    01  D0  08CD  1904          MOVL    #1, 4(SP)           ; "found-one" flag, & init. none found.
                55 00000000'9F  D0  08D1  1905          MOVL    @#SCS$GL_RDT, R5    ; Pick up pointer to list.
                   55    E8 A5  DE  08D8  1906          MOVAL   RDT$L_WAITFL(R5), R5 ; Point at wait queue head.
                      55    DD  08DC  1907          PUSHL   R5                  ; Save end of queue information.
                         55 65  D0  08DE  1908 30$:   MOVL    (R5), R5            ; Pick up next entry.
                         6E 55  D1  08E1  1909 33$:   CMPL    R5, (SP)            ; End of queue?
                            13  13  08E4  1910          BEQL    60$                 ; Yes, quit.
                   53    24 A5  D1  08E6  1911          CMPL    CDRP$L_CDT(R5), R3  ; CDT's match?
                            F2  12  08EA  1912          BNEQ    30$                 ; No, try again.
                      08 AE    D4  08EC  1913          CLRL    8(SP)               ; Raise "found-one" flag.
                         65    DD  08EF  1914          PUSHL   (R5)                ; Save next entry address.
                      08 BE    16  08F1  1915          JSB     @8(SP)              ; Call action routine.
                         55 8ED0  08F4  1916          POPL    R5                  ; Restore address of next entry.
                            E8   11  08F7  1917          BRB     33$                 ; Since R5 already has the address
                                     08F9  1918                                  ; of the next entry, skip the queue
                                     08F9  1919                                  ; traversal step of this loop.
                5E  08 AE    9E  08F9  1920 60$:   MOVAB   8(SP), SP           ; Cleanup the stack.
                      35    BA  08FD  1921          POPR    #^M<R0,R2,R4,R5>    ; Restore saved "found-one" flag and
                            05  08FF  1922          RSB                         ; other preserved registers.
```

```
0900  1924          .SBTTL  SCS$LKP_MSGWAIT - Scan message wait queues
0900  1925
0900  1926 ;++
0900  1927 ;
0900  1928 ; SCS$LKP_MSGWAIT - Scan message wait queues
0900  1929 ;
0900  1930 ; Functional description:
0900  1931 ;
0900  1932 ;       Search send credit wait and message buffer wait queues for CDRPs with
0900  1933 ;       CDRP$L_CDT that matches CDT specified in R3.  For each CDRP found,
0900  1934 ;       call user-supplied action routine.
0900  1935 ;
0900  1936 ;       This routine completely searches the message buffer wait queue before
0900  1937 ;       searching the send credit wait queue.  Both queues are searched front
0900  1938 ;       to back.  This is the reverse of the order in which the PADRIVER adds
0900  1939 ;       entries to these queues.  This searching stratagy has been choosen to
0900  1940 ;       maximize the possibility of locating CDRPs in the order in which the
0900  1941 ;       PADRIVER processed them.  This is very important to the connection
0900  1942 ;       manager which is trying to assure sequential delivery of specific
0900  1943 ;       message types to remote systems.  The lock manager, in turn, depends
0900  1944 ;       upon this sequential delivery promise.
0900  1945 ;
0900  1946 ;       DO NOT break this attempt at sequential location of CDRPs.  Do not
0900  1947 ;       change the ordering of queue insertion (in the PADRIVER or other port
0900  1948 ;       drivers used to maintain VAX-to-VAX connections) or the order of queue
0900  1949 ;       scanning here.
0900  1950 ;
0900  1951 ; Calling sequence:
0900  1952 ;
0900  1953 ;       JSB     SCS$LKP_MSGWAIT
0900  1954 ;
0900  1955 ; INPUTS:
0900  1956 ;
0900  1957 ;       R0 -->  Addr of action routine
0900  1958 ;       R3 -->  Addr of CDT
0900  1959 ;       R4 -->  Addr of PDT
0900  1960 ;
0900  1961 ; OUTPUTS:
0900  1962 ;
0900  1963 ;       R0 -->  status code
0900  1964 ;               success ==> RDT contains no CDRPs with matching CDT
0900  1965 ;               failure ==> RDT contains one or more CDRPs with matching CDT
0900  1966 ;       R1      not altered or preserved by this routine; may be used as a
0900  1967 ;               communication mechanism between action routine and caller of
0900  1968 ;               this routine
0900  1969 ;       R2 - R5 preserved
0900  1970 ;
0900  1971 ; ACTION ROUTINE INPUTS:
0900  1972 ;
0900  1973 ;       R1 -->  unchanged from value set by this routine's caller
0900  1974 ;       R3 -->  Addr of CDT
0900  1975 ;       R4 -->  unchanged from value set by this routine's caller
0900  1976 ;       R5 -->  Addr of CDRP
0900  1977 ;
0900  1978 ; ACTION ROUTINE OUTPUTS:
0900  1979 ;
0900  1980 ;       R0      scratch
```

```
                     0900  1981 ;       R1      value to be returned to this routine's caller
                     0900  1982 ;       R2      scratch
                     0900  1983 ;       R3      Addr of CDT; MUST be preserved
                     0900  1984 ;       R4 - R5 scratch
                     0900  1985 ;
                     0900  1986 ;   STACK USAGE:
                     0900  1987 ;       This routine manipulates the stack in a rather odd mannor.  Therefore,
                     0900  1988 ;       the following map of the stack, as it is at the time of the JSB to the
                     0900  1989 ;       action routine, is provided to describe how the stack is used.
                     0900  1990 ;
                     0900  1991 ;           |-------------------------------------------------------------|
                     0900  1992 ;           |                  Saved next entry address R5                | :(SP)
                     0900  1993 ;           |-------------------------------------------------------------|
                     0900  1994 ;           |                  Saved end of list address                  |
                     0900  1995 ;           |-------------------------------------------------------------|
                     0900  1996 ;           | r0               Action routine address                     |
                     0900  1997 ;           |-------------------------------------------------------------|
                     0900  1998 ;           | r1                  "found-one" flag                 R0    |
                     0900  1999 ;           |-------------------------------------------------------------|
                     0900  2000 ;           | r2                    Saved register 2               R2    |
                     0900  2001 ;           |-------------------------------------------------------------|
                     0900  2002 ;           | r4                  Saved register 4 (PDT)           R4    |
                     0900  2003 ;           |-------------------------------------------------------------|
                     0900  2004 ;           | r5                    Saved register 5               R5    |
                     0900  2005 ;           |-------------------------------------------------------------|
                     0900  2006 ;             ^                                                     ^
                     0900  2007 ;             |-as saved by PUSHR                  as restored by POPR-|
                     0900  2008 ;
                     0900  2009 ;--
                     0900  2010
                     0900  2011 SCS$LKP_MSGWAIT:
           37    BB  0900  2012           PUSHR   #^M<R0,R1,R2,R4,R5>     ; Save registers, make place for saved
     04 AE   01    D0  0902  2013           MOVL    #1, 4(SP)               ; "found-one" flag, & init. none found.
55   00AC C4    DE  0906  2014           MOVAL   PDT$L_WAITQFL(R4), R5   ; Point at msg. buf. wait queue header.
           55    DD  090B  2015           PUSHL   R5                      ; Save end of queue information.
     55    65    D0  090D  2016 30$:      MOVL    (R5), R5                ; Pick up next entry.
     6E    55    D1  0910  2017 33$:      CMPL    R5, (SP)                ; End of queue?
           13    13  0913  2018           BEQL    60$                     ; Yes, quit.
53     24 A5    D1  0915  2019           CMPL    CDRP$L_CDT(R5), R3      ; CDT's match?
           F2    12  0919  2020           BNEQ    30$                     ; No, try again.
        08 AE    D4  091B  2021           CLRL    8(SP)                   ; Raise "found-one" flag.
           65    DD  091E  2022           PUSHL   (R5)                    ; Save next entry address.
        08 BE    16  0920  2023           JSB     @8(SP)                  ; Call action routine.
           55 8ED0  0923  2024           POPL    R5                      ; Restore address of next entry.
           E8    11  0926  2025           BRB     33$                     ; Since R5 already has the address
                     0928  2026                                           ; of the next entry, skip the queue
                     0928  2027                                           ; traversal step of this loop.
55     38 A3    DE  0928  2028 60$:      MOVAL   CDT$L_CRWAITQFL(R3), R5 ; Point at send credit wait queue head.
   6E    55    D0  092C  2029           MOVL    R5, (SP)                ; Save end of queue information.
   55    65    D0  092F  2030 80$:      MOVL    (R5), R5                ; Pick up next entry.
   6E    55    D1  0932  2031 83$:      CMPL    R5, (SP)                ; End of queue?
           13    13  0935  2032           BEQL    100$                    ; Yes, quit.
53     24 A5    D1  0937  2033           CMPL    CDRP$L_CDT(R5), R3      ; CDT's match?
           F2    12  093B  2034           BNEQ    80$                     ; No, try again.
        08 AE    D4  093D  2035           CLRL    8(SP)                   ; Raise "found-one" flag.
           65    DD  0940  2036           PUSHL   (R5)                    ; Save next entry address.
        08 BE    16  0942  2037           JSB     @8(SP)                  ; Call action routine.
```

```
                55 8ED0  0945  2038          POPL    R5                      ; Restore address of next entry.
                E8   11  0948  2039          BRB     83$                     ; Since R5 already has the address
                         094A  2040                                          ; of the next entry, skip the queue
                         094A  2041                                          ; traversal step of this loop.
                         094A  2042
          5E    08 AE    9E  094A  2043 100$:  MOVAB   8(SP), SP              ; Cleanup the stack.
                35   BA  094E  2044          POPR    #^M<R0,R2,R4,R5>        ; Restore saved "found-one" flag and
                     05  0950  2045          RSB                             ; other preserved registers.
                         0951  2046
```

```
                     0951  2048          .SBTTL  SCS$NEW_SB - New/Reused System Block Available for Polling
                     0951  2049
                     0951  2050 ;++
                     0951  2051 ;
                     0951  2052 ; FUNCTIONAL DESCRIPTION:
                     0951  2053 ;
                     0951  2054 ;       This routine is called whenever SCS builds a new system block
                     0951  2055 ;       (SB) or reuses an existing SB with a new software incarnation
                     0951  2056 ;       number.  This routine initializes the poller fields in the
                     0951  2057 ;       system block.
                     0951  2058 ;
                     0951  2059 ; CALLING SEQUENCE:
                     0951  2060 ;
                     0951  2061 ;       JSB     SCS$NEW_SB
                     0951  2062 ;       IPL must be at IPL$_SCS, IPL$_TIMER
                     0951  2063 ;
                     0951  2064 ; INPUT PARAMETERS:
                     0951  2065 ;
                     0951  2066 ;       R2 contains the address of the system block of the new system.
                     0951  2067 ;
                     0951  2068 ; OUTPUT PARAMETERS:
                     0951  2069 ;
                     0951  2070 ;       NONE
                     0951  2071 ;
                     0951  2072 ; COMPLETION CODES:
                     0951  2073 ;
                     0951  2074 ;       NONE
                     0951  2075 ;
                     0951  2076 ; SIDE EFFECTS:
                     0951  2077 ;
                     0951  2078 ;       R0 and R1 are destroyed
                     0951  2079 ;
                     0951  2080 ;--
                     0951  2081
                     0951  2082 ; Use of SB$W_TIMEOUT:
                     0951  2083 ;
                     0951  2084 ;       SB$W_TIMEOUT indicates that state of polling on a particular node
                     0951  2085 ;
                     0951  2086 ;               >0  : Seconds remaining until polling is triggered
                     0951  2087 ;               =0  : Ready to be polled
                     0951  2088 ;               -1  : Polling in progress, reset timer on completion
                     0951  2089 ;               -2  : Polling in progress, set to 0 on completion to repoll
                     0951  2090 ;
                     0951  2091 ; Use of SB$B_ENBMSK:
                     0951  2092 ;
                     0951  2093 ;       Each process name to be polled is assigned a bit in the SB$B_ENBMSK
                     0951  2094 ;       array.  If the bit is set, then polling for the process name is
                     0951  2095 ;       enabled; if the bit is clear, polling for the name is disabled.
                     0951  2096 ;       This allows system by system control over which processes are
                     0951  2097 ;       polled.
                     0951  2098 ;
                     0951  2099
                     0951  2100 SCS$NEW_SB:
           3C   BB   0951  2101          PUSHR   #^M<R2,R3,R4,R5>        ; Save registers
        58 A2   B4   0953  2102          CLRW    SB$W_TIMEOUT(R2)        ; Trigger for immediate polling
00000000'GF    02   28 0956  2103          MOVC3   #SB$S_ENBMSK, -         ; Initialize polling mask
        5A A2        095D  2104                  G^SCS$GA_DFLTMSK, -
```

```
                095F  2105                    SB$B_ENBMSK(R2)
3C      BA      095F  2106          POPR      #^M<R2,R3,R4,R5>        ; Restore registers
        05      0961  2107          RSB
                0962  2108
```

```
                        0962  2110          .SBTTL  SCS$POLL_PROC - Declare a process name to the poller
                        0962  2111
                        0962  2112 ;++
                        0962  2113 ;
                        0962  2114 ; FUNCTIONAL DESCRIPTION:
                        0962  2115 ;
                        0962  2116 ;       This routine is called to declare a process to be polled.
                        0962  2117 ;       The routine builds a Poller Process block and links it.
                        0962  2118 ;       An unused bit is allocated and assigned to this process.
                        0962  2119 ;
                        0962  2120 ;       Polling for the process is NOT enabled.
                        0962  2121 ;
                        0962  2122 ;       Note: Things could be simplified by omitting this routine
                        0962  2123 ;       and wiring the necessary (and short) list of processes to
                        0962  2124 ;       be polled into this module.
                        0962  2125 ;
                        0962  2126 ; CALLING SEQUENCE:
                        0962  2127 ;
                        0962  2128 ;       JSB     SCS$POLL_PROC
                        0962  2129 ;       IPL must be at IPL$_SCS, IPL$_TIMER
                        0962  2130 ;
                        0962  2131 ; INPUT PARAMETERS:
                        0962  2132 ;
                        0962  2133 ;       R0 contains the address of the notification routine
                        0962  2134 ;       R1 contains the context to be passed to the notification routine
                        0962  2135 ;       R2 contains the address of the process name
                        0962  2136 ;
                        0962  2137 ; OUTPUT PARAMETERS:
                        0962  2138 ;
                        0962  2139 ;       R1 contains address of the SPPB (to be used as context to other calls)
                        0962  2140 ;
                        0962  2141 ; COMPLETION CODES:
                        0962  2142 ;
                        0962  2143 ;       R0 contains status
                        0962  2144 ;
                        0962  2145 ; SIDE EFFECTS:
                        0962  2146 ;
                        0962  2147 ;       R0 is destroyed
                        0962  2148 ;       New data structures are built
                        0962  2149 ;
                        0962  2150 ;--
                        0962  2151
                        0962  2152 SCS$POLL_PROC:
                  3F    BB    0962  2153          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers
55  00000000'GF   3E    0964  2154          MOVAW   G^SCS$GW_NEXTBIT,R5     ; Address of next bit to allocate
        10  65    B1    096B  2155          CMPW    (R5),#SB$S_ENBMSK*8     ; Any bits left?
            33    18    096E  2156          BGEQ    20$                     ; Branch if all bits used
        51  28    3C    0970  2157          MOVZWL  #SPPB$K_LENGTH,R1       ; Size of a SPPB
          FEC0    30    0973  2158          BSBW    SCS_ALONONPAGED         ; Allocate a SPPB
        2C 50     E9    0976  2159          BLBC    R0,30$                  ; Exit on error
    08 A2   51    B0    0979  2160          MOVW    R1,SPPB$W_SIZE(R2)      ; Store size
                        097D  2161          ASSUME  SPPB$B_SUBTYP,EQ,SPPB$B_TYPE+1
 0A A2  0860 8F    B0    097D  2162          MOVW    #DYN$C_SCS! -           ; Store type and subtype
                        0983  2163                  <DYN$C_SCS_SPPB@8>, -
                        0983  2164                  SPPB$B_TYPE(R2)
                        0983  2165          ASSUME  SPPB$L_RTN+4,EQ,SPPB$L_CTX
    1C A2   8E    7D    0983  2166          MOVQ    (SP)+,SPPB$L_RTN(R2)    ; Store routine address and context
```

```
                24 A2    65    B0  0987  2167          MOVW    (R5),SPPB$W_BIT(R2)      ; Store allocated mask bit
                         65    B6  098B  2168          INCW    (R5)                     ; Mark bit used
                         52    DD  098D  2169          PUSHL   R2                       ; Save SPPB address
             0C A2    04 BE    10  28  098F  2170          MOVC3   #SPPB$S_PROCNAM, -       ; Store process name
                                   0995  2171                  @4(SP),SPPB$B_PROCNAM(R2)
        00000000'GF    00 BE    0E  0995  2172          INSQUE  @(SP),G^SCS$GQ_POLL      ; Link into head of list of SPPB's
                   50    01    D0  099D  2173          MOVL    S^#SS$_NORMAL,R0         ; Success
                         3E    BA  09A0  2174  10$:    POPR    #^M<R1,R2,R3,R4,R5>
                               05  09A2  2175          RSB
                         50    D4  09A3  2176  20$:    CLRL    R0                       ; Set failure return
                      5E 04    C0  09A5  2177  30$:    ADDL2   #4,SP                    ; Balance stack with main path
                               F6  09A8  2178          BRB     10$
                                   09AA  2179
```

```
                                09AA  2181          .SBTTL  SCS$POLL_MODE - Enable/Disable polling for a process
                                09AA  2182
                                09AA  2183 ;++
                                09AA  2184 ;
                                09AA  2185 ; FUNCTIONAL DESCRIPTION:
                                09AA  2186 ;
                                09AA  2187 ;       This routine enable or disable polling for a process on
                                09AA  2188 ;       either a specified system or a global basis.  If polling
                                09AA  2189 ;       is enabled on any existing system, a poll is requested.
                                09AA  2190 ;
                                09AA  2191 ; CALLING SEQUENCE:
                                09AA  2192 ;
                                09AA  2193 ;       JSB     SCS$POLL_MODE
                                09AA  2194 ;       IPL must be at IPL$_SCS, IPL$_TIMER
                                09AA  2195 ;
                                09AA  2196 ; INPUT PARAMETERS:
                                09AA  2197 ;
                                09AA  2198 ;       R0 contains 0 to disable or 1 to enable polling
                                09AA  2199 ;       R1 contains the address of the SPPB block
                                09AA  2200 ;       R2 contains the address of the System ID or 0 (for global)
                                09AA  2201 ;
                                09AA  2202 ; OUTPUT PARAMETERS:
                                09AA  2203 ;
                                09AA  2204 ;       NONE
                                09AA  2205 ;
                                09AA  2206 ; COMPLETION CODES:
                                09AA  2207 ;
                                09AA  2208 ;       R0 contains status
                                09AA  2209 ;
                                09AA  2210 ; SIDE EFFECTS:
                                09AA  2211 ;
                                09AA  2212 ;       All registers preserved
                                09AA  2213 ;
                                09AA  2214 ;--
                                09AA  2215
                                09AA  2216 SCS$POLL_MODE:
                          3E    BB  09AA  2217          PUSHR   #^M<R1,R2,R3,R4,R5>
                      53  50    D0  09AC  2218          MOVL    R0,R3                   ; Set/clear flag
                  54    24    A1  09AF  2219          MOVZWL  SPPB$W_BIT(R1),R4       ; Process' bit number
                          52    D5  09B3  2220          TSTL    R2                      ; Is system name specified?
                          1F    12  09B5  2221          BNEQ    20$                     ; Branch if yes
00000000'GF   01    54    53    F0  09B7  2222          INSV    R3,R4,#1,-              ; Set bit in default mask
                                09C0  2223                  G^SCS$GA_DFLTMSK
          50    00000000'GF   DE  09C0  2224          MOVAL   G^SCS$GQ_CONFIG,R0      ; Address of SB listhead
                      51    60    D0  09C7  2225          MOVL    (R0),R1                 ; Address of first SB
                      50    51    D1  09CA  2226 10$:     CMPL    R1,R0                   ; End of list?
                            14    13  09CD  2227          BEQL    30$                     ; Branch if done
                            18    10  09CF  2228          BSBB    100$                    ; Handle a system
                      51    61    D0  09D1  2229          MOVL    SB$L_FLINK(R1),R1       ; Link to next SB
                            F4    11  09D4  2230          BRB     10$                     ; Iterate
                                09D6  2231
                      51    52    D0  09D6  2232 20$:     MOVL    R2,R1                   ; Address of system ID
                            52    D4  09D9  2233          CLRL    R2                      ; No output buffer
                        FA5D    30  09DB  2234          BSBW    SCS$CONFIG_SYS          ; Lookup system ID
                     05 50      E9  09DE  2235          BLBC    R0,40$                  ; Branch if System ID not found
                            06    10  09E1  2236          BSBB    100$                    ; Enable/disable polling
                      50    01    D0  09E3  2237 30$:     MOVL    S^#SS$_NORMAL,R0        ; Success
```

```
                3E   BA   09E6  2238 40$:    POPR    #^M<R1,R2,R3,R4,R5>
                05   09E8  2239          RSB
                     09E9  2240
                     09E9  2241 ;
                     09E9  2242 ; R3 is set/clear flag
                     09E9  2243 ; R4 is bit number
                     09E9  2244 ; R1 is address of the SB
                     09E9  2245 ;
             13 53   E9   09E9  2246 100$:   BLBC    R3,120$                      ; Branch to disable polling
13 5A A1        54   E2   09EC  2247         BBSS    R4,SB$B_ENBMSK(R1),130$      ; Branch if already enabled
          58 A1      B5   09F1  2248         TSTW    SB$W_TIMEOUT(R1)             ; Check timeout
                04   19   09F4  2249         BLSS    110$                         ; Branch if in progress to retrigger
          58 A1      B4   09F6  2250         CLRW    SB$W_TIMEOUT(R1)             ; Set to 0 to trigger polling
                05   09F9  2251         RSB                                  ; Return
    58 A1       02   AE   09FA  2252 110$:   MNEGW   #2,SB$W_TIMEOUT(R1)      ; Polling in progress, retrigger
                05   09FE  2253         RSB
00 5A A1        54   E5   09FF  2254 120$:   BBCC    R4,SB$B_ENBMSK(R1),130$ ; Disable polling
                05   0A04  2255 130$:   RSB                                  ; Return
                     0A05  2256
```

```
                     0A05  2258          .SBTTL  START_POLL - Start poll of requested processes
                     0A05  2259
                     0A05  2260 ;++
                     0A05  2261 ;
                     0A05  2262 ; FUNCTIONAL DESCRIPTION:
                     0A05  2263 ;
                     0A05  2264 ;       This routine is called as a result of a timeout.  It must
                     0A05  2265 ;       scan the list of system decrementing the timeouts.  If
                     0A05  2266 ;       a system has timed out, a poll of the processes listed
                     0A05  2267 ;       for that system must be begun.
                     0A05  2268 ;
                     0A05  2269 ; CALLING SEQUENCE:
                     0A05  2270 ;
                     0A05  2271 ;       JSB     START_POLL
                     0A05  2272 ;       IPL must be at IPL$_SCS, IPL$_TIMER
                     0A05  2273 ;
                     0A05  2274 ; INPUT PARAMETERS:
                     0A05  2275 ;
                     0A05  2276 ;       R5 is the address of a TQE block
                     0A05  2277 ;       R4 is the address of the SB which is the start of the
                     0A05  2278 ;          circular sequence used to select the next system to
                     0A05  2279 ;          poll.  If R4 does not point to an SB, the sequence
                     0A05  2280 ;          begins with the first SB.
                     0A05  2281 ;
                     0A05  2282 ; OUTPUT PARAMETERS:
                     0A05  2283 ;
                     0A05  2284 ;       R4 is the new SB at which the search sequence is to
                     0A05  2285 ;          start.
                     0A05  2286 ;
                     0A05  2287 ; COMPLETION CODES:
                     0A05  2288 ;
                     0A05  2289 ;       NONE
                     0A05  2290 ;
                     0A05  2291 ; SIDE EFFECTS:
                     0A05  2292 ;
                     0A05  2293 ;       R0,R1,R2,R3 are destroyed
                     0A05  2294 ;
                     0A05  2295 ;--
                     0A05  2296
                     0A05  2297 START_POLL:
52   00000000'GF   DE  0A05  2298          MOVAL   G^SCS$GQ_CONFIG,R2      ; Address system listhead
     50    54     D0  0A0C  2299          MOVL    R4,R0                   ; Position in scan
           53     7C  0A0F  2300          CLRQ    R3                      ; No timed-out system found yet
     51    62     D0  0A11  2301          MOVL    (R2),R1                 ; First SB
     51    52     D1  0A14  2302 10$:     CMPL    R2,R1                   ; All done?
           4D     13  0A17  2303          BEQL    80$                     ; Return
     51    50     D1  0A19  2304          CMPL    R0,R1                   ; At start of new scan?
           05     12  0A1C  2305          BNEQ    20$
     54    53     D0  0A1E  2306          MOVL    R3,R4                   ; Save preceding match
           53     D4  0A21  2307          CLRL    R3                      ; Look for succeeding match
        58 A1     B7  0A23  2308 20$:     DECW    SB$W_TIMEOUT(R1)        ; Decrement timer
           0F     15  0A26  2309          BLEQ    60$                     ; Timed out
     51    61     D0  0A28  2310 30$:     MOVL    SB$L_FLINK(R1),R1       ; Advance to next SB
           E7     11  0A2B  2311          BRB     10$
                      0A2D  2312
           03     BA  0A2D  2313 40$:     POPR    #^M<R0,R1>              ; Restore registers
58 A1 7FFF     8F  B0  0A2F  2314 50$:     MOVW    #^X7FFF,-               ; Set max timeout -- this will be
```

```
                                                     H 5
                     0A35  2315                    SB$W_TIMEOUT(R1)        ;       reduced when a bit is set
                F1   11   0A35  2316              BRB     30$             ; Continue scan
                          0A37  2317
                05   13   0A37  2318 60$:        BEQL    70$             ; Have just timed out
              58 A1  B6   0A39  2319              INCW    SB$W_TIMEOUT(R1)        ; Poll in progress, unmung timer
                EA   12   0A3C  2320              BNEQ    30$             ; Was -2, ignore for now
                          0A3E  2321 ;
                          0A3E  2322 ; here if polling is triggered but not yet begun
                          0A3E  2323 ;
                03   BB   0A3E  2324 70$:        PUSHR   #^M<R0,R1>      ; Save registers
           50   0C A1   9E   0A40  2325              MOVAB   SB$L_PBFL(R1),R0        ; Address of path block queue header
                60   50   D1   0A44  2326              CMPL    R0,(R0)         ; Is there a path block?
                E4   13   0A47  2327              BEQL    40$             ; Branch if no
           5A A1   02   00   3B   0A49  2328              SKPC    #0,#SB$S_ENBMSK, -      ; Any polling bits set?
                                 0A4E  2329                      SB$B_ENBMSK(R1)
                DD   13   0A4E  2330              BEQL    40$             ; Branch if no bits are set
                03   BA   0A50  2331              POPR    #^M<R0,R1>      ; Restore registers
      00000000'8F    51   D1   0A52  2332              CMPL    R1,#SCS$GA_LOCALSB      ; Local system?
                D4   13   0A59  2333              BEQL    50$             ; Wait a long time
                53   D5   0A5B  2334              TSTL    R3              ; Found one yet this pass?
                C9   12   0A5D  2335              BNEQ    30$             ; Branch if yes
                53   51   D0   0A5F  2336              MOVL    R1,R3           ; Remember this one
                54   D4   0A62  2337              CLRL    R4              ; Forget lower precedence type
                C2   11   0A64  2338              BRB     30$             ; Rejoin main flow
                          0A66  2339
                          0A66  2340 ;
                          0A66  2341 ; The scan over all SB's is completed.
                          0A66  2342 ; R3 NEQ 0:
                          0A66  2343 ;       R3 is the first SB ready for polling following the mark
                          0A66  2344 ;       point set during the last pass, or if there is not
                          0A66  2345 ;       such system, R3 is the first SB ready for polling following
                          0A66  2346 ;       the list header.  R4 is 0.
                          0A66  2347 ; R3 EQL 0:
                          0A66  2348 ;       R4 is 0 or the address of the first SB ready for polling
                          0A66  2349 ;       preceding mark point set during the last pass and no SB ready
                          0A66  2350 ;       for polling was found following the mark point.
                          0A66  2351 ;
                          0A66  2352 ; If any systems are ready for polling, initiate polling one
                          0A66  2353 ; of these.
                          0A66  2354 ;
             53   54   C8   0A66  2355 80$:        BISL2   R4,R3           ; Address is in R3
                03   13   0A69  2356              BEQL    90$             ; Branch if no system
                54   63   D0   0A6B  2357              MOVL    SB$L_FLINK(R3),R4       ; Address of scan restart point
           14 A5   54   D0   0A6E  2358 90$:        MOVL    R4,TQE$L_FR4(R5)        ; Save restart point
                71   13   0A72  2359              BEQL    140$            ; Branch if no system ready for polling
                55   DD   0A74  2360              PUSHL   R5              ; Save address of TQE
        51      00000000'GF   3C   0A76  2361              MOVZWL  G^SCS$GW_NEXTBIT,R1     ; Number of possible SPPBs
                51   04   C4   0A7D  2362              MULL2   #4,R1           ; One longword per SPPB
                51   1C   C0   0A80  2363              ADDL2   #SPNB$C_HDRSIZ+4,R1     ; Space for listhead and trailer
                53   DD   0A83  2364              PUSHL   R3              ; Save SB address
                   FDAE   30   0A85  2365              BSBW    SCS_ALONONPAGED         ; Get pool
                53 8ED0   0A88  2366              POPL    R3              ; Restore SB address
              54   50   E9   0A8B  2367              BLBC    R0,130$         ; Can't, so exit nicely
                3F   BB   0A8E  2368              PUSHR   #^M<R0,R1,R2,R3,R4,R5>
62   51   00   62   00   2C   0A90  2369              MOVC5   #0,(R2),#0,R1,(R2)      ; Zero allocated block
                3F   BA   0A96  2370              POPR    #^M<R0,R1,R2,R3,R4,R5>
           08 A2   51   B0   0A98  2371              MOVW    R1,SPNB$W_SIZE(R2)      ; Store size
```

```
                                                     1  5
                 15 A2    B4   0A9C  2372          CLRW    SPNB$W_REFC(R2)          ; Init reference count
                              0A9F  2373          ASSUME  SPNB$B_SUBTYP EQ SPNB$B_TYPE+1
          0A A2    0960 8F    B0   0A9F  2374          MOVW    #DYN$C_SCS!-
                              0AA5  2375                  <DYN$C_SCS_SPNB@8>,SPNB$B_TYPE(R2) ; Set type
             0C A2    53    D0   0AA5  2376          MOVL    R3,SPNB$L_SB(R2)         ; Address of SB
           10 A2     E6'AF   9E   0AA9  2377          MOVAB   B^300$,SPNB$L_ROUTINE(R2) ; Address of local notification routine
             54     18 A2    9E   0AAE  2378          MOVAB   SPNB$B_NAMLST(R2),R4     ; Address data area
        55    00000000'GF    DE   0AB2  2379          MOVAL   G^SCS$GQ_POLL,R5         ; Poller process list head
                 50    65    D0   0AB9  2380          MOVL    (R5),R0                  ; Address of first SPPB
                 50    55    D1   0ABC  2381 100$:    CMPL    R5,R0                    ; All done?
                       15    13   0ABF  2382          BEQL    120$                     ; Branch if yes
                 51    24 A0    3C   0AC1  2383          MOVZWL  SPPB$W_BIT(R0),R1        ; Get process bit number
             07 5A A3    51    E1   0AC5  2384          BBC     R1,SB$B_ENBMSK(R3),110$ ; Branch if poll not enabled
                 84    0C A0    9E   0ACA  2385          MOVAB   SPPB$B_PROCNAM(R0), -    ; Save address of process name
                              0ACE  2386                  (R4)+
                    15 A2    B6   0ACE  2387          INCW    SPNB$W_REFC(R2)          ; Increment reference count
                 50    60    D0   0AD1  2388 110$:    MOVL    SPPB$L_FLINK(R0),R0      ; Advance to next process
                       E6    11   0AD4  2389          BRB     100$                     ; Iterate over all SPPBs
                              0AD6  2390
                       84    D4   0AD6  2391 120$:    CLRL    (R4)+                    ; List end mark
                              0AD8  2392
                              0AD8  2393 ;
                              0AD8  2394 ; call SCS$DIRECTORY_LOOKUP to do lookups
                              0AD8  2395 ;
                              0AD8  2396 ;       R3 : address of SB
                              0AD8  2397 ;       R2 : address of allocated argument list
                              0AD8  2398 ;
                    58 A3    B7   0AD8  2399          DECW    SB$W_TIMEOUT(R3)         ; Set timeout to -1 to flag scan
                       18    BB   0ADB  2400          PUSHR   #^M<R3,R4>               ; Save regs
                     004F    30   0ADD  2401          BSBW    SCS$DIR_LOOKUP           ; Call lookup routine
                       18    BA   0AE0  2402          POPR    #^M<R3,R4>
                       55 8ED0    0AE2  2403 130$:    POPL    R5                       ; Restore TQE address
                              05   0AE5  2404 140$:    RSB
                              0AE6  2405
                              0AE6  2406 ;
                              0AE6  2407 ; Get here when a polled process is found;
                              0AE6  2408 ;
                              0AE6  2409 ;       R1 is address of process name in SPPB
                              0AE6  2410 ;       R2 is address of SB
                              0AE6  2411 ;       R3 is address of process info in message buffer
                              0AE6  2412 ;       Preserves R1-R5
                              0AE6  2413 ;
                              0AE6  2414 300$:
             50    18 A1    3C   0AE6  2415          MOVZWL  SPPB$W_BIT-SPPB$B_PROCNAM(R1),R0 ; Polling bit number
          1F 5A A2    50    E1   0AEA  2416          BBC     R0,SB$B_ENBMSK(R2),310$ ; Branch if not interested
                       3E    BB   0AEF  2417          PUSHR   #^M<R1,R2,R3,R4,R5>      ; Save some registers
             54     44 A2    9E   0AF1  2418          MOVAB   SB$T_NODENAME(R2),R4     ; Address of node name
             52     18 A2    9E   0AF5  2419          MOVAB   SB$B_SYSTEMID(R2),R2     ; Address of system ID
             50     14 A1    D0   0AF9  2420          MOVL    SPPB$L_CTX-SPPB$B_PROCNAM(R1),R0 ; Context longword
                              0AFD  2421 ;
                              0AFD  2422 ; Call notification routine with:
                              0AFD  2423 ;
                              0AFD  2424 ;       R0 is context longword
                              0AFD  2425 ;       R1 is address of process name
                              0AFD  2426 ;       R2 is address of system ID
                              0AFD  2427 ;       R3 is address of process information
                              0AFD  2428 ;       R4 is address of node name
```

```
                                0AFD  2429 ;
                                0AFD  2430 ;       If the notification routine returns success, disable polling
                                0AFD  2431 ;       for the process on the system on which is was found
                                0AFD  2432 ;
                      10 B1  16 0AFD  2433          JSB     @SPPB$L_RTN-SPPB$B_PROCNAM(R1)  ; Call notification routine
                         3E  BA 0B00  2434          POPR    #^M<R1,R2,R3,R4,R5>             ; Restore registers
                      09 50  E9 0B02  2435          BLBC    R0,310$                         ; Branch to leave polling enabled
                   50 18 A1  3C 0B05  2436          MOVZWL  SPPB$W_BIT-SPPB$B_PROCNAM(R1),R0        ; Polling bit number
                00 5A A2  50  E5 0B09  2437          BBCC    R0,SB$B_ENBMSK(R2),310$ ; Clear polling bit
                          05  0B0E  2438 310$:       RSB                             ; Have handled event
                              0B0F  2439
```

```
                                0B0F  2441          .SBTTL SCS$DIR_LOOKUP - look up process names on remote node
                                0B0F  2442 ;++
                                0B0F  2443 ;  SCS$DIR_LOOKUP
                                0B0F  2444 ;
                                0B0F  2445 ;  INPUTS
                                0B0F  2446 ;       R2      - address of SPNB (System Process Name Block)
                                0B0F  2447 ;       R3      - address of target SB
                                0B0F  2448 ;       IPL     - IPL$SCS
                                0B0F  2449 ;
                                0B0F  2450 ;  FUNCTION
                                0B0F  2451 ;       Connect to SCS$DIRECTORY service on remote system, look
                                0B0F  2452 ;       up process names in specified list, and notify caller
                                0B0F  2453 ;       of the ones which exist
                                0B0F  2454 ;--
                                0B0F  2455
4B 4F 4F 4C 5F 52 49 44 24 53 43 53  0B0F  2456 MY_PROC:        .ASCII /SCS$DIR_LOOKUP  /       ; Name of process on local node
                   20 20 50 55  0B1B
52 4F 54 43 45 52 49 44 24 53 43 53  0B1F  2457 REM_PROC:       .ASCII /SCS$DIRECTORY   /       ; Name of process on remote node
                   20 20 20 59  0B2B
                                0B2F  2458 SCS$DIR_LOOKUP:
                                0B2F  2459
                                0B2F  2460 ;
                                0B2F  2461 ; Connect to remote node
                                0B2F  2462 ;
                          55    0B2F  2463          MOVL    R2,R5                           ; Save context in safe register
                                0B32  2464          CONNECT MSGADR=MSG_INPUT,-              ; Try connect to directory
                                0B32  2465                  ERRADR=ERR_ROUTINE,-            ; service on remote node
                                0B32  2466                  RPRNAM=REM_PROC,-
                                0B32  2467                  LPRNAM=MY_PROC,-
                                0B32  2468                  RSYSID=SB$B_SYSTEMID(R3),-
                                0B32  2469                  INITCR=#1,-
                                0B32  2470                  AUXSTR=(R2)
                       0D 50    0B66  2471          BLBS    R0,10$                          ; If LBS, okay
                    50    55    0B69  2472          MOVL    R5,R0                           ; Get SPNB
                       00D4     0B6C  2473          BSBW    RESET_TIMER                     ; Re-enable polling on this node
                00000000'GF     0B6F  2474          JSB     G^COM$DRVDEALMEM                ; And deallocate
                                0B75  2475          RSB                                     ; Return
                                0B76  2476
                52    5C A3     0B76  2477 10$:     MOVL    CDT$L_AUXSTRUC(R3),R2           ; Restore context
                14 A2   01      0B7A  2478          MNEGB   #1,SPNB$B_INDEX(R2)             ; Init index
                                0B7E  2479
                                0B7E  2480 ;
                                0B7E  2481 ; Allocate CDRP and set it up
                                0B7E  2482 ;
                51    2C    3C  0B7E  2483          MOVZWL  #CDRP$C_LENGTH,R1               ; Specify number of bytes needed
                         53   DD  0B81  2484          PUSHL   R3
                      FCB0    30  0B83  2485          BSBW    SCS_ALONONPAGED                 ; Allocate from npagedpool
                         53 8ED0  0B86  2486          POPL    R3
                   58 50    E9  0B89  2487          BLBC    R0,25$                          ; No memory available
                      3F    BB  0B8C  2488          PUSHR   #^M<R0,R1,R2,R3,R4,R5>          ; Save regs destroyed by MOVC
62    2C    00    62    00    2C  0B8E  2489          MOVC5   #0,(R2),#0,#CDRP$C_LENGTH,(R2)  ; Zero out block
                      3F    BA  0B94  2490          POPR    #^M<R0,R1,R2,R3,R4,R5>          ; Restore regs
                55    52    D0  0B96  2491          MOVL    R2,R5                           ; Move CDRP to arg register
             08 A5    51    B0  0B99  2492          MOVW    R1, CDRP$W_CDRPSIZE(R5)         ; Set up block size
             0A A5    39    9B  0B9D  2493          MOVZBW  #DYN$C_CDRP, -
                                0BA1  2494                  CDRP$B_CD_TYPE(R5)              ; Set type
             24 A5    53    D0  0BA1  2495          MOVL    R3,CDRP$L_CDT(R5)               ; Set CDT
```

```
                        0BA5  2496
                        0BA5  2497 ;
                        0BA5  2498 ; Allocate message buffer and store next process name to look for
                        0BA5  2499 ;
                        0BA5  2500 15$:     ALLOC_MSG_BUF
      53    24 A5   D0  0BA8  2501         MOVL    CDRP$L_CDT(R5),R3               ; Get CDT
         00C0 8F    BB  0BAC  2502         PUSHR   #^M<R6,R7>                      ; Save CDT
      56    5C A3   D0  0BB0  2503         MOVL    CDT$L_AUXSTRUC(R3),R6           ; Get SPNB
         14 A6      96  0BB4  2504         INCB    SPNB$B_INDEX(R6)                ; Point to next name
      57    14 A6   9A  0BB7  2505         MOVZBL  SPNB$B_INDEX(R6),R7             ; Get index
   57    18 A647    D0  0BBB  2506         MOVL    SPNB$B_NAMLST(R6)[R7],R7        ; Get next process name
            1A      13  0BC0  2507         BEQL    20$                             ; If zero, end of list
            62      D4  0BC2  2508         CLRL    (R2)                            ; Zero out first longword of msg
      04 A2    67    7D  0BC4  2509         MOVQ    (R7),4(R2)                      ; Move to message buffer
   0C A2    08 A7    7D  0BC8  2510         MOVQ    8(R7),12(R2)                    ; (16 bytes)
         00C0 8F    BA  0BCD  2511         POPR    #^M<R6,R7>                      ; Restore CDT
         51   24    D0  0BD1  2512         MOVL    #36,R1                          ; Maximum size of msg (for HSC)
            20 A5   D4  0BD4  2513         CLRL    CDRP$L_RSPID(R5)                ; No response ID
                        0BD7  2514         SEND_CNT_MSG_BUF                        ; Send it
               C9   11  0BDA  2515         BRB     15$                             ; More names to look for?
                        0BDC  2516
                        0BDC  2517 ;
                        0BDC  2518 ; No more processes to look for - clean up
                        0BDC  2519 ;
                        0BDC  2520 20$:
         00C0 8F    BA  0BDC  2521         POPR    #^M<R6,R7>                      ; Restore CDT and stack
                        0BE0  2522         DEALLOC_MSG_BUF                         ; Deallocate message buffer
                     05 0BE3  2523         RSB
                        0BE4  2524
                        0BE4  2525 ;
                        0BE4  2526 ; Deallocate SPNB (when CDRP allocate fails)
                        0BE4  2527 ;
      50    5C A3   D0  0BE4  2528 25$:     MOVL    CDT$L_AUXSTRUC(R3),R0           ; Get SPNB
               59   10  0BE8  2529         BSBB    RESET_TIMER                     ; Re-enable polling for node
00000000'GF    16  0BEA  2530         JSB     G^COM$DRVDEALMEM                ; And deallocate
                        0BF0  2531         DISCONNECT                              ; Break connection
                     05 0BF6  2532         RSB
```

```
                                                M 5
                     0BF7  2534 ;++
                     0BF7  2535 ;  MSG_INPUT
                     0BF7  2536 ;       Process responses from SCS$DIRECTORY
                     0BF7  2537 ;
                     0BF7  2538 ;  INPUTS
                     0BF7  2539 ;       R2      - Address of response message
                     0BF7  2540 ;       R3      - CDT address
                     0BF7  2541 ;       R4      - PDT address
                     0BF7  2542 ;       IPL     - Port device fork IPL
                     0BF7  2543 ;
                     0BF7  2544 ;  FUNCTION
                     0BF7  2545 ;       Notifies lookup initiator of found process
                     0BF7  2546 ;
                     0BF7  2547 ;--
                     0BF7  2548
                     0BF7  2549 MSG_INPUT:
          31 62   E9 0BF7  2550         BLBC    (R2),35$                        ; If no success, return
      00FE 8F   BB 0BFA  2551         PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7>       ; Save registers
   55   5C A3   D0 0BFE  2552         MOVL    CDT$L_AUXSTRUC(R3),R5           ; Get address of SPNB
      54   52   D0 0C02  2553         MOVL    R2,R4                           ; Move to safe register
                     0C05  2554
                     0C05  2555 ;
                     0C05  2556 ;  Search for name in list of names we've asked about
                     0C05  2557 ;
             56   D4 0C05  2558         CLRL    R6                              ; Init index
   57   18 A546   D0 0C07  2559 10$:    MOVL    SPNB$B_NAMLST(R5)[R6],R7        ; Get next name
             2D   13 0C0C  2560         BEQL    90$                             ; If eql, end of list
04 A4   67   10   29 0C0E  2561         CMPC3   #16,(R7),4(R4)                  ; Look for match
             04   13 0C13  2562         BEQL    20$                             ; Found one
             56   D6 0C15  2563         INCL    R6                              ; Next item in list
             EE   11 0C17  2564         BRB     10$                             ; Loop
                     0C19  2565
                     0C19  2566 ;
                     0C19  2567 ;  Send on notice that remote node has process
                     0C19  2568 ;
   51   57   D0 0C19  2569 20$:    MOVL    R7,R1                           ; Get address of process name
53   14 A4   9E 0C1C  2570         MOVAB   <4+16>(R4),R3                   ; Get address of process info
52   0C A5   D0 0C20  2571         MOVL    SPNB$L_SB(R5),R2                ; Get address of SB
     10 B5   16 0C24  2572         JSB     @SPNB$C_ROUTINE(R5)             ; Pass to our notification rtn
                     0C27  2573 ;
                     0C27  2574 ;  Clean up
                     0C27  2575 ;
   00FE 8F   BA 0C27  2576 30$:    POPR    #^M<R1,R2,R3,R4,R5,R6,R7>       ; Restore registers
                     0C2B  2577 35$:
                     0C2B  2578         DEALLOC_MSG_BUF_REG                     ; Return message buf to SCS
50   5C A3   D0 0C2E  2579         MOVL    CDT$L_AUXSTRUC(R3),R0           ; Get address of SPNB
     15 A0   B7 0C32  2580         DECW    SPNB$W_REFC(R0)                 ; Decrement reference count
        01   13 0C35  2581         BEQL    40$                             ; Branch if done on connection
             05 0C37  2582         RSB                                     ; And return
                     0C38  2583
        25   10 0C38  2584 40$:    BSBB    ERR_ROUTINE                     ; Clean up on connection
                     0C3A  2585                                             ; & disconnect
             05 0C3A  2586         RSB
                     0C3B  2587
                     0C3B  2588 ;
                     0C3B  2589 ;  We get here if there is no match in the list of names we are polling
                     0C3B  2590 ;  for.  The reference count is not decremented - the connection will
```

```
                     0C3B  2591 ;  remain open without further polling as a way of flagging this error.
                     0C3B  2592 ;
00FE 8F    BA        0C3B  2593 90$:    POPR    #^M<R1,R2,R3,R4,R5,R6,R7>       ; Just exit here
                     0C3F  2594         DEALLOC_MSG_BUF_REG                     ; Get rid of buffer
           05        0C42  2595         RSB                                     ;
```

```
                                0C43  2597 ;++
                                0C43  2598 ;  RESET_TIMER
                                0C43  2599 ;       Re-enable polling on a node
                                0C43  2600 ;
                                0C43  2601 ;  INPUT
                                0C43  2602 ;       R0      - address of SPNB
                                0C43  2603 ;
                                0C43  2604 ;  OUTPUT
                                0C43  2605 ;       Polling is re-enabled on node
                                0C43  2606 ;--
                                0C43  2607
                                0C43  2608 RESET_TIMER:
                          52    DD  0C43  2609          PUSHL   R2                              ; Save R2
                    52    0C A0   D0  0C45  2610          MOVL    SPNB$L_SB(R0),R2                ; Get address of system block
                       58 A2    B6  0C49  2611          INCW    SB$W_TIMEOUT(R2)                ; Polling complete
                          05    13  0C4C  2612          BEQL    10$                             ; Branch to reset timer
                       58 A2    B4  0C4E  2613          CLRW    SB$W_TIMEOUT(R2)                ; Request polling
                          08    11  0C51  2614          BRB     20$
                                0C53  2615
          58 A2    00000000'GF   B0  0C53  2616 10$:    MOVW    G^SCS$GW_PRCPOLINT, -           ; Reset timer
                                0C5B  2617                      SB$W_TIMEOUT(R2)
                          52 8ED0   0C5B  2618 20$:    POPL    R2                              ; Restore R2
                             05     0C5E  2619          RSB                                     ; Return
                                0C5F  2620
```

```
                     ;++
                     ;  ERR_ROUTINE
                     ;       Handle broken connections
                     ;
                     ;  INPUTS
                     ;       R0      - error status
                     ;       R3      - address of CDT
                     ;       R4      - address of PDT
                     ;
                     ;  FUNCTION
                     ;       Deallocate all message buffers and CDRP'S associated with
                     ;       given CDT, and disconnect.
                     ;--
                     ERR_ROUTINE:
                             SCAN_MSGBUF_WAIT -                  ; Find CDRP'S
                                     ACTION = CLEANUP_RTN
                             DISCONNECT                          ; Break connection
                             RSB
```

```
                     0C73  2641 ;++
                     0C73  2642 ;  CLEANUP_RTN
                     0C73  2643 ;
                     0C73  2644 ;  INPUTS
                     0C73  2645 ;        R3      - address of CDT
                     0C73  2646 ;        R4      - address of PDT
                     0C73  2647 ;        R5      - address of CDRP
                     0C73  2648 ;--
                     0C73  2649
                     0C73  2650 CLEANUP_RTN:
                     0C73  2651
                     0C73  2652 ;
                     0C73  2653 ;  Remove CDRP from queue, if needed
                     0C73  2654 ;
        52    DD     0C73  2655          PUSHL   R2
      04 A5   D5     0C75  2656          TSTL    CDRP$L_FQBL(R5)         ; Is there a queue pointer?
         11   13     0C78  2657          BEQL    5$                      ; If EQL, none
55    04 B5   D1     0C7A  2658          CMPL    @CDRP$L_FQBL(R5),R5     ; Is CDRP in a queue?
         0B   12     0C7E  2659          BNEQ    5$                      ; If NEQ, no
   55    65   0F     0C80  2660          REMQUE  (R5),R5                 ; Dequeue CDRP
52    28 A5   D0     0C83  2661          MOVL    CDRP$L_RWCPTR(R5),R2    ; Get wait count pointer
         02   13     0C87  2662          BEQL    5$                      ; Branch if none
         62   B7     0C89  2663          DECW    (R2)                    ; Adjust
                     0C8B  2664
                     0C8B  2665 ;
                     0C8B  2666 ;  Deallocate associated message buffer, if any
                     0C8B  2667 ;
        52 8ED0      0C8B  2668 5$:      POPL    R2                      ; Restore reg
      1C A5   D5     0C8E  2669          TSTL    CDRP$L_MSG_BUF(R5)      ; Do we have any message buffers?
         03   13     0C91  2670          BEQL    10$                     ; If EQL, no
                     0C93  2671          DEALLOC_MSG_BUF                 ; Get rid of it
                     0C96  2672
                     0C96  2673 ;
                     0C96  2674 ;  Return CDRP to non-paged pool
                     0C96  2675 ;
    50   55    D0     0C96  2676 10$:     MOVL    R5,R0                   ; Address of CDRP in R0
00000000'GF    16     0C99  2677          JSB     G^COM$DRVDEALMEM        ; Give back to pool
 50   5C A3    D0     0C9F  2678          MOVL    CDT$L_AUXSTRUC(R3),R0   ; Get address of SPNB
         9E   10     0CA3  2679          BSBB    RESET_TIMER             ; Re-enable polling
00000000'GF    16     0CA5  2680          JSB     G^COM$DRVDEALMEM        ; Return to pool
                05     0CAB  2681          RSB                             ; Cleanup done
```

```
                     0CAC  2683          .SBTTL  SCS$POLL_MBX - Declare Polling Notification Mailbox
                     0CAC  2684
                     0CAC  2685 ;++
                     0CAC  2686 ;
                     0CAC  2687 ; FUNCTIONAL DESCRIPTION:
                     0CAC  2688 ;
                     0CAC  2689 ;       This routine is called to declare a mailbox to receive notification
                     0CAC  2690 ;       when a process name appears in an SCS directory.
                     0CAC  2691 ;
                     0CAC  2692 ;       This routine, and its companion SCS$CANCEL_MBX, are assumed to be
                     0CAC  2693 ;       used as follows:
                     0CAC  2694 ;
                     0CAC  2695 ;               $CREMBX ...                     ;create mailbox
                     0CAC  2696 ;               BLBC    R0,xxx
                     0CAC  2697 ;               JSB     SCS$POLL_MBX            ;declare polling notification
                     0CAC  2698 ;               BLBC    R0,xxx
                     0CAC  2699 ;
                     0CAC  2700 ;               ...
                     0CAC  2701 ;
                     0CAC  2702 ;               $QIO    #IO$_READVBLK,...       ;read messages from mailbox
                     0CAC  2703 ;
                     0CAC  2704 ;               ...
                     0CAC  2705 ;
                     0CAC  2706 ;               JSB     SCS$CANCEL_MBX          ;cancel polling notification
                     0CAC  2707 ;               $DASSGN ...                     ;deassign mailbox
                     0CAC  2708 ;
                     0CAC  2709 ;               Note that this sequence must be complete before the process
                     0CAC  2710 ;               exists, for whatever reason!
                     0CAC  2711 ;
                     0CAC  2712 ; CALLING SEQUENCE:
                     0CAC  2713 ;
                     0CAC  2714 ;       JSB     SCS$POLL_MBX
                     0CAC  2715 ;       IPL must be at IPL$_ASTDEL
                     0CAC  2716 ;
                     0CAC  2717 ; INPUT PARAMETERS:
                     0CAC  2718 ;
                     0CAC  2719 ;       R0 is the mailbox channel number
                     0CAC  2720 ;       R2 is the address of process name address
                     0CAC  2721 ;
                     0CAC  2722 ; OUTPUT PARAMETERS:
                     0CAC  2723 ;
                     0CAC  2724 ;       R1 is the address of the SPPB created to service this request.
                     0CAC  2725 ;
                     0CAC  2726 ; COMPLETION CODES:
                     0CAC  2727 ;
                     0CAC  2728 ;       R0 contains status
                     0CAC  2729 ;
                     0CAC  2730 ; SIDE EFFECTS:
                     0CAC  2731 ;
                     0CAC  2732 ;       R0,R1 are destroyed
                     0CAC  2733 ;
                     0CAC  2734 ;--
                     0CAC  2735
                     0CAC  2736 SCS$POLL_MBX:
           0C   BB   0CAC  2737          PUSHR   #^M<R2,R3>
00000000'GF   16   0CAE  2738          JSB     G^IOC$VERIFYCHAN        ; Verify channel number
      21 50   E9   0CB4  2739          BLBC    R0,20$                  ; Branch on invalid channel
```

```
                52   6E   7D  0CB7  2740          MOVQ    (SP),R2                 ; Restore registers
                51   61   D0  0CBA  2741          MOVL    CCB$L_UCB(R1),R1        ; UCB address
                  5C A1   B6  0CBD  2742          INCW    UCB$W_REFC(R1)          ; Nail down UCB -- security blanket
            50    DB'AF   9E  0CC0  2743          MOVAB   B^100$,R0               ; Address of notication routine
                          0CC4  2744          DSBINT  #IPL$_SCS               ; Raise IPL
                   FC95   30  0CCA  2745          BSBW    SCS$POLL_PROC           ; Declare process to poll
                 05 50    E9  0CCD  2746          BLBC    R0,10$                  ; Branch on error
                    52    D4  0CD0  2747          CLRL    R2                      ; Set polling mode for all systems
                          0CD2  2748                                              ; R1 is address of SPPB
                          0CD2  2749                                              ; R0 is odd to enable polling
                   FCD5   30  0CD2  2750          BSBW    SCS$POLL_MODE           ; Enable polling
                          0CD5  2751  10$:    ENBINT                          ; Restore IPL
                      0C  BA  0CD8  2752  20$:    POPR    #^M<R2,R3>              ; Restore registers
                          05  0CDA  2753          RSB
                          0CDB  2754
                          0CDB  2755  ;
                          0CDB  2756  ; Mailbox message format:
                          0CDB  2757  ;
                          0CDB  2758  ;        31            24 23           16 15           08 07           00
                          0CDB  2759  ;       +---------------+---------------+---------------+---------------+
                          0CDB  2760  ;    00 |                          System ID (l.o.)                     |
                          0CDB  2761  ;       +---------------+---------------+---------------+---------------+
                          0CDB  2762  ;    04 |       Unused (zero)           |          System ID (h.o.)     |
                          0CDB  2763  ;       +---------------+---------------+---------------+---------------+
                          0CDB  2764  ;    08 |                          Node Name                            |
                          0CDB  2765  ;       |                          ( 16 bytes )                         |
                          0CDB  2766  ;       |                                                               |
                          0CDB  2767  ;       |                                                               |
                          0CDB  2768  ;       +---------------+---------------+---------------+---------------+
                          0CDB  2769  ;    24 |                  Process                                      |
                          0CDB  2770  ;       |                      Name                                     |
                          0CDB  2771  ;       |                          ( 16                                 |
                          0CDB  2772  ;       |                            Bytes )                            |
                          0CDB  2773  ;       +---------------+---------------+---------------+---------------+
                          0CDB  2774  ;    40 |                  Directory                                    |
                          0CDB  2775  ;       |                      Information                              |
                          0CDB  2776  ;       |                          ( 16                                 |
                          0CDB  2777  ;       |                            Bytes )                            |
                          0CDB  2778  ;       +---------------+---------------+---------------+---------------+
                          0CDB  2779  ;    56
                          0CDB  2780  ;
                          0CDB  2781  ;
                          0CDB  2782  ; Notification routine called when process name is found
                          0CDB  2783  ;
                          0CDB  2784  ; Inputs:
                          0CDB  2785  ;       R0 is UCB address
                          0CDB  2786  ;       R1 is address of process name
                          0CDB  2787  ;       R2 is address of system ID
                          0CDB  2788  ;       R3 is address of process information
                          0CDB  2789  ;       R4 is address of node name
                          0CDB  2790  ;
                          0CDB  2791  ; Outputs:
                          0CDB  2792  ;       R1-R5 destroyed
                          0CDB  2793  ;       R0 is status
                          0CDB  2794  ;
            55    50    D0  0CDB  2795  100$:   MOVL    R0,R5                   ; Notification still wanted?
                  39    13  0CDE  2796          BEQL    110$                    ; Branch if no
```

```
                        0CE0  2797          ASSUME  PRCPOL$C_SIZ,EQ,56      ; Ensure size is longword aligned
        5E   38   C2    0CE0  2798          SUBL2   #PRCPOL$C_SIZ,SP         ; Make room on stack
        50   5E   D0    0CE3  2799          MOVL    SP,R0                    ; Get address of allocated space
                        0CE6  2800          ASSUME  $DIR$S_PROCINF,EQ,16     ; Use R0 to point to message on stack
30 A0   08 A3   7D      0CE6  2801          MOVQ    8(R3),PRCPOL$B_DIRINF+8(R0) ; Process information --
   28 A0   63   7D      0CEB  2802          MOVQ    (R3),PRCPOL$B_DIRINF(R0)    ;   16 bytes
                        0CEF  2803          ASSUME  $DIR$S_PROCNAM,EQ,16
20 A0   08 A1   7D      0CEF  2804          MOVQ    8(R1),PRCPOL$B_PRCNAM+8(R0) ; Process name --
   18 A0   61   7D      0CF4  2805          MOVQ    (R1),PRCPOL$B_PRCNAM(R0)    ;   16 bytes
                        0CF8  2806          ASSUME  SB$T_NODENAME+16  EQ  SB$L_DDB
10 A0   08 A4   7D      0CF8  2807          MOVQ    8(R4),PRCPOL$T_NODNAM+8(R0) ; Node name --
   08 A0   64   7D      0CFD  2808          MOVQ    (R4),PRCPOL$T_NODNAM(R0)    ;   16 bytes
                        0D01  2809          ASSUME  SB$S_SYSTEMID,EQ,6
04 A0   04 A2   3C      0D01  2810          MOVZWL  4(R2),PRCPOL$W_SYSIDH(R0)   ; System ID --
      60   62   D0      0D06  2811          MOVL    (R2),PRCPOL$L_SYSIDL(R0)    ;   6 bytes
      54   50   D0      0D09  2812          MOVL    R0,R4                    ; Address of message
      53   38   9A      0D0C  2813          MOVZBL  #PRCPOL$C_SIZ,R3         ; Length of message
00000000'GF   16   0D0F  2814          JSB     G^EXE$WRTMAILBOX         ; Put message into mailbox
      5E   38   C0      0D15  2815          ADDL2   #PRCPOL$C_SIZ,SP         ; Clean stack
                05      0D18  2816          RSB                              ; On success, disable polling
      50   01   D0      0D19  2817 110$:    MOVL    S^#SS$_NORMAL,R0         ; Disable polling
                05      0D1C  2818          RSB
```

```
                                                H 6
                                .SBTTL  SCS$CANCEL_MBX - Cancel Polling Notification Mailbox
                                ;++
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       This routine is called to cancel mailbox delivery of notification
                                ;       when a process name appears in an SCS directory.
                                ;
                                ;       The user of a notification mailbox must call this routine before
                                ;       exiting and before deassigning its channel to the mailbox.  If this
                                ;       constraint is not satisfied, the mailbox may remain forever,
                                ;       undeleted.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       JSB     SCS$CANCEL_MBX
                                ;       IPL must be in range 0 to IPL$_SCS, IPL$_TIMER
                                ;
                                ; INPUT PARAMETERS:
                                ;
                                ;       R1 is the address of the SPPB
                                ;
                                ; OUTPUT PARAMETERS:
                                ;
                                ;       NONE
                                ;
                                ; COMPLETION CODES:
                                ;
                                ;       R0 contains status
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       R0,R1,R2 are destroyed
                                ;
                                ;--

                                SCS$CANCEL_MBX:
                                        DSBINT  #IPL$_SCS               ; Raise IPL
50  20 A1   D0                         MOVL    SPPB$L_CTX(R1),R0       ; Mailbox UCB address
    20 A1   D4                         CLRL    SPPB$L_CTX(R1)          ; Forget mailbox UCB address
    5C A0   B7                         DECW    UCB$W_REFC(R0)          ; Release grip on UCB -- security blanket
       52   D4                         CLRL    R2                      ; For all systems
       50   D4                         CLRL    R0                      ;    disable polling
     FC76   30                         BSBW    SCS$POLL_MODE           ; Disable polling
                                        ENBINT                          ; Restore IPL
            05                          RSB
```

```
                                                                I  6
                     0D38  2867          .SBTTL  SCS$SHUTDOWN - Shutdown all SCS virtual circuits
                     0D38  2868 ;++
                     0D38  2869 ;
                     0D38  2870 ; FUNCTIONAL DESCRIPTION:
                     0D38  2871 ;
                     0D38  2872 ;       This routine runs down the singly linked list of SCS speaking
                     0D38  2873 ;       PDT's, calling the associated port driver for each one it finds.
                     0D38  2874 ;       The port driver will close all virtual circuits it has outstanding.
                     0D38  2875 ;       Control will return upon the port being offline, encountering any
                     0D38  2876 ;       errors, or successful termination.
                     0D38  2877 ;
                     0D38  2878 ; CALLING SEQUENCE:
                     0D38  2879 ;
                     0D38  2880 ;       JSB     SCS$SHUTDOWN
                     0D38  2881 ;       IPL = 31.
                     0D38  2882 ;
                     0D38  2883 ; INPUT PARAMETERS:
                     0D38  2884 ;
                     0D38  2885 ;       NONE
                     0D38  2886 ;
                     0D38  2887 ; OUTPUT PARAMETERS:
                     0D38  2888 ;
                     0D38  2889 ;       NONE
                     0D38  2890 ;
                     0D38  2891 ; COMPLETION CODES:
                     0D38  2892 ;
                     0D38  2893 ;       NONE
                     0D38  2894 ;
                     0D38  2895 ; SIDE EFFECTS:
                     0D38  2896 ;
                     0D38  2897 ;       R0-R4 are destroyed
                     0D38  2898 ;
                     0D38  2899 ;--
                     0D38  2900
                     0D38  2901 SCS$SHUTDOWN:
54   00000000'9F     DE   0D38  2902          MOVAL   @#SCS$GL_PDT,R4         ; Pick up list head of PDT's
     54   64         D0   0D3F  2903 10$:     MOVL    (R4),R4                 ; Get next entry
          06         13   0D42  2904          BEQL    20$                     ; There is none
     0080 D4         16   0D44  2905          JSB     @PDT$L_STOP_VCS(R4)     ; Call the port driver to close VC's
          F5         11   0D48  2906          BRB     10$                     ; Loop for next one
                     0D4A  2907
                     05   0D4A  2908 20$:     RSB                             ; Finish, leave
```

```
00000D4B  0D4B  2910          LC=.
00000D50  0D4B  2911          .=<LC+15>&-16                   ; Align on 16 byte boundary
          0D50  2912 SCS_END:
          0D50  2913          .ENDC                           ; End of conditional code
          0D50  2914          .END
```

```
BLANK                           00000574 R      02
BUG$_INVRSPID                   ********   X    02
CCB$C_UCB                     = 00000000
CDL$C_LENGTH                  = 00000010
CDL$L_FREECDT                 = FFFFFFF4
CDL$W_MAXCONIDX               = FFFFFFF0
CDRP$B_CD_TYPE                = 0000000A
CDRP$C_LENGTH                 = 0000002C
CDRP$L_CDT                    = 00000024
CDRP$L_FPC                    = 0000000C
CDRP$L_FQBL                   = 00000004
CDRP$L_FQFL                   = 0000C000
CDRP$L_FR3                    = 00000010
CDRP$L_MSG_BUF                = 0000001C
CDRP$L_RSPID                  = 00000020
CDRP$L_RWCPTR                 = 00000028
CDRP$L_SAVD_RTN               = 00000018
CDRP$L_UCB                    = FFFFFFBC
CDRP$W_CDRPSIZE               = 00000008
CDT$B_PRIORITY                = 0000004E
CDT$B_RSTATION                = 00000020
CDT$B_TYPE                    = 0000000A
CDT$C_CLOSED                  = 00000000
CDT$C_LENGTH                  = 000000A0
CDT$C_LISTEN                  = 00000001
CDT$L_AUXSTRUC                = 0000005C
CDT$L_BADRSP                  = 00000060
CDT$L_CDTLST                  = 0000006C
CDT$L_CONDAT                  = 00000058
CDT$L_CRWAITQBL               = 0000003C
CDT$L_CRWAITQFL               = 00000038
CDT$L_DGINPUT                 = 00000004
CDT$L_ERRADDR                 = 0000000C
CDT$L_LCONID                  = 00000018
CDT$L_LINK                    = 00000000
CDT$L_LPROCNAM                = 00000054
CDT$L_MSGINPUT                = 00000000
CDT$L_PB                      = 0000001C
CDT$L_PDT                     = 00000010
CDT$L_RPROCNAM                = 00000050
CDT$L_WAITQBL                 = 00000034
CDT$L_WAITQFL                 = 00000030
CDT$W_BLKSTATE                = 0000002A
CDT$W_DGREC                   = 0000004C
CDT$W_INITLREC                = 00000048
CDT$W_MINSEND                 = 0000004A
CDT$W_SIZE                    = 00000008
CDT$W_STATE                   = 00000028
CLEANUP_RTN                     00000C73 R      02
CODE_HEADER                     000002B0 R      02
COM$DRVDEALMEM                  ********   X    02
DDT$L_START                   = 00000000
DIRERR                          000006B5 R      02
DIRINFO                         00000127 R      02
DIRNAME                         00000117 R      02
DIR_CLEANUP                     000006C9 R      02
DYN$C_CDRP                    = 00000039
DYN$C_LC_SCS                  = 00000004
DYN$C_LOADCODE                = 00000062
DYN$C_SCS                     = 00000060
DYN$C_SCS_CDL                 = 00000001
DYN$C_SCS_CDT                 = 00000002
DYN$C_SCS_DIR                 = 00000003
DYN$C_SCS_RDT                 = 00000006
DYN$C_SCS_SPNB                = 00000009
DYN$C_SCS_SPPB                = 00000008
DYN$C_TQE                     = 0000000F
ERR_ROUTINE                     00000C5F R      02
EXE$ALONONPAGED                 ********   X    02
EXE$DEALLOCATE                  ********   X    02
EXE$GL_NONPAGED                 ********   X    02
EXE$GQ_SYSTIME                  ********   X    02
EXE$GQ_TODCBASE                 ********   X    02
EXE$INSTIMQ                     ********   X    02
EXE$WRTMAILBOX                  ********   X    02
INIT_BEGIN                      000000AF R      02
INIT_CDL                        00000159 R      02
INIT_RDT                        000001FF R      02
IOC$VERIFYCHAN                  ********   X    02
IPL$_SCS                      = 00000008
IPL$_SYNCH                    = 00000008
IPL$_TIMER                    = 00000008
LC                            = 00000D4B R      02
LISTENERR                       000006F6 R      02
MSG_INPUT                       00000BF7 R      02
MY_PROC                         00000B0F R      02
NO_CDT                          00000370 R      02
PB$B_CBL_STS                  = 00000028
PB$B_P0_STS                   = 00000029
PB$B_P1_STS                   = 0000002A
PB$B_RSTATE                   = 00000021
PB$B_RSTATION                 = 0000000C
PB$B_RST_PORT                 = 00000020
PB$C_OPEN                     = 00000003
PB$L_CDTLST                   = 00000034
PB$L_FLINK                    = 00000000
PB$L_PDT                      = 0000002C
PB$L_RPORT_FCN                = 0000001C
PB$L_RPORT_REV                = 00000018
PB$L_RPORT_TYP                = 00000014
PB$T_LPORT_NAME               = 00000024
PB$W_RETRY                    = 00000022
PB$W_STATE                    = 00000012
PBO$B_CBL_STS                 = 0000001C
PBO$B_NXT_RSTAT               = 00000020
PBO$B_P0_STS                  = 0000001D
PBO$B_P1_STS                  = 0000001E
PBO$B_RSTATE                  = 00000015
PBO$B_RSTATION                = 00000000
PBO$B_RST_PORT                = 00000014
PBO$B_SYSTEMID                = 0000002C
PBO$L_RPORT_FCN               = 00000010
PBO$L_RPORT_REV               = 0000000C
PBO$L_RPORT_TYP               = 00000008
```

```
PBO$T_LPORT_NAME        =  00000018
PBO$T_NXT_LPORT         =  00000028
PBO$W_RETRY             =  00000016
PBO$W_STATE             =  00000006
PDT$L_ACCEPT            =  0000000C
PDT$L_ALLOCMSG          =  00000014
PDT$L_CONNECT           =  00000018
PDT$L_DCONNECT          =  00000028
PDT$L_DEALLOMSG         =  00000020
PDT$L_DEALRGMSG         =  00000024
PDT$L_RCLMSGBUF         =  00000048
PDT$L_REJECT            =  0000004C
PDT$L_SNDCNTMSG         =  00000060
PDT$L_STOP_VCS          =  00000080
PDT$L_WAITQFL           =  000000AC
PDT_JMP                    00000555 R      02
POLLER_INIT                0000024E R      02
PR$_IPL                    ********   X    02
PRCPOL$B_DIRINF         =  00000028
PRCPOL$B_PRCNAM         =  00000018
PRCPOL$C_SIZ            =  00000038
PRCPOL$L_SYSIDL         =  00000000
PRCPOL$T_NODNAM         =  00000008
PRCPOL$W_SYSIDH         =  00000004
PRMSW                   =  00000001
PRT$C_ER                =  00000007
PRT$C_EW                =  00000005
RD$C_LENGTH             =  00000008
RD$L_CDRP               =  00000000
RD$L_LINK               =  00000000
RD$V_BUSY               =  00000000
RD$W_SEQNUM             =  00000006
RD$W_STATE              =  00000004
RDT$C_LENGTH            =  00000018
RDT$L_FREERD            =  FFFFFFF4
RDT$L_MAXRDIDX          =  FFFFFFF8
RDT$L_WAITBL            =  FFFFFFEC
RDT$L_WAITFL            =  FFFFFFE8
REM_PROC                   00000B1F R      02
RESET_TIMER                00000C43 R      02
SB$B_ENBMSK             =  0000005A
SB$B_HWVERS             =  00000038
SB$B_SYSTEMID           =  00000018
SB$L_DDB                =  00000054
SB$L_FLINK              =  00000000
SB$L_PBCONNX            =  00000014
SB$L_PBFL               =  0000000C
SB$Q_SWINCARN           =  0000002C
SB$S_ENBMSK             =  00000002
SB$S_SYSTEMID           =  00000006
SB$T_HWTYPE             =  00000034
SB$T_NODENAME           =  00000044
SB$T_SWTYPE             =  00000024
SB$T_SWVERS             =  00000028
SB$W_MAXDG              =  00000020
SB$W_MAXMSG             =  00000022
SB$W_TIMEOUT            =  00000058
SBO$B_HWVERS            =  00000020
SBO$B_NXT_SYSID         =  00000048
SBO$B_RSTATION1         =  0000003C
SBO$B_SYSTEMID          =  00000000
SBO$Q_SWINCARN          =  00000014
SBO$T_HWTYPE            =  0000001C
SBO$T_LPORT1            =  00000044
SBO$T_NODENAME          =  0000002C
SBO$T_SWTYPE            =  0000000C
SBO$T_SWVERS            =  00000010
SBO$W_MAXDG             =  00000008
SBO$W_MAXMSG            =  0000000A
SB_COPY_LEN             =  0000003C
SCS$ACCEPT                 000002D6 RG     02
SCS$ALLOC_CDT              000002F3 R      02
SCS$ALLOC_RSPID            00000376 R      02
SCS$CANCEL_MBX             00000D1D R      02
SCS$CONFIG_PTH             000003B7 R      02
SCS$CONFIG_SYS             0000043B R      02
SCS$CONNECT                000004B5 R      02
SCS$DEALL_CDT              00000584 R      02
SCS$DEALL_CDT1             000005A9 R      02
SCS$DEALL_RSPID            000005C5 R      02
SCS$DIRECTORY              00000626 R      02
SCS$DIR_LOOKUP             00000B2F R      02
SCS$DISCONNECT             000006FD R      02
SCS$ENTER                  00000718 R      02
SCS$FIND_RDTE              0000075F R      02
SCS$GA_DFLTMSK             ********   X    02
SCS$GA_EXISTS              ********   X    02
SCS$GA_LOCALSB             ********   X    02
SCS$GL_CDL                 ********   X    02
SCS$GL_PDT                 ********   X    02
SCS$GL_RDT                 ********   X    02
SCS$GQ_CONFIG              ********   X    02
SCS$GQ_DIRECT              ********   X    02
SCS$GQ_POLL                ********   X    02
SCS$GW_CDTCNT              ********   X    02
SCS$GW_CDTMAX              000002D4 RG     02
SCS$GW_NEXTBIT             ********    X   02
SCS$GW_PRCPOLINT           ********    X   02
SCS$GW_RDTCNT              ********    X   02
SCS$K_CDTEXT            =  00000064  G
SCS$LISTEN                 0000078A RG     02
SCS$LKP_MSGWAIT            00000900 R      02
SCS$LKP_RDTCDRP            00000888 R      02
SCS$LKP_RDTWAIT            000008CB R      02
SCS$LOCLOOKUP              000007B3 R      02
SCS$NEW_SB                 00000951 R      02
SCS$POLL_MBX               00000CAC R      02
SCS$POLL_MODE              000009AA R      02
SCS$POLL_PROC              00000962 R      02
SCS$RECYL_RSPID            000005C3 R      02
SCS$REMOVE                 00000806 R      02
SCS$RESUMEWAITR            0000084B R      02
SCS$SHUTDOWN               00000D38 R      02
SCS$UNSTALLUCB             00000868 R      02
```

```
SCS_ALONONPAGED                 00000836 R      02
SCS_DEALNONPAGD                 00000845 R      02
SCS_END                         00000D50 R      02
SCS_START                       00000000 R      02
SDIR$B_PROCINF                  =  0000001C
SDIR$B_PROCNAM                  =  0000000C
SDIR$C_LENGTH                   =  00000030
SDIR$L_CONID                    =  0000002C
SDIR$S_PROCINF                  =  00000010
SDIR$S_PROCNAM                  =  00000010
SDIR$W_SIZE                     =  00000008
SEQ_NO                          00000137 R      02
SPNB$B_INDEX                    =  00000014
SPNB$B_NAMLST                   =  00000018
SPNB$B_SUBTYP                   =  0000000B
SPNB$B_TYPE                     =  0000000A
SPNB$C_HDRSIZ                   =  00000018
SPNB$L_ROUTINE                  =  00000010
SPNB$L_SB                       =  0000000C
SPNB$W_REFC                     =  00000015
SPNB$W_SIZE                     =  00000008
SPPB$B_PROCNAM                  =  0000000C
SPPB$B_SUBTYP                   =  0000000B
SPPB$B_TYPE                     =  0000000A
SPPB$K_LENGTH                   =  00000028
SPPB$L_CTX                      =  00000020
SPPB$L_FLINK                    =  00000000
SPPB$L_RTN                      =  0000001C
SPPB$S_PROCNAM                  =  00000010
SPPB$W_BIT                      =  00000024
SPPB$W_SIZE                     =  00000008
SS$_ALRDYCLOSED                 =  000006A9
SS$_INSFCDT                     =  000021AC
SS$_INSFMEM                     =  00000124
SS$_NORMAL                      =  00000001
SS$_NOSUCHNODE                  =  0000028C
SS$_NOSUCHOBJ                   =  000020A4
SS$_UNREACHABLE                 =  00002094
START_POLL                      00000A05 R      02
THREAD_HAS_RWAITCNT             00000859 R      02
TQE$B_RQTYPE                    =  0000000B
TQE$B_TYPE                      =  0000000A
TQE$C_SSREPT                    =  00000005
TQE$K_LENGTH                    =  00000030
TQE$L_FPC                       =  0000000C
TQE$L_FR3                       =  00000010
TQE$L_FR4                       =  00000014
TQE$Q_DELTA                     =  00000020
TQE$W_SIZE                      =  00000008
UCB$L_DDT                       =  00000088
UCB$L_IOQFL                     =  0000004C
UCB$M_CANCEL                    =  00000008
UCB$M_TIMOUT                    =  00000040
UCB$W_REFC                      =  0000005C
UCB$W_RWAITCNT                  =  00000056
UCB$W_STS                       =  00000064
```

### Psect synopsis

| PSECT name | Allocation | PSECT No. | Attributes |
|---|---|---|---|
| . ABS . | 00000000 ( 0.) | 00 ( 0.) | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| $ABS$ | 00000000 ( 0.) | 01 ( 1.) | NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE |
| $$$000 | 00000D50 ( 3408.) | 02 ( 2.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG |

### Performance indicators

| Phase | Page faults | CPU Time | Elapsed Time |
|---|---|---|---|
| Initialization | 37 | 00:00:00.03 | 00:00:00.62 |
| Command processing | 144 | 00:00:00.52 | 00:00:02.29 |
| Pass 1 | 536 | 00:00:15.60 | 00:00:52.75 |
| Symbol table sort | 0 | 00:00:01.78 | 00:00:05.97 |
| Pass 2 | 408 | 00:00:05.01 | 00:00:20.37 |
| Symbol table output | 1 | 00:00:00.16 | 00:00:00.16 |
| Psect synopsis output | 0 | 00:00:00.01 | 00:00:00.01 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 1128 | 00:00:23.14 | 00:01:22.19 |

The working set limit was 2400 pages.
135472 bytes (265 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1608 non-local and 139 local symbols.
2915 source lines were read in Pass 1, producing 23 object records in Pass 2.
53 pages of virtual memory were used to define 49 macros.

### Macro library statistics

| Macro library name | Macros defined |
|---|---|
| _$255$DUA28:[SYS.OBJ]LIB.MLB;1 | 39 |
| _$255$DUA28:[SYSLIB]STARLET.MLB;2 | 7 |
| TOTALS (all libraries) | 46 |

1946 GETS were required to define 46 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:SCSLOA/OBJ=OBJ$:SCSLOA MASD$:[SYS.SRC]PRMSW/UPDATE=(MASD$:[SYS.ENH]PRMSW)+MASD$:[SYS.SRC]SCSVEC/UPDATE=(MASD$:[SYS.EN
```

UTILKEY.LIS

TECOLBR.LIS

TECOCMD.LIS

SCSLOA.LIS

TECO

TECO.MAP

TECONAT.LIS