```
UUU          UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSS  YYY          YYY
UUU          UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSS  YYY          YYY
UUU          UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSS  YYY          YYY
UUU          UUU  EEE                    TTT        PPP          PPP  SSS              YYY          YYY
UUU          UUU  EEE                    TTT        PPP          PPP  SSS              YYY          YYY
UUU          UUU  EEE                    TTT        PPP          PPP  SSS              YYY          YYY
UUU          UUU  EEE                    TTT        PPP          PPP  SSS                 YYY    YYY
UUU          UUU  EEE                    TTT        PPP          PPP  SSS                 YYY    YYY
UUU          UUU  EEE                    TTT        PPP          PPP  SSS                 YYY    YYY
UUU          UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP       SSSSSSSSS              YYY
UUU          UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP       SSSSSSSSS              YYY
UUU          UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP       SSSSSSSSS              YYY
UUU          UUU  EEE                    TTT        PPP                        SSS              YYY
UUU          UUU  EEE                    TTT        PPP                        SSS              YYY
UUU          UUU  EEE                    TTT        PPP                        SSS              YYY
UUU          UUU  EEE                    TTT        PPP                        SSS              YYY
UUU          UUU  EEE                    TTT        PPP                        SSS              YYY
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP                SSSSSSSSSSS              YYY
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP                SSSSSSSSSSS              YYY
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP                SSSSSSSSSSS              YYY
```

SA
VO

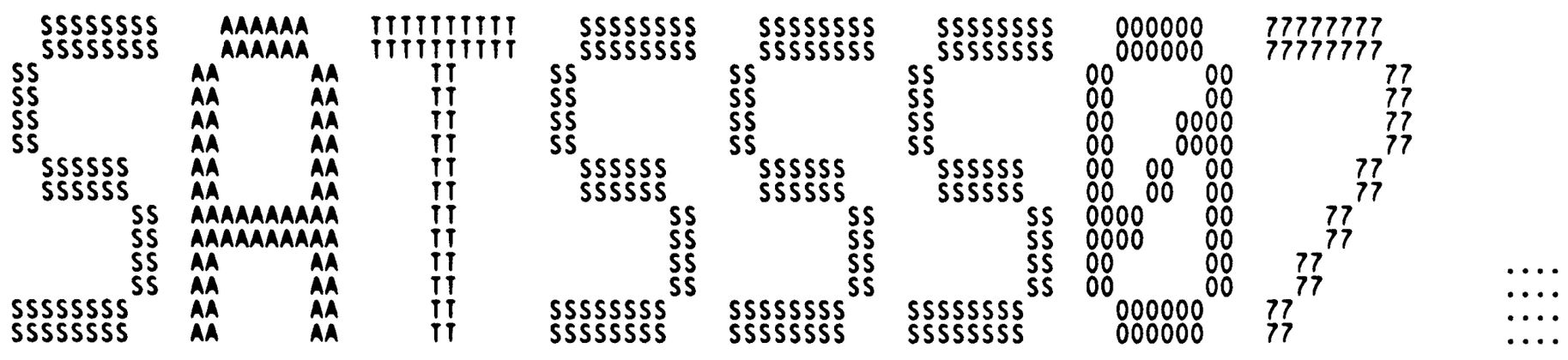
SSSSSSSS AAAAAA TTTTTTTTTT SSSSSSSS SSSSSSSS SSSSSSSS 000000 77777777
SSSSSSSS AAAAAA TTTTTTTTTT SSSSSSSS SSSSSSSS SSSSSSSS 000000 77777777
SS AA AA TT SS SS SS 00 00 77
SS AA AA TT SS SS SS 00 00 77
SS AA AA TT SS SS SS 00 0000 77
SS AA AA TT SS SS SS 00 0000 77
SSSSSS AA AA TT SSSSSS SSSSSS SSSSSS 00 00 00 77
SSSSSS AA AA TT SSSSSS SSSSSS SSSSSS 00 00 00 77
SS AAAAAAAAAA TT SS SS SS 0000 00 77
SS AAAAAAAAAA TT SS SS SS 0000 00 77
SS AA AA TT SS SS SS 00 00 77
SS AA AA TT SS SS SS 00 00 77
SSSSSSSS AA AA TT SSSSSSSS SSSSSSSS SSSSSSSS 000000 77
SSSSSSSS AA AA TT SSSSSSSS SSSSSSSS SSSSSSSS 000000 77
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

Table of contents

```
0000     1          .TITLE  SATSSS07 - SATS SYSTEM SERVICE TESTS  (SUCC S.C.)
0000     2          .IDENT  'V04-000'
0000     3          .DEFAULT DISPLACEMENT,WORD
0000     4          .ENABLE SUPPRESSION
0000     5  ;
0000     6  ;*****************************************************************************
0000     7  ;*                                                                           *
0000     8  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     9  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    10  ;*  ALL RIGHTS RESERVED.                                                     *
0000    11  ;*                                                                           *
0000    12  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    13  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    14  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    15  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    16  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    17  ;*  TRANSFERRED.                                                             *
0000    18  ;*                                                                           *
0000    19  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    20  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    21  ;*  CORPORATION.                                                             *
0000    22  ;*                                                                           *
0000    23  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    24  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    25  ;*                                                                           *
0000    26  ;*                                                                           *
0000    27  ;*****************************************************************************
0000    28  ;
0000    29
0000    30  ;++
0000    31  ; FACILITY:     SATS SYSTEM SERVICE TESTS
0000    32  ;
0000    33  ; ABSTRACT:     The SATSSS07 module tests the execution of the following
0000    34  ;               VMS system services:
0000    35  ;
0000    36  ;               $CREMBX
0000    37  ;               $DELMBX
0000    38  ;
0000    39  ; ENVIRONMENT:  User mode image.
0000    40  ;               Needs TMPMBX and PRMMBX privilege.
0000    41  ;
0000    42  ; AUTHOR: Paula Tirak,                  CREATION DATE: SEPTEMBER, 1979
0000    43  ;
0000    44  ; MODIFIED BY:
0000    45  ;
0000    46  ;       V03-005 LDJ0002         Larry D. Jones,         08-May-1984
0000    47  ;               Fixed bug in V03-004.
0000    48  ;
0000    49  ;       V03-004 LDJ0001         Larry D. Jones,         23-Aug-1983
0000    50  ;               Converted to new logical name length limit of 255.
0000    51  ;
0000    52  ;       V03-003 KDM0002         Kathleen D. Morse       28-Jun-1982
0000    53  ;               Added $IODEF, $PRDEF, $PSLDEF, and $SSDEF.
0000    54  ;
0000    55  ;--
0000    56  ;--
```

```
          0000          .SBTTL  DECLARATIONS
          0000  ;
          0000  ; MACRO LIBRARY CALLS
          0000  ;
          0000          .LIBRARY /SYS$LIBRARY:STARLET.MLB/
          0000          $DIBDEF                         ; device information block definitions
          0000          $DSCDEF                         ; string descriptor definitions
          0000          $IODEF                          ; i/o function code definitions
          0000          $PRDEF                          ; processor register definitions
          0000          $PRVDEF                         ; privilege name definitions
          0000          $PSLDEF                         ; program status longword definitions
          0000          $SFDEF                          ; stack frame definitions
          0000          $SHR_MESSAGES UETP,116,<<TEXT,INFO>> ; UETP$_TEXT definition
          0000          $SSDEF                          ; system status code definitions
          0000          $STSDEF                         ; STS definitions
          0000          $UETPDEF                        ; UETP message definitions
          0000  ;
          0000  ; Equated symbols
          0000  ;
00000000  0000  WARNING         = 0                     ; warning severity value for msgs
00000001  0000  SUCCESS         = 1                     ; success      ''     ''   ''  ''
00000002  0000  ERROR           = 2                     ; error        ''     ''   ''  ''
00000003  0000  INFO            = 3                     ; information  ''     ''   ''  ''
00000004  0000  SEVERE          = 4                     ; fatal        ''     ''   ''  ''
00000400  0000  MBBUF           = 1024                  ; MBX max. size for messages
00002000  0000  HIMSG           = 8192
00000001  0000  LOMSG           = 1
00000100  0000  BUF_SIZ_S       = 256                   ; size for _S form
00000400  0000  BUF_SIZ_G       =1024                   ; size for _G form
00000001  0000  ENABLE          = 1                     ; for SETPRV calls
00000000  0000  DISABLE         = 0                     ;  ''    ''
00000100  0000  TEXT_BUFFER     = 256                   ; misc. message manipulations
          0000  ;
          0000  ;
          0000  ; MACROS
          0000  ;
```

```
                                   00000000      95           .PSECT  RODATA,RD,NOWRT,NOEXE,LONG
                                       0000      96 ;
                                       0000      97 ARGLST:
                           00000001    0000      98           .LONG   1
                           00000BAC'   0004      99           .ADDRESS SUPER_MODE
                                       0008     100 CREMBX:
             58 42 4D 45 52 43 00'     0008     101           .ASCIC  /CREMBX/
                                06     0008
                                       000F     102 CS1:                                ; failure messages
21 20 74 73 65 54 00000017'010E0000'   000F     103           .ASCID  \Test !AC service name !AC step !UL failed.\
6E 20 65 63 69 76 72 65 73 20 43 41    001D
70 65 74 73 20 43 41 21 20 65 6D 61    0029
2E 64 65 6C 69 61 66 20 4C 55 21 20    0035
                                       0041     104 CS2:
74 63 65 70 78 45 00000049'010E0000'   0041     105           .ASCID  \Expected !AS = !XL received !AS = !XL\
4C 58 21 20 3D 20 53 41 21 20 64 65    004F
41 21 20 64 65 76 69 65 63 65 72 20    005B
             4C 58 21 20 3D 20 53      0067
                                       006E     106 CS3:
74 63 65 70 78 45 00000076'010E0000'   006E     107           .ASCID  \Expected !AS!UB = !XL received !AS!UB = !XL\
20 3D 20 42 55 21 53 41 21 20 64 65    007C
64 65 76 69 65 63 65 72 20 4C 58 21    0088
58 21 20 3D 20 42 55 21 53 41 21 20    0094
                                4C     00A0
                                       00A1     108 CS4:
72 69 75 71 65 52 000000A9'010E0000'   00A1     109           .ASCID  \Required channel not received.\
6E 20 6C 65 6E 6E 61 68 63 20 64 65    00AF
2E 64 65 76 69 65 63 65 72 20 74 6F    00BB
                                       00C7     110 CS5:
77 20 65 64 6F 4D 000000CF'010E0000'   00C7     111           .ASCID  \Mode was !AS.\
             2E 53 41 21 20 73 61      00D5
                                       00DC     112 NSSSF:
75 73 2D 6E 6F 4E 000000E4'010E0000'   00DC     113           .ASCID  \Non-subject system service failure of : !/!_      !AS\
6D 65 74 73 79 73 20 74 63 65 6A 62    00EA
69 61 66 20 65 63 69 76 72 65 73 20    00F6
2F 21 20 3A 20 66 6F 20 65 72 75 6C    0102
   53 41 21 20 20 20 20 20 20 5F 21    010E
                                       0119     114 DCLCMH:
             48 4D 43 4C 43 44 00'     0119     115           .ASCIC  /DCLCMH/
                                06     0119
                                       0120     116 DELMBX:
             58 42 4D 4C 45 44 00'     0120     117           .ASCIC  /DELMBX/
                                06     0120
                                       0127     118 EM:
54 55 43 45 58 45 0000012F'010E0000'   0127     119           .ASCID  /EXECUTIVE/
                          45 56 49     0135
                                       0138     120 EXP:
73 75 74 61 74 73 00000140'010E0000'   0138     121           .ASCID  \status\
                                       0146     122 KM:
4C 45 4E 52 45 4B 0000014E'010E0000'   0146     123           .ASCID  /KERNEL/
                                       0154     124 LOGNAMG:
37 30 53 5F 42 4D 0000015C'010E0000'   0154     125           .ASCID  /MB_S07_G/
                             47 5F     0162
                                       0164     126 LOGNAMG1:
37 30 53 5F 42 4D 0000016C'010E0000'   0164     127           .ASCID  /MB_S07_PG/
                          47 50 5F     0172
                                       0175     128 LOGNAMS:
```

```
37 30 53 5F 42 4D 0000017D'010E0000' 0175   129          .ASCID  /MB_S07_S/
                        53 5F  0183
                               0185   130 LOGNAMS1:
37 30 53 5F 42 4D 0000018D'010E0000' 0185   131          .ASCID  /MB_S07_PS/
                     53 50 5F  0193
                               0196   132 SYS_DEV:                                        ; system disk descriptor
59 53 24 53 59 53 0000019E'010E0000' 0196   133          .ASCID  /SYS$SYSDEVICE/
                  45 43 49 56 45 44 53  01A4
                               01AB   134 TMP_PRV_MASK:                                   ; create temp mailbox priv
                         00008000  01AB   135          .LONG   <1@PRV$V_TMPMBX>
                         00000000  01AF   136          .LONG   0
                               01B3   137 PRM_PRV_MASK:                                   ; create perm mailbox priv
                         00000800  01B3   138          .LONG   <1@PRV$V_PRMMBX>
                         00000000  01B7   139          .LONG   0
                               01BB   140 TMP_PRM_PRV_MASK:                               ; create perm & temp priv
                         00008800  01BB   141          .LONG   <1@PRV$V_PRMMBX>!<1@PRV$V_TMPMBX>
                         00000000  01BF   142          .LONG   0
                               01C3   143 GRP_PRV_MASK:                                   ; GRPNAM privilege mask
                         00000008  01C3   144          .LONG   <1@PRV$V_GRPNAM>
                         00000000  01C7   145          .LONG   0
                               01CB   146 CHANNEL_ZERO:                                   ; MBX channel = 0
                             0000  01CB   147          .WORD   0
                               01CD   148 LEN_00_DESCR:
                             0000  01CD   149          .WORD   0                         ; zero length descriptor
                               0E  01CF   150          .BYTE   DSC$K_DTYPE_T
                               01  01D0   151          .BYTE   DSC$K_CLASS_S
                         00000185' 01D1   152          .ADDRESS LOGNAMS1
                               01D5   153 LEN_256_DESCR:
                             0100  01D5   154          .WORD   256                       ; 60 char length descriptor
                               0E  01D7   155          .BYTE   DSC$K_DTYPE_T
                               01  01D8   156          .BYTE   DSC$K_CLASS_S
                         000001E5' 01D9   157          .ADDRESS LEN_256_NAME
                               01DD   158 LEN_63_DESCR:
                             003B  01DD   159          .WORD   59                        ; 59 char length descriptor
                               0E  01DF   160          .BYTE   DSC$K_DTYPE_T
                               01  01E0   161          .BYTE   DSC$K_CLASS_S
                         000001E5' 01E1   162          .ADDRESS LEN_256_NAME
                               01E5   163 LEN_256_NAME:
4F 4C 5F 41 5F 53 49 5F 53 49 48 54  01E5   164          .ASCII  /THIS_IS_A_LONG_MAILBOX_NAME_WITH/
4E 5F 58 4F 42 4C 49 41 4D 5F 47 4E  01F1
         48 54 49 57 5F 45 4D 41  01FD
5F 36 35 32 30 52 45 42 4D 55 4E 5F  0205   165          .ASCII  /_NUMBER0256_CHARACTERS_IN_IT/  ; CREMBX prefixes MBX$
49 5F 53 52 45 54 43 41 52 41 48 43  0211
                  54 49 5F 4E  021D
41'41'41'41'41'41'41'41'41'41'41'41' 0221   166          .BYTE   ^A/A/[196]
41'41'41'41'41'41'41'41'41'41'41'41' 022D
41'41'41'41'41'41'41'41'41'41'41'41' 0239
41'41'41'41'41'41'41'41'41'41'41'41' 0245
41'41'41'41'41'41'41'41'41'41'41'41' 0251
41'41'41'41'41'41'41'41'41'41'41'41' 025D
41'41'41'41'41'41'41'41'41'41'41'41' 0269
41'41'41'41'41'41'41'41'41'41'41'41' 0275
41'41'41'41'41'41'41'41'41'41'41'41' 0281
41'41'41'41'41'41'41'41'41'41'41'41' 028D
41'41'41'41'41'41'41'41'41'41'41'41' 0299
41'41'41'41'41'41'41'41'41'41'41'41' 02A5
41'41'41'41'41'41'41'41'41'41'41'41' 02B1
```

```
                                                                D 11
41'41'41'41'41'41'41'41'41'41'41'41' 02BD
41'41'41'41'41'41'41'41'41'41'41'41' 02C9
41'41'41'41'41'41'41'41'41'41'41'41' 02D5
                    41'41'41'41'     02E1
                                     02E5   167 ZERO_ADDR_DESCR:                                  ; address 0 for ACCVIO err
                              0009   02E5   168         .WORD   9
                                0E   02E7   169         .BYTE   DSC$K_DTYPE_T
                                01   02E8   170         .BYTE   DSC$K_CLASS_S
                          00000000   02E9   171         .LONG   0
                                     02ED   172 D_LOGIC_NAME:
                              0005   02ED   173         .WORD   5
                                0E   02EF   174         .BYTE   DSC$K_DTYPE_T
                                01   02F0   175         .BYTE   DSC$K_CLASS_S
                          00000004'  02F1   176         .ADDRESS LOGIC_NAME
                                     02F5   177 D_EQUIV_NAME:
                              0005   02F5   178         .WORD   5
                                0E   02F7   179         .BYTE   DSC$K_DTYPE_T
                                01   02F8   180         .BYTE   DSC$K_CLASS_S
                          00000009'  02F9   181         .ADDRESS EQUIV_NAME
                                     02FD   182 D_MBX_LOGIC_NAME:
                              0009   02FD   183         .WORD   9                       ; for prefixing 'MBX$'
                                0E   02FF   184         .BYTE   DSC$K_DTYPE_T
                                01   0300   185         .BYTE   DSC$K_CLASS_S
                          00000000'  0301   186         .ADDRESS MBX_LOGIC_NAME
                                     0305   187 MBA:
          41 42 4D 0000030D'010E0000' 0305   188         .ASCID  \MBA\
                                     0310   189 MSG1:
65 74 61 65 72 43 00000318'010E0000' 0310   190         .ASCID  /Created mailbox was not permanent./
61 77 20 78 6F 62 6C 69 61 6D 20 64  031E
6E 61 6D 72 65 70 20 74 6F 6E 20 73  032A
                    2E 74 6E 65      0336
                                     033A   191 MSGVEC:
                          00000003   033A   192         .LONG   3                       ; PUTMSG message vector
                          00741133   033E   193         .LONG   UETP$_TEXT
                          00000001   0342   194         .LONG   1
                          00002580'  0346   195         .ADDRESS MESSAGEL
                                     034A   196 PROT:
4B 53 4D 4F 52 50 00000352'010E0000' 034A   197         .ASCID  /PROMSK/                ; protection mask for $CREMBX
   52 45 50 55 53 00000360'010E0000' 0358   198 SM:
                                     0358   199         .ASCID  /SUPER/
                                     0365   200 TEST_DATA:
                          00000001   0365   201         A=1
                                     0365   202         .REPT   BUF_SIZ_G/2
                                     0365   203         .WORD   A
                                     0365   204         A=A+1
                              0001   0365   205         .ENDR
                                     0765   206 TEST_MOD_BEGIN:                            ; start end and fail messages
                    6E 75 67 65 62 00' 0765   207         .ASCIC  /begun/
                                 05  0765
                                     076B   208 TEST_MOD_FAIL:
                 64 65 6C 69 61 66 00' 076B   209         .ASCIC  /failed/
                                 06  076B
                                     0772   210 TEST_MOD_NAME:
          37 30 53 53 53 54 41 53 00' 0772   211         .ASCIC  /SATSSS07/              ; needed for SATSMS message
                                 08  0772
                                     077B   212 TEST_MOD_NAME_D:
53 53 53 54 41 53 00000783'010E0000' 077B   213         .ASCID /SATSSS07/               ; module name
```

```
                                37 30  0789
                                       078B   214 TEST_MOD_SUCC:
   6C 75 66 73 73 65 63 63 75 73 00'   078B   215             .ASCIC  /successful/
                                   0A  078B
                                       0796   216 UM:                                    ; mode messages
      72 65 73 75 0000079E'010E0000'   0796   217             .ASCID  \user\
```

```
                     07A2   219 ;
                     07A2   220          .SBTTL  R/W PSECT
                 00000000   221          .PSECT  RWDATA,RD,WRT,NOEXE,LONG
                     0000   222 ;
                     0000   223 ;        ************************************************************
                     0000   224 ;        **                                                        **
                     0000   225 ;        ** The following 2 declaratives must be CONTIGUOUS !! **
                     0000   226 ;        **                                                        **
                     0000   227 ;        ************************************************************
                     0000   228 MBX_LOGIC_NAME:
          24 58 42 4D  0000   229          .ASCII  /MBX$/                  ; for $CREMBX call
                     0004   230 LOGIC_NAME:                              ; for $CRELOG sys service
      42 45 4D 41 4E  0004   231          .ASCII  /NAMEB/
                     0009   232 EQUIV_NAME:
      41 45 4D 41 4E  0009   233          .ASCII  /NAMEA/
                     000E   234 CREATED_FLAG:                            ; signalling successful mailbox creation
                  00 000E   235          .BYTE                           ; for delete service
                     000F   236 ;
                     000F   237 ARGLST1:                                 ; argument list for BUF_CHECK
             00000305' 000F   238          .ADDRESS MBA
             00000023  0013   239          .BLKL   4
                     0023   240 BUF:
             00000073  0023   241          .BLKB   80
                     0073   242 BUFFER:
             00001FB3  0073   243          .BLKB   8000
                     1FB3   244 CHM_CONT:
             00000000  1FB3   245          .LONG   0                       ; change mode continue address
                     1FB7   246 CRE:
                     1FB7   247          $CREMBX 0,MBCHANG,0,0,0,0,0     ; CREMBX paramter list
                     1FD7   248 CTRSTR:
             00000084  1FD7   249          .LONG   132                     ; same as above
             00001FDF' 1FDB   250          .ADDRESS .+4
             00002063  1FDF   251          .BLKB   132
                     2063   252 CURRENT_TC:
             00000000  2063   253          .LONG   0                       ; ptr to current test case
                     2067   254          .ALIGN LONG                     ; put it on a long word boundry
                     2068   255 DEL:
                     2068   256          $DELMBX MBCHANG                 ; DELMBX parameter list
                     2070   257 GETBUF:
             00000084  2070   258          .LONG   132                     ; same as above
             00002078' 2074   259          .ADDRESS .+4
             000020FC  2078   260          .BLKB   132
                     20FC   261 MBCHAN:
                 0000  20FC   262          .WORD   0                       ; channel for the $GETCHN
                     20FE   263 MBCHANG:                                 ; _G mailbox channel
                 0000  20FE   264          .WORD   0
                     2100   265 MBCHANS:                                 ; _S mailbox channel
                 0000  2100   266          .WORD   0
                     2102   267 MBCHAND:                                 ; _D mailbox channel for deletions
                 0000  2102   268          .WORD   0
                     2104   269 MBCHAR:
             00000074  2104   270          .LONG   DIB$K_LENGTH            ; length of PROMSK for $GETCHN
             0000210C' 2108   271          .ADDRESS .+4
                     210C   272 MBCHR:
             00002180  210C   273          .BLKB   DIB$K_LENGTH            ; buffer for PROMSK in $GETCHN
                     2180   274 MBX_BUF:
             00002580  2180   275          .BLKB   BUF_SIZ_G               ; buffer for QIO reads
```

```
                              2580  MESSAGEL:
                  00000000    2580          .LONG   0                   ; message desc.
                  00000023'   2584          .ADDRESS BUF
                  00002078'   2588          .ADDRESS GETBUF+8
                              258C  ML:
                  00000000    258C          .LONG   0                   ; desc. for BUF_CHECK routine
                  00002078'   2590          .ADDRESS GETBUF+8
                              2594  MOD_MSG_CODE:
                  007480D9    2594          .LONG   UETP$_SATSMS        ; test module message code for putmsg
                              2598  MODE:
                  00000000    2598          .LONG   0                   ; current mode string pointer
                              259C  MSGL:
                  00000050    259C          .LONG   80                  ; buffer desc.
                  00000023'   25A0          .ADDRESS BUF
                              25A4  MSGVEC1:                            ; PUTMSG message vector
                  00000003    25A4          .LONG   3
                  00741133    25A8          .LONG   UETP$_TEXT
                  00000001    25AC          .LONG   1
                  00000000    25B0          .LONG   0
                              25B4  PB:
                  00000074    25B4          .LONG   DIB$K_LENGTH
                  000025BC'   25B8          .ADDRESS .+4
                  00002630    25BC          .BLKB   DIB$K_LENGTH
                              2630  PRIVMASK:
         00000000 00000000    2630          .QUAD   0                   ; priv. mask
                              2638  PRVHND1:
                  00000000    2638          .LONG   0                   ; previous handler address 1
                              263C  PRVPRT:
                        00    263C          .BYTE   0                   ; protection return byte for SETPRT
                              263D  REG:
74 73 69 67 65 72 00002645'010E0000'  263D  .ASCID  \register R\
               52 20 72 65    264B
                              264F  REG_SAVE_AREA:
                  0000268B    264F          .BLKL   15                  ; register save area
                              268B  REGNUM:
                  00000000    268B          .LONG   0                   ; register number
                              268F  RETADR:
                  00002697    268F          .BLKL   2                   ; returned address's from SETPRT
                              2697  SERV_NAME:
                  00000000    2697          .LONG   0                   ; service name pointer
                              269B  STATUS:
                  00000000    269B          .LONG   0
                              269F  TMD_ADDR:
                  00000765'   269F          .ADDRESS TEST_MOD_BEGIN
                              26A3  TMN_ADDR:
                  00000772'   26A3          .ADDRESS TEST_MOD_NAME
                              26A7  TPID:
                  00000000    26A7          .LONG   0                   ; PID for this process
                              26AB  MSG1L:
                  00000100    26AB          .LONG   TEXT_BUFFER         ; Buffer #1 desc.
                  000026B3'   26AF          .ADDRESS BUF1
                              26B3
                              26B3  BUF1:
                  000027B3    26B3          .BLKB   TEXT_BUFFER
                              27B3
                              27B3  MESSAGE1L:
                  00000000    27B3          .LONG   0                   ; Message length
```

```
                                                H 11
000026B3' 27B7   332             .ADDRESS BUF1
          27BB   333 BUFFER_PTR:
0000 0100 27BB   334             .WORD   TEXT_BUFFER,0          ; Fake .ASCID buffer for misc. strings
00000073' 27BF   335             .ADDRESS BUFFER                ; A word for length, a word for desc.
```

```
00000000   337          .PSECT  SATSSS07,RD,WRT,EXE,LONG
     0000  338          .SBTTL  SATSSS07
     0000  339 ;++
     0000  340 ; FUNCTIONAL DESCRIPTION:
     0000  341 ;
     0000  342 ;       After performing some initial housekeeping, such as
     0000  343 ; printing the module begin message and acquiring needed privileges,
     0000  344 ; the system services are tested in each of their normal conditions.
     0000  345 ; Detected failures are identified and  an error message is printed
     0000  346 ; on the terminal.  Upon completion of the test a success or fail
     0000  347 ; message is printed on the terminal.
     0000  348 ;
     0000  349 ; CALLING SEQUENCE:
     0000  350 ;
     0000  351 ;       $ RUN SATSSS07  ...  (DCL COMMAND)
     0000  352 ;
     0000  353 ; INPUT PARAMETERS:
     0000  354 ;
     0000  355 ;       none
     0000  356 ;
     0000  357 ; IMPLICIT INPUTS:
     0000  358 ;
     0000  359 ;       none
     0000  360 ;
     0000  361 ; OUTPUT PARAMETERS:
     0000  362 ;
     0000  363 ;       none
     0000  364 ;
     0000  365 ; IMPLICIT OUTPUTS:
     0000  366 ;
     0000  367 ;       Messages to SYS$OUTPUT are the only output from SATSSS07.
     0000  368 ;       They are of the form:
     0000  369 ;
     0000  370 ;               %UETP-S-SATSMS, TEST MODULE SATSSS07 BEGUN ... (BEGIN MSG)
     0000  371 ;               %UETP-S-SATSMS, TEST MODULE SATSSS07 SUCCESSFUL ... (END MSG)
     0000  372 ;               %UETP-E-SATSMS, TEST MODULE SATSSS07 FAILED ... (END MSG)
     0000  373 ;               %UETP-I-TEXT, ... (VARIABLE INFORMATION ABOUT A TEST MODULE FAILURE)
     0000  374 ;
     0000  375 ; COMPLETION CODES:
     0000  376 ;
     0000  377 ;       The SATSSS07 routine terminates with a $EXIT to the
     0000  378 ;       operating system with a status code defined by UETP$_SATSMS.
     0000  379 ;
     0000  380 ; SIDE EFFECTS:
     0000  381 ;
     0000  382 ;       none
     0000  383 ;
     0000  384 ;--
     0000  385
     0000  386          TEST_START SATSSS07                     ; let the test begin
```

```
                                      0000
                                0000  0000              .ENTRY SATSSS07,0
                  2063'CF   D4  0002              CLRL    W^CURRENT_TC
                       00   DD  0006              PUSHL   #0
                  26A7'CF   DF  0008              PUSHAL  W^TPID
         00000000'GF   02   FB  000C              CALLS   #2,G^SYS$WAKE
         00000000'GF   00   FB  0013              CALLS   #0,G^SYS$HIBER
                  077B'CF   7F  001A              PUSHAQ  W^TEST_MOD_NAME_D
         00000000'GF   01   FB  001E              CALLS   #1,G^SYS$SETPRN
                     15C8   30  0025              BSBW    W^MOD_MSG_PRINT
         269F'CF  078B'CF   DE  0028              MOVAL   W^TEST_MOD_SUCC,W^TMD_ADDR
2594'CF   03   00      01   F0  002F              INSV    #SUCCESS,#0,#3,W^MOD_MSG_CODE
                       00   DD  0036              PUSHL   #0
            0E16'CF    01   FB  0038              CALLS #1,W^REG_SAVE
                                003D      STP0:
```

```
                                003D   388          .SBTTL  CREMBX TESTS
                                003D   389 ;
                                003D   390 ;
                                003D   391 ; $CREMBX tests
                                003D   392 ;
                                003D   393 ; Test temporary mailbox.
                                003D   394 ;
                                003D   395 ;-
    2697'CF    0008'CF    DE   003D   396          MOVAL   W^CREMBX,W^SERV_NAME            ; set service name
    2598'CF    0796'CF    DE   0044   397          MOVAL   W^UM,W^MODE                     ; set mode
                                004B   398          $CREMBX_S PRMFLG=#0,-
                                004B   399                    CHAN  =W^MBCHANS,-        ; try _s form
                                004B   400                    MAXMSG=#MBBUF,-           ; set the max. message size
                                004B   401                    BUFQUO=#BUF_SIZ_S,-       ; set the buffer quota
                                004B   402                    LOGNAM=W^LOGNAMS          ; try _s form
                                006C   403          FAIL_CHECK SS$_NORMAL              ; check success
                      01    DD  006C                        PUSHL   #SS$_NORMAL
            0E20'CF   01    FB  006E                        CALLS   #1,W^REG_CHECK
                 1FBB'CF    D4  0073   404          CLRL    W^CRE+CREMBX$_PRMFLG       ; make it temporary
    1FC3'CF    00000400    8F  D0  0077   405          MOVL    #MBBUF,W^CRE+CREMBX$_MAXMSG ; set the max. message size
    1FC7'CF    00000400    8F  D0  0080   406          MOVL    #BUF_SIZ_G,W^CRE+CREMBX$_BUFQUO ; set the buffer quota
        1FD3'CF    0154'CF  DE  0089   407          MOVAL   W^LOGNAMG,W^CRE+CREMBX$_LOGNAM ; set the lognam
                                0090   408          $CREMBX_G W^CRE                    ; try _g form
                                0099   409          FAIL_CHECK SS$_NORMAL              ; check success
                      01    DD  0099                        PUSHL   #SS$_NORMAL
            0E20'CF   01    FB  009B                        CALLS   #1,W^REG_CHECK
            0CFC'CF   00    FB  00A0   410          CALLS   #0,W^VERIFY_MBX            ; read,write,verify,delete MBX
            0C3D'CF   00    FB  00A5   411          CALLS   #0,W^ERLBUF_DUMP           ; dump errors
                                00AA   412 ;+
                                00AA   413 ;
                                00AA   414 ; Test permanent mailbox
                                00AA   415 ;
                                00AA   416 ;-
                                00AA   417          NEXT_TEST
                                00AA
                                00AA       STP1:
        2063'CF   01    DO  00AA                        MOVL    #1,W^CURRENT_TC
                  00    DD  00AF                        PUSHL   #0
            0E16'CF   01    FB  00B1                    CALLS   #1,W^REG_SAVE
    2697'CF    0008'CF    DE   00B6   418          MOVAL   W^CREMBX,W^SERV_NAME            ; set service name
    2598'CF    0796'CF    DE   00BD   419          MOVAL   W^UM,W^MODE                     ; set mode
                                00C4   420          $CREMBX_S PRMFLG=#1,-
                                00C4   421                    CHAN  =W^MBCHANS,-
                                00C4   422                    MAXMSG=#MBBUF,-
                                00C4   423                    BUFQUO=#BUF_SIZ_S,-
                                00C4   424                    LOGNAM=W^LOGNAMS1         ; try _s form
                                00E5   425          FAIL_CHECK SS$_NORMAL              ; check success
                      01    DD  00E5                        PUSHL   #SS$_NORMAL
            0E20'CF   01    FB  00E7                        CALLS   #1,W^REG_CHECK
                                00EC   426          $DASSGN_S CHAN=W^MBCHANS           ; deassign mailbox
                                00F8   427          $ASSIGN_S DEVNAM=W^LOGNAMS1,-
                                00F8   428                    CHAN  =W^MBCHANS         ; try to reassign MBX
                01    50    D1  0109   429          CMPL    R0,#SS$_NORMAL             ; is the MBX permanent?
                      02    13  010C   430          BEQL    10$                        ; br if yes
                      50    11  010E   431          BRB     20$                        ; print perm. MBX error
                                0110   432 10$:
        1FBB'CF   01    DO  0110   433          MOVL    #1,W^CRE+CREMBX$_PRMFLG         ; set prmflg for perm. MBX
```

```
                                                L 11
1FC3'CF    00000400 8F    D0  0115   434          MOVL    #MBBUF,W^CRE+CREMBX$_MAXMSG     ; set the max. message size
1FC7'CF    00000400 8F    D0  011E   435          MOVL    #BUF_SIZ_G,W^CRE+CREMBX$_BUFQUO ; set the buffer quota_G
    1FD3'CF    0164'CF    DE  0127   436          MOVAL   W^LOGNAMG1,W^CRE+CREMBX$_LOGNAM ; set the lognam
                          0127   437          $CREMBX_G W^CRE                         ; try _G form
                          0137   438          FAIL_CHECK SS$_NORMAL                   ; check success
                    01    DD  0137                    PUSHL   #SS$_NORMAL
            0E20'CF 01    FB  0139                    CALLS   #1,W^REG_CHECK
                          013E   439          $DASSGN_S CHAN=W^MBCHANG                ; deassign the channel
                          014A   440          $ASSIGN_S DEVNAM=W^LOGNAMG1,-
                          014A   441                    CHAN  =W^MBCHANG              ; assign the mailbox
                50  01    D1  015B   442          CMPL    R0,#SS$_NORMAL              ; is the MBX permanent?
                09  13    BEQL    443          BEQL    30$                         ; br if yes
                          0160   444 20$:
            0310'CF DF    PUSHAL  W^MSG1
            1501'CF 01    FB  0164   446          CALLS   #1,W^PRINT_FAIL             ; print failure
                          0169   447 30$:
            0CFC'CF 00    FB  0169   448          CALLS   #0,W^VERIFY_MBX             ; read,write,verify,delete MBX
            0C3D'CF 00    FB  016E   449          CALLS   #0,W^ERLBUF_DUMP            ; dump errors
                          0173   450 ;+
                          0173   451 ;
                          0173   452 ; $DELMBX test
                          0173   453 ;
                          0173   454 ;-
                          0173   455          $ASSIGN_S DEVNAM=W^LOGNAMS1,-
                          0173   456                    CHAN  =W^MBCHANS              ; reassign the mailbox_S
        2697'CF 0120'CF   DE  0184   457          MOVAL   W^DELMBX,W^SERV_NAME        ; set service name
                00  DD  018B   458          PUSHL   #0
            0E16'CF 01    FB  018D   459          CALLS   #1,W^REG_SAVE               ; save the registers
                          0192   460          $DELMBX_S CHAN=W^MBCHANS                ; delete the _S mailbox
                          019E   461          FAIL_CHECK SS$_NORMAL
                    01    DD  019E                    PUSHL   #SS$_NORMAL
            0E20'CF 01    FB  01A0                    CALLS   #1,W^REG_CHECK
                          01A5   462          $ASSIGN_S DEVNAM=W^LOGNAMG1,-
                          01A5   463                    CHAN  =W^MBCHANG              ; reassign the mailbox_S
206C'CF    20FE'CF    3C  01B6   464          MOVZWL  W^MBCHANG,W^DEL+DELMBX$_CHAN
                          01BD   465          $DELMBX_G W^DEL                         ; delete the _G mailbox
                          01C6   466          FAIL_CHECK SS$_NORMAL
                    01    DD  01C6                    PUSHL   #SS$_NORMAL
            0E20'CF 01    FB  01C8                    CALLS   #1,W^REG_CHECK
                          01CD   467 ;+
                          01CD   468 ;
                          01CD   469 ; Test MAXMSG low limit (1)
                          01CD   470 ;
                          01CD   471 ;-
                          01CD   472          NEXT_TEST
                          01CD
                          01CD        STP2:
            2063'CF 02    DO  01CD                    MOVL    #2,W^CURRENT_TC
                00  DD  01D2                    PUSHL   #0
            0E16'CF 01    FB  01D4                    CALLS   #1,W^REG_SAVE
2697'CF    0008'CF    DE  01D9   473          MOVAL   W^CREMBX,W^SERV_NAME        ; set service name
2598'CF    0796'CF    DE  01E0   474          MOVAL   W^UM,W^MODE                 ; set mode
                          01E7   475          $CREMBX_S PRMFLG=#0,-
                          01E7   476                    CHAN  =W^MBCHANS,-
                          01E7   477                    MAXMSG=#LOMSG,-             ; set the max. message size
                          01E7   478                    BUFQUO=#BUF_SIZ_S,-         ; set the buffer size
                          01E7   479                    LOGNAM=W^LOGNAMS            ; try _s form
```

```
                                      0204      FAIL_CHECK SS$_NORMAL                  ; check success
                          01      DD  0204              PUSHL   #SS$_NORMAL
              0E20'CF     01      FB  0206              CALLS   #1,W^REG_CHECK
                                      020B
                1FBB'CF           D4  020B      CLRL    W^CRE+CREMBX$_PRMFLG           ; make it temporary
              1FC3'CF     01      D0  020F      MOVL    #LOMSG,W^CRE+CREMBX$_MAXMSG    ; set the max. message size
      1FC7'CF    00000400 8F      D0  0214      MOVL    #BUF_SIZ_G,W^CRE+CREMBX$_BUFQUO ; set the buffer quota
         1FD3'CF     0154'CF      DE  021D      MOVAL   W^LOGNAMG,W^CRE+CREMBX$_LOGNAM ; move lognam in
                                      0224      $CREMBX_G W^CRE                        ; try _g form
                                      022D      FAIL_CHECK SS$_NORMAL                  ; check success
                          01      DD  022D              PUSHL   #SS$_NORMAL
              0E20'CF     01      FB  022F              CALLS   #1,W^REG_CHECK
                                      0234      $QIO_S  CHAN=W^MBCHANS,-
                                      0234              FUNC=#IO$_WRITEVBLK,-
                                      0234              P1  =W^TEST_DATA,-
                                      0234              P2  =#LOMSG                    ; write to the mailbox_S
  0100 8F    00   2180'CF     00      2C  0253      MOVC5   #0,W^MBX_BUF,#0,#256,W^MBX_BUF  ; zero the MBX buffer
                  2180'CF             025C
                                      025F      $QIO_S  CHAN=W^MBCHANS,-
                                      025F              FUNC=#IO$_READVBLK,-
                                      025F              P1  =W^MBX_BUF,-
                                      025F              P2  =#LOMSG                    ; read from the mailbox
              56  2180'CF         DE  027E      MOVAL   W^MBX_BUF,R6                   ; set the MBX buffer
              57  0365'CF         DE  0283      MOVAL   W^TEST_DATA,R7                 ; find the master data
                  58      01      D0  0288      MOVL    #LOMSG,R8
         20FC'CF  2100'CF         B0  028B      MOVW    W^MBCHANS,W^MBCHAN             ; get the channel number
              0C7A'CF     00      FB  0292      CALLS   #0,W^BUF_CHECK                 ; check the data
                                      0297      $DASSGN_S CHAN=W^MBCHANS               ; deassign the channel
                                      02A3      $QIO_S  CHAN=W^MBCHANG,-
                                      02A3              FUNC=#IO$_WRITEVBLK,-
                                      02A3              P1  =W^TEST_DATA,-
                                      02A3              P2  =#LOMSG                    ; write to the mailbox_G
  0400 8F    00   2180'CF     00      2C  02C2      MOVC5   #0,W^MBX_BUF,#0,#1024,W^MBX_BUF ; zero the MBX buffer
                  2180'CF             02CB
                                      02CE      $QIO_S  CHAN=W^MBCHANG,-
                                      02CE              FUNC=#IO$_READVBLK,-
                                      02CE              P1  =W^MBX_BUF,-
                                      02CE              P2  =#LOMSG                    ; read from the mailbox
              56  2180'CF         DE  02ED      MOVAL   W^MBX_BUF,R6                   ; set the MBX buffer
              57  0365'CF         DE  02F2      MOVAL   W^TEST_DATA,R7                 ; find the master data
                  58      01      D0  02F7      MOVL    #LOMSG,R8
         20FC'CF  20FE'CF         B0  02FA      MOVW    W^MBCHANG,W^MBCHAN             ; get the channel number
              0C7A'CF     00      FB  0301      CALLS   #0,W^BUF_CHECK                 ; check the data
                                      0306      $DASSGN_S CHAN=W^MBCHANG               ; deassign the channel
                                      0312  ;
                                      0312  ; Test BUFQUO using BUFQUO = 256 using _S form
                                      0312  ;                   BUFQUO = 512 using _G form
                                      0312  ;
                                      0312  ;-
                                      0312          NEXT_TEST
                                      0312
                                      0312  STP3:
              2063'CF     03      D0  0312                  MOVL    #3,W^CURRENT_TC
                          00      DD  0317                  PUSHL   #0
              0E16'CF     01      FB  0319                  CALLS   #1,W^REG_SAVE
         2697'CF  0008'CF         DE  031E      MOVAL   W^CREMBX,W^SERV_NAME           ; set service name
         2598'CF  0796'CF         DE  0325      MOVAL   W^UM,W^MODE                    ; set mode
```

```
                                 032C    526          $CREMBX_S PRMFLG=#0,-
                                 032C    527                    CHAN  =W^MBCHANS,-
                                 032C    528                    MAXMSG=#MBBUF,-
                                 032C    529                    BUFQUO=#BUF_SIZ_S,-
                                 032C    530                    LOGNAM=W^LOGNAMS                ; try _S
                                 034D    531          FAIL_CHECK SS$_NORMAL                     ; check success
                         01      DD      034D                   PUSHL   #SS$_NORMAL
                 0E20'CF 01      FB      034F                   CALLS   #1,W^REG_CHECK
                     1FBB'CF     D4      0354    532          CLRL    W^CRE+CREMBX$_PRMFLG
        1FC3'CF   00000400 8F    D0      0358    533          MOVL    #MBBUF,W^CRE+CREMBX$_MAXMSG     ; set the max. message size
        1FC7'CF   00000200 8F    D0      0361    534          MOVL    #BUF_SIZ_S*2,W^CRE+CREMBX$_BUFQUO ; set the buffer quota
            1FD3'CF   0154'CF    DE      036A    535          MOVAL   W^LOGNAM,W^CRE+CREMBX$_LOGNAM
                                 0371    536          $CREMBX_G W^CRE                           ; try _G form
                                 037A    537          FAIL_CHECK SS$_NORMAL                     ; check success
                         01      DD      037A                   PUSHL   #SS$_NORMAL
                 0E20'CF 01      FB      037C                   CALLS   #1,W^REG_CHECK
                                 0381    538          $QIO_S  CHAN=W^MBCHANS,-
                                 0381    539                  FUNC=#IO$_WRITEVBLK,-
                                 0381    540                  P1  =W^TEST_DATA,-
                                 0381    541                  P2  =#BUF_SIZ_S                   ; write to the mailbox_S
  0100 8F    00    2180'CF    00    2C    03A4    542          MOVC5   #0,W^MBX_BUF,#0,#256,W^MBX_BUF  ; zero the mailbox buffer
                         2180'CF         03AD
                                 03B0    543          $QIO_S  CHAN=W^MBCHANS,-
                                 03B0    544                  FUNC=#IO$_READVBLK,-
                                 03B0    545                  P1  =W^MBX_BUF,-
                                 03B0    546                  P2  =#BUF_SIZ_S                   ; read from the mailbox_S
                 56      2180'CF DE      03D3    547          MOVAL   W^MBX_BUF,R6              ; set the MBX buffer
                 57      0365'CF DE      03D8    548          MOVAL   W^TEST_DATA,R7            ; find the master data
              58    00000100 8F    D0      03DD    549          MOVL    #BUF_SIZ_S,R8
         20FC'CF      2100'CF    B0      03E4    550          MOVW    W^MBCHANS,W^MBCHAN        ; get the channel number
                 0C7A'CF 00      FB      03EB    551          CALLS   #0,W^BUF_CHECK            ; check the data
                                 03F0    552          $DASSGN_S CHAN=W^MBCHANS                  ; deassign the channel
         20FC'CF      2100'CF    B0      03FC    553          MOVW    W^MBCHANS,W^MBCHAN        ; get the channel number
                                 0403    554          $QIO_S  CHAN=W^MBCHANG,-
                                 0403    555                  FUNC=#IO$_WRITEVBLK,-
                                 0403    556                  P1  =W^TEST_DATA,-
                                 0403    557                  P2  =#BUF_SIZ_S*2                 ; write to the mailbox_G
  0200 8F    00    2180'CF    00    2C    0426    558          MOVC5   #0,W^MBX_BUF,#0,#512,W^MBX_BUF  ; zero the mailbox buffer
                         2180'CF         042F
                                 0432    559          $QIO_S  CHAN=W^MBCHANG,-
                                 0432    560                  FUNC=#IO$_READVBLK,-
                                 0432    561                  P1  =W^MBX_BUF,-
                                 0432    562                  P2  =#BUF_SIZ_S*2                 ; read from the mailbox_G
                 56      2180'CF DE      0455    563          MOVAL   W^MBX_BUF,R6              ; set the mailbox buffer
                 57      0365'CF DE      045A    564          MOVAL   W^TEST_DATA,R7            ; find the master data
              58    00000200 8F    D0      045F    565          MOVL    #BUF_SIZ_S*2,R8
         20FC'CF      20FE'CF    B0      0466    566          MOVW    W^MBCHANG,W^MBCHAN        ; get the channel number
                 0C7A'CF 00      FB      046D    567          CALLS   #0,W^BUF_CHECK            ; check the data
                                 0472    568          $DASSGN_S CHAN=W^MBCHANG                  ; deassign the channel
                                 047E    569 ;+
                                 047E    570 ;
                                 047E    571 ; Test PROMSK (protection mask)
                                 047E    572 ;
                                 047E    573 ;-
                                 047E    574          NEXT_TEST
                                 047E
                                 047E        STP4:
```

```
                2063'CF    04    D0  047E                        MOVL    #4,W^CURRENT_TC
                           00    DD  0483                        PUSHL   #0
                0E16'CF    01    FB  0485                        CALLS   #1,W^REG_SAVE
        2697'CF    0008'CF    DE  048A   575          MOVAL   W^CREMBX,W^SERV_NAME            ; set service name
        2598'CF    0796'CF    DE  0491   576          MOVAL   W^UM,W^MODE                     ; set mode
                              0498   577          $CREMBX_S PRMFLG=#0,-
                              0498   578                    CHAN  =W^MBCHANS,-
                              0498   579                    MAXMSG=#MBBUF,-
                              0498   580                    BUFQUO=#BUF_SIZ_S,-
                              0498   581                    PROMSK=#^X3303,-
                              0498   582                    LOGNAM=LOGNAMS           ; try _s form
                              04BD   583          FAIL_CHECK SS$_NORMAL            ; check success
                           01    DD  04BD                        PUSHL   #SS$_NORMAL
                0E20'CF    01    FB  04BF                        CALLS   #1,W^REG_CHECK
                              04C4   584          $GETCHN_S CHAN=W^MBCHANS,-
                              04C4   585                    PRIBUF=W^MBCHAR          ; get the MBX_S channel char.
        2124'CF    3303 8F    B1  04DA   586          CMPW    #^X3303,W^MBCHR+DIB$W_VPROT
                           13    13  04E1   587          BEQL    10$                     ; branch if O.K.
                     2124'CF    DD  04E3   588          PUSHL   W^MBCHR+DIB$W_VPROT
                 00003303 8F    DD  04E7   589          PUSHL   #^X3303
                     034A'CF    DF  04ED   590          PUSHAL  W^PROT
                1501'CF    03    FB  04F1   591          CALLS   #3,W^PRINT_FAIL         ; print the error
                              04F6   592 10$:
                     1FBB'CF    D4  04F6   593          CLRL    W^CRE+CREMBX$_PRMFLG    ; make it temporary
    1FC3'CF    00000400 8F    D0  04FA   594          MOVL    #MBBUF,W^CRE+CREMBX$_MAXMSG     ; set max. message size
    1FC7'CF    00000400 8F    D0  0503   595          MOVL    #BUF_SIZ_G,W^CRE+CREMBX$_BUFQUO ; set the buffer quota
    1FCB'CF    00003303 8F    D0  050C   596          MOVL    #^X3303,W^CRE+CREMBX$_PROMSK    ; clear protection mask
        1FD3'CF    0154'CF    DE  0515   597          MOVAL   W^LOGNAMG,W^CRE+CREMBX$_LOGNAM  ; set the lognam
                              051C   598          $CREMBX_G W^CRE                 ; try _g form
                              0525   599          FAIL_CHECK SS$_NORMAL           ; check success
                           01    DD  0525                        PUSHL   #SS$_NORMAL
                0E20'CF    01    FB  0527                        CALLS   #1,W^REG_CHECK
                              052C   600          $GETCHN_S CHAN=W^MBCHANG,-
                              052C   601                    PRIBUF=W^MBCHAR          ; get MBX_G channel char.
    2124'CF    3303 8F    B1  0542   602          CMPW    #^X3303,W^MBCHR+DIB$W_VPROT     ; CHECK PROMSK
                           13    13  0549   603          BEQL    20$                     ; BR if O.K.
                     2124'CF    DD  054B   604          PUSHL   W^MBCHR+DIB$W_VPROT
                 00003303 8F    DD  054F   605          PUSHL   #^X3303
                     034A'CF    DF  0555   606          PUSHAL  W^PROT
                1501'CF    03    FB  0559   607          CALLS   #3,W^PRINT_FAIL         ; print the error
                              055E   608 20$:
                0CFC'CF    00    FB  055E   609          CALLS   #0,W^VERIFY_MBX         ; read,write,verify,delete MB
                0C3D'CF    00    FB  0563   610          CALLS   #0,W^ERLBUF_DUMP        ; dump errors
                              0568   611 ;+
                              0568   612 ;
                              0568   613 ; Test ACMODE (super mode)
                              0568   614 ;
                              0568   615 ;-
                              0568   616          NEXT_TEST
                              0568
                              0568       STP5:
                2063'CF    05    D0  0568                        MOVL    #5,W^CURRENT_TC
                           00    DD  056D                        PUSHL   #0
                0E16'CF    01    FB  056F                        CALLS   #1,W^REG_SAVE
        2697'CF    0008'CF    DE  0574   617          MOVAL   W^CREMBX,W^SERV_NAME            ; set service name
        2598'CF    0358'CF    DE  057B   618          MOVAL W^SM,W^MODE                       ; declare super mode
                              0582   619          $CMKRNL_S W^SETUP_SUPER,W^ARGLST        ; declare CHMS handler
```

```
                5E   10'  C0  0591   620          ADDL2   S^#EXE$C_CMSTKSZ+16,SP          ; adjust user stack ptr.
                5D   5E   D0  0594   621          MOVL    SP,FP                           ; fix the FP
          0C3D'CF    00   FB  0597   622          CALLS   #0,W^ERLBUF_DUMP                ; dump errors
                     01   BE  059C   623          CHMS    #1                              ; do the super tests
          0C3D'CF    00   FB  059E   624          CALLS   #0,W^ERLBUF_DUMP                ; dump any errors
                          05A3   625 ;+
                          05A3   626 ;
                          05A3   627 ; Test EXEC mode
                          05A3   628 ;
                          05A3   629 ;-
                          05A3   630          NEXT_TEST
                          05A3
                          05A3          STP6:
          2063'CF    06   D0  05A3                  MOVL    #6,W^CURRENT_TC
                     00   DD  05A8                  PUSHL   #0
          0E16'CF    01   FB  05AA                  CALLS   #1,W^REG_SAVE
    2697'CF    0008'CF   DE  05AF   631          MOVAL   W^CREMBX,W^SERV_NAME            ; set service name
    2598'CF    0127'CF   DE  05B6   632          MOVAL   W^EM,W^MODE                     ; declare exec mode
                         05BD   633          $CMEXEC_S B^10$                         ; go to exec mode
                0067     31  05C9   634          BRW     20$
                         05CC   635 10$:
                     00   DD  05CC   636          PUSHL   #0
          0E16'CF    01   FB  05CE   637          CALLS   #1,W^REG_SAVE                   ; save the registers
                         05D3   638          $CREMBX_S PRMFLG=#0,-
                         05D3   639                    CHAN  =W^MBCHANS,-
                         05D3   640                    MAXMSG=#MBBUF,-                 ; set the max. message size
                         05D3   641                    BUFQUO=#BUF_SIZ_S,-             ; set the buffer quota
                         05D3   642                    ACMODE=#PSL$C_EXEC,-            ; set the access mode (super)
                         05D3   643                    LOGNAM=W^LOGNAMS                ; try _S form
                         05F4   644          FAIL_CHECKNP SS$_NORMAL                 ; check success
                     01   DD  05F4                  PUSHL   #SS$_NORMAL
          1443'CF    01   FB  05F6                  CALLS   #1,W^REG_CHECKNP
               1FBB'CF   D4  05FB   645          CLRL    W^CRE+CREMBX$_PRMFLG            ; make it temporary
1FC3'CF     00000400 8F   D0  05FF   646          MOVL    #MBBUF,W^CRE+CREMBX$_MAXMSG     ; set the max. message size
1FC7'CF     00000400 8F   D0  0608   647          MOVL    #BUF_SIZ_G,W^CRE+CREMBX$_BUFQUO ; set the buffer quota
     1FCF'CF    01   D0  0611   648          MOVL    #PSL$C_EXEC,W^CRE+CREMBX$_ACMODE ; move in ACMODE
1FD3'CF     0154'CF   DE  0616   649          MOVAL   W^LOGNAMG,W^CRE+CREMBX$_LOGNAM  ; set the lognam
                         061D   650          $CREMBX_G W^CRE                         ; try _G form
                         0626   651          FAIL_CHECKNP SS$_NORMAL                 ; check success
                     01   DD  0626                  PUSHL   #SS$_NORMAL
          1443'CF    01   FB  0628                  CALLS   #1,W^REG_CHECKNP
          0CFC'CF    00   FB  062D   652          CALLS   #0,W^VERIFY_MBX                 ; read,write,verify,delete MBX
                     04   0632   653          RET
                          0633   654 20$:
          0C3D'CF    00   FB  0633   655          CALLS   #0,W^ERLBUF_DUMP                ; dump errors
                          0638   656 ;+
                          0638   657 ;
                          0638   658 ; Test KERNEL mode
                          0638   659 ;
                          0638   660 ;-
                          0638   661          NEXT_TEST
                          0638
                          0638          STP7:
          2063'CF    07   D0  0638                  MOVL    #7,W^CURRENT_TC
                     00   DD  063D                  PUSHL   #0
          0E16'CF    01   FB  063F                  CALLS   #1,W^REG_SAVE
    2697'CF    0008'CF   DE  0644   662          MOVAL   W^CREMBX,W^SERV_NAME            ; set service name
```

```
  2598'CF   0146'CF   DE  064B   663          MOVAL   W^KM,W^MODE                          ; set mode
                          0652   664          $CMKRNL_S B^10$
                0067  31  065E   665          BRW     20$
                          0661   666 10$:
                  00  DD  0661   667          PUSHL   #0
     0E16'CF      01  FB  0663   668          CALLS   #1,W^REG_SAVE                        ; save the registers
                          0668   669          $CREMBX_S PRMFLG=#0,-
                          0668   670                  CHAN  =W^MBCHANS,-
                          0668   671                  MAXMSG=#MBBUF,-
                          0668   672                  BUFQUO=#BUF_SIZ_S,-
                          0668   673                  ACMODE=#PSL$C_KERNEL,-
                          0668   674                  LOGNAM=W^LOGNAMS                     ; try _s form
                          0689   675          FAIL_CHECKNP SS$_NORMAL                      ; check success
                  01  DD  0689                        PUSHL   #SS$_NORMAL
     1443'CF      01  FB  068B                        CALLS   #1,W^REG_CHECKNP
         1FBB'CF      D4  0690   676          CLRL    W^CRE+CREMBX$_PRMFLG                 ; make it temporary
 1FC3'CF   00000400 8F DO 0694   677          MOVL    #MBBUF,W^CRE+CREMBX$_MAXMSG          ; get MAXMSG parameter
 1FC7'CF   00000400 8F DO 069D   678          MOVL    #BUF_SIZ_G,W^CRE+CREMBX$_BUFQUO      ; set the buffer quota
     1FCF'CF      00  D0  06A6   679          MOVL    #PSL$C_KERNEL,W^CRE+CREMBX$_ACMODE   ; move in ACMODE
  1FD3'CF   0154'CF   DE  06AB   680          MOVAL   W^LOGNAMG,W^CRE+CREMBX$_LOGNAM       ; set the lognam
                          06B2   681          $CREMBX_G W^CRE                              ; try _g form
                          06BB   682          FAIL_CHECKNP SS$_NORMAL                      ; check success
                  01  DD  06BB                        PUSHL   #SS$_NORMAL
     1443'CF      01  FB  06BD                        CALLS   #1,W^REG_CHECKNP
     0CFC'CF      00  FB  06C2   683          CALLS   #0,W^VERIFY_MBX                      ; read,write,verify,delete MBX
                      04  06C7   684          RET
                          06C8   685 20$:
     0C3D'CF      00  FB  06C8   686          CALLS   #0,W^ERLBUF_DUMP                     ; dump errors
                          06CD   687 ;+
                          06CD   688 ;
                          06CD   689 ; Test USER mode
                          06CD   690 ;
                          06CD   691 ;-
                          06CD   692          NEXT_TEST
                          06CD
                          06CD       STP8:
     2063'CF      08  D0  06CD                        MOVL    #8,W^CURRENT_TC
                  00  DD  06D2                        PUSHL   #0
     0E16'CF      01  FB  06D4                        CALLS   #1,W^REG_SAVE
  2598'CF   0796'CF   DE  06D9   693          MOVAL   W^UM,W^MODE                          ; reset user mode
  2697'CF   0008'CF   DE  06E0   694          MOVAL   W^CREMBX,W^SERV_NAME                 ; set service name
                          06E7   695          $CREMBX_S PRMFLG=#0,-
                          06E7   696                  CHAN  =W^MBCHANS,-
                          06E7   697                  MAXMSG=#MBBUF,-
                          06E7   698                  BUFQUO=#BUF_SIZ_S,-
                          06E7   699                  ACMODE=#PSL$C_USER,-
                          06E7   700                  LOGNAM=W^LOGNAMS                     ; try _S form
                          0708   701          FAIL_CHECK SS$_NORMAL                        ; check success
                  01  DD  0708                        PUSHL   #SS$_NORMAL
     0E20'CF      01  FB  070A                        CALLS   #1,W^REG_CHECK
         1FBB'CF      D4  070F   702          CLRL    W^CRE+CREMBX$_PRMFLG                 ; make it temporary
 1FC3'CF   00000400 8F DO 0713   703          MOVL    #MBBUF,W^CRE+CREMBX$_MAXMSG          ; set MAXMSG parameter
 1FC7'CF   00000400 8F DO 071C   704          MOVL    #BUF_SIZ_G,W^CRE+CREMBX$_BUFQUO      ; set the buffer quota
     1FCF'CF      01  D0  0725   705          MOVL    #PSL$C_EXEC,W^CRE+CREMBX$_ACMODE     ; set the acmode
  1FD3'CF   0154'CF   DE  072A   706          MOVAL   W^LOGNAMG,W^CRE+CREMBX$_LOGNAM       ; set the lognam
                          0731   707          $CREMBX_G W^CRE                              ; try _G form
                          073A   708          FAIL_CHECK SS$_NORMAL                        ; check success
```

```
                01   DD  073A                        PUSHL   #SS$_NORMAL
0E20'CF         01   FB  073C                        CALLS   #1,W^REG_CHECK
0CFC'CF         00   FB  0741   709          CALLS   #0,W^VERIFY_MBX                 ; read,write,verify,delete MBX
0C3D'CF         00   FB  0746   710          CALLS   #0,W^ERLBUF_DUMP                ; dump errors
                         074B   711          .SBTTL  FAIL CREMBX TESTS
                         074B   712 ;+++
                         074B   713 ;
                         074B   714 ; Test access violation for logical name
                         074B   715 ;
                         074B   716 ;---
                         074B   717          NEXT_TEST
                         074B
                         074B       STP9:
2063'CF         09   D0  074B                        MOVL    #9,W^CURRENT_TC
                00   DD  0750                        PUSHL   #0
0E16'CF         01   FB  0752                        CALLS   #1,W^REG_SAVE
                         0757   718          $CREMBX_S CHAN  =MBCHANS,-
                         0757   719                    LOGNAM=ZERO_ADDR_DESCR       ; page zero
                         076E   720          FAIL_CHECK SS$_ACCVIO                   ; check success
                0C   DD  076E                        PUSHL   #SS$_ACCVIO
0E20'CF         01   FB  0770                        CALLS   #1,W^REG_CHECK
                         0775   721 ;+++
                         0775   722 ;
                         0775   723 ; Test access violation for channel number
                         0775   724 ;
                         0775   725 ;---
                         0775   726          NEXT_TEST
                         0775
                         0775       STP10:
2063'CF         0A   D0  0775                        MOVL    #10,W^CURRENT_TC
                00   DD  077A                        PUSHL   #0
0E16'CF         01   FB  077C                        CALLS   #1,W^REG_SAVE
                         0781   727          $CREMBX_S CHAN  =0                      ; channel zero
                         0792   728          FAIL_CHECK SS$_ACCVIO                   ; check success
                0C   DD  0792                        PUSHL   #SS$_ACCVIO
0E20'CF         01   FB  0794                        CALLS   #1,W^REG_CHECK
                         0799   729 ;+++
                         0799   730 ;
                         0799   731 ; Test invalid logical name descriptor
                         0799   732 ;
                         0799   733 ;---
                         0799   734          NEXT_TEST
                         0799
                         0799       STP11:
2063'CF         0B   D0  0799                        MOVL    #11,W^CURRENT_TC
                00   DD  079E                        PUSHL   #0
0E16'CF         01   FB  07A0                        CALLS   #1,W^REG_SAVE
                         07A5   735          $CREMBX_S CHAN  =MBCHANS,-
                         07A5   736                    LOGNAM=LEN_00_DESCR          ; zero length name!
                         07BC   737          FAIL_CHECK SS$_IVLOGNAM                 ; check success
 00000154       8F   DD  07BC                        PUSHL   #SS$_IVLOGNAM
0E20'CF         01   FB  07C2                        CALLS   #1,W^REG_CHECK
                         07C7   738 ;+++
                         07C7   739 ;
                         07C7   740 ; Test valid 255 character logical name descriptor
                         07C7   741 ;                (CREMBX service prefixes 'MBX$' to the string!)
                         07C7   742 ;
```

```
                     07C7   743 ;---
                     07C7   744          NEXT_TEST
                     07C7
                     07C7        STP12:
    2063'CF    0C    D0  07C7                MOVL    #12,W^CURRENT_TC
               00    DD  07CC                PUSHL   #0
    0E16'CF    01    FB  07CE                CALLS   #1,W^REG_SAVE
    000E'CF    01    90  07D3   745          MOVB    #1,CREATED_FLAG                 ; assume it works
                     07D8   746          $CREMBX_S CHAN  =MBCHAND,-
                     07D8   747                    LOGNAM=LEN_63_DESCR             ; 255 chars long
          04 50    E8  07EF   748          BLBS    R0,10$                          ; did it work?
       000E'CF    94  07F2   749          CLRB    CREATED_FLAG                    ; apparently not
                     07F6   750 10$:     FAIL_CHECK SS$_NORMAL                   ; check success
               01    DD  07F6                PUSHL   #SS$_NORMAL
    0E20'CF    01    FB  07F8                CALLS   #1,W^REG_CHECK
                     07FD   751 ;+++
                     07FD   752 ;
                     07FD   753 ; Test DELMBX sys service invalid channel ( 0 )
                     07FD   754 ;
                     07FD   755 ;---
                     07FD   756          NEXT_TEST
                     07FD
                     07FD        STP13:
    2063'CF    0D    D0  07FD                MOVL    #13,W^CURRENT_TC
               00    DD  0802                PUSHL   #0
    0E16'CF    01    FB  0804                CALLS   #1,W^REG_SAVE
2697'CF  0120'CF    DE  0809   757          MOVAL   DELMBX,SERV_NAME                ; set service name
                     0810   758          $DELMBX_S CHAN = CHANNEL_ZERO
                     081C   759          FAIL_CHECK SS$_IVCHAN                   ; check int. failure
     0000013C 8F    DD  081C                PUSHL   #SS$_IVCHAN
    0E20'CF    01    FB  0822                CALLS   #1,W^REG_CHECK
                     0827   760 ;+++
                     0827   761 ;
                     0827   762 ; Test DELMBX sys service invalid channel ( >#channels )
                     0827   763 ;
                     0827   764 ;---
                     0827   765          NEXT_TEST
                     0827
                     0827        STP14:
    2063'CF    0E    D0  0827                MOVL    #14,W^CURRENT_TC
               00    DD  082C                PUSHL   #0
    0E16'CF    01    FB  082E                CALLS   #1,W^REG_SAVE
    7E   2102'CF    B0  0833   766          MOVW    MBCHAND,-(SP)                   ; save the channel #
2102'CF  00000000'9F    B0  0838   767          MOVW    @#CTL$GW_CHINDX,MBCHAND         ; get max # channel
    2102'CF    10    A0  0841   768          ADDW2   #^X10,MBCHAND                   ; one nibble more
                     0846   769          $DELMBX_S CHAN = MBCHAND
                     0852   770          FAIL_CHECK SS$_IVCHAN                   ; check int. failure
     0000013C 8F    DD  0852                PUSHL   #SS$_IVCHAN
    0E20'CF    01    FB  0858                CALLS   #1,W^REG_CHECK
    2102'CF    8E    B0  085D   771          MOVW    (SP)+,MBCHAND                   ; get back the channel
                     0862   772 ;+++
                     0862   773 ;
                     0862   774 ; Test DELMBX sys service - delete one just created
                     0862   775 ;
                     0862   776 ;---
       000E'CF    95  0862   777          TSTB    CREATED_FLAG                    ; not if it wasn't created
             1F    13  0866   778          BEQL    SKIP_DELETE
```

```
                     0868   779          NEXT_TEST
                     0868
                     0868          STP15:
2063'CF    0F   DO   0868                MOVL    #15,W^CURRENT_TC
           00   DD   086D                PUSHL   #0
0E16'CF    01   FB   086F                CALLS   #1,W^REG_SAVE
                     0874   780          $DELMBX_S CHAN = MBCHAND
                     0880   781          FAIL_CHECK SS$_NORMAL                   ; should be O.K.
           01   DD   0880                PUSHL   #SS$_NORMAL
0E20'CF    01   FB   0882                CALLS   #1,W^REG_CHECK
                     0887   782 SKIP_DELETE:
                     0887   783 ;+++
                     0887   784 ;
                     0887   785 ; Test DELMBX sys service - channel not assigned
                     0887   786 ;
                     0887   787 ;---
                     0887   788          NEXT_TEST
                     0887
                     0887          STP16:
2063'CF    10   DO   0887                MOVL    #16,W^CURRENT_TC
           00   DD   088C                PUSHL   #0
0E16'CF    01   FB   088E                CALLS   #1,W^REG_SAVE
                     0893   789          $DASSGN_S CHAN = MBCHAND                 ; deassign the channel
        07 50   E8   089F   790          BLBS    R0,120$                         ; Skip error report if OK
           50   DD   08A2   791          PUSHL   R0                              ; Save error code
14BF'CF    01   FB   08A4   792          CALLS   #1,NONSUB_SSE                   ; Print the failure
                     08A9   793 120$:     $DELMBX_S CHAN = MBCHAND                ; check int. failure
                     08B5   794          FAIL_CHECK SS$_NOPRIV                   ; 'channel not assigned'
           24   DD   08B5                PUSHL   #SS$_NOPRIV
0E20'CF    01   FB   08B7                CALLS   #1,W^REG_CHECK
                     08BC   795 ;+++
                     08BC   796 ;
                     08BC   797 ; Test DELMBX sys service - device not mailbox
                     08BC   798 ;
                     08BC   799 ;---
                     08BC   800          NEXT_TEST
                     08BC
                     08BC          STP17:
2063'CF    11   DO   08BC                MOVL    #17,W^CURRENT_TC
           00   DD   08C1                PUSHL   #0
0E16'CF    01   FB   08C3                CALLS   #1,W^REG_SAVE
                     08C8   801          $ASSIGN_S DEVNAM = SYS_DEV,-            ; system disk assigned
                     08C8   802                    CHAN = MBCHAND
        07 50   E8   08D9   803          BLBS    R0,120$                         ; Skip error report if OK
           50   DD   08DC   804          PUSHL   R0                              ; Save error code
14BF'CF    01   FB   08DE   805          CALLS   #1,NONSUB_SSE                   ; Print the failure
                     08E3   806 120$:     $DELMBX_S CHAN = MBCHAND
                     08EF   807          FAIL_CHECK SS$_DEVNOTMBX                ; check int. failure
 00000074  8F   DD   08EF                PUSHL   #SS$_DEVNOTMBX
0E20'CF    01   FB   08F5                CALLS   #1,W^REG_CHECK
                     08FA   808          $DASSGN_S CHAN = MBCHAND                 ; deassign SYS$SYSDEVICE
                     0906   809 ;+++
                     0906   810 ;
                     0906   811 ; Test no privilege for delete permanent mailbox
                     0906   812 ;
                     0906   813 ;---
                     0906   814          NEXT_TEST
```

```
                         0906
                         0906        STP18:
  2063'CF    12     D0   0906                  MOVL    #18,W^CURRENT_TC
             00     DD   090B                  PUSHL   #0
  0E16'CF    01     FB   090D                  CALLS   #1,W^REG_SAVE
  000E'CF    01     90   0912           MOVB    #1,CREATED_FLAG              ; assume it works
                         0917           $CREMBX_S PRMFLG=#T,-
                         0917                   CHAN  =MBCHAND
       0B 50        E8   092A           BLBS    R0,120$                      ; Skip error report if OK
      000E'CF       94   092D           CLRB    CREATED_FLAG                 ; it didn't work
             50     DD   0931           PUSHL   R0                           ; Save error code
  14BF'CF    01     FB   0933           CALLS   #1,NONSUB_SSE                ; Print the failure
                         0938   120$:
      000E'CF       95   0938           TSTB    CREATED_FLAG                 ; was it created?
             5F     13   093C           BEQL    420$                         ; branch if not
                         093E           $SETPRV_S ENBFLG=#DISABLE,-          ; disable perm privilege
                         093E                   PRVADR=PRM_PRV_MASK
       07 50        E8   094F           BLBS    R0,220$                      ; Skip error report if OK
             50     DD   0952           PUSHL   R0                           ; Save error code
  14BF'CF    01     FB   0954           CALLS   #1,NONSUB_SSE                ; Print the failure
                         0959   220$:
                         0959           $DELMBX_S CHAN=MBCHAND               ; attempt deletion
                         0965           FAIL_CHECK SS$_NOPRIV                ; should fail
             24     DD   0965                  PUSHL   #SS$_NOPRIV
  0E20'CF    01     FB   0967                  CALLS   #1,W^REG_CHECK
                         096C           $SETPRV_S ENBFLG=#ENABLE,-           ; re-enable perm privilege
                         096C                   PRVADR=PRM_PRV_MASK
       07 50        E8   097D           BLBS    R0,320$                      ; Skip error report if OK
             50     DD   0980           PUSHL   R0                           ; Save error code
  14BF'CF    01     FB   0982           CALLS   #1,NONSUB_SSE                ; Print the failure
                         0987   320$:
                         0987           $DELMBX_S CHAN=MBCHAND               ; delete the mailbox
       07 50        E8   0993           BLBS    R0,420$                      ; Skip error report if OK
             50     DD   0996           PUSHL   R0                           ; Save error code
  14BF'CF    01     FB   0998           CALLS   #1,NONSUB_SSE                ; Print the failure
                         099D   420$:
                         099D   ;+++
                         099D   ;
                         099D   ; Test invalid logical name descriptor ( >255 characters )
                         099D   ;               (CREMBX service prefixes 'MBX$' to the string!);
                         099D   ;
                         099D   ;---
                         099D           NEXT_TEST
                         099D
                         099D        STP19:
  2063'CF    13     D0   099D                  MOVL    #19,W^CURRENT_TC
             00     DD   09A2                  PUSHL   #0
  0E16'CF    01     FB   09A4                  CALLS   #1,W^REG_SAVE
2697'CF   0008'CF   DE   09A9           MOVAL   CREMBX,SERV_NAME             ; set service name
                         09B0           $CREMBX_S CHAN  =MBCHANS,-
                         09B0                   LOGNAM=LEN_256_DESCR         ; >255 chars long
                         09C7           FAIL_CHECK SS$_IVLOGNAM              ; check success
   00000154 8F      DD   09C7                  PUSHL   #SS$_IVLOGNAM
  0E20'CF    01     FB   09CD                  CALLS   #1,W^REG_CHECK
                         09D2   ;+++
                         09D2   ;
                         09D2   ; Test logical name translation limit (=10)
```

```
                     09D2    858 ;
                     09D2    859 ;---
                     09D2    860         NEXT_TEST
                     09D2
                     09D2        STP20:
    2063'CF    14    D0   09D2                MOVL    #20,W^CURRENT_TC
               00    DD   09D7                PUSHL   #0
    0E16'CF    01    FB   09D9                CALLS   #1,W^REG_SAVE
               04    BB   09DE    861         PUSHR   #^M<R2>                         ; Save R2's contents
               52    D4   09E0    862         CLRL    R2                              ; Set index variable
  000D'CF    41 8F    90   09E2    863         MOVB    #^A/A/,EQUIV_NAME+4             ; init equivalence name
  0008'CF    42 8F    90   09E8    864         MOVB    #^A/B/,LOGIC_NAME+4             ; init logical name
                     09EE    865 ;
                     09EE    866 ;       Create 9 levels of logical names ( MBX$NAMEB := NAMEA )
                     09EE    867 ;           (10 translations)
                     09EE    868 ;
                     09EE    869 10$:
                     09EE    870         $CRELOG_S LOGNAM = D_MBX_LOGIC_NAME,-
                     09EE    871                   TBLFLG = #2,-
                     09EE    872                   EQLNAM = D_EQUIV_NAME         ; Create a level of logical name
          07 50      E8   0A01    873         BLBS    R0,120$                         ; Skip error report if OK
             50      DD   0A04    874         PUSHL   R0                              ; Save error code
    14BF'CF    01    FB   0A06    875         CALLS   #1,NONSUB_SSE                   ; Print the failure
        0008'CF      96   0A0B    876 120$:   INCB    LOGIC_NAME+4                    ; Bump the logical name by one
        000D'CF      96   0A0F    877         INCB    EQUIV_NAME+4                    ; Bump the equiv name by one
      D7 52    09    F2   0A13    878         AOBLSS  #9,R2,10$                       ; Make 9 levels
               04    BA   0A17    879         POPR    #^M<R2>                         ; Restore R2
        0008'CF      97   0A19    880         DECB    LOGIC_NAME+4                    ; decrement due to loop incr
                     0A1D    881         $CREMBX_S CHAN  =MBCHANS,-
                     0A1D    882                   LOGNAM=D_LOGIC_NAME           ; should be O.K.
                     0A34    883         FAIL_CHECK SS$_NORMAL                       ; check success
               01    DD   0A34                PUSHL   #SS$_NORMAL
    0E20'CF    01    FB   0A36                CALLS   #1,W^REG_CHECK
                     0A3B    884 ;+++
                     0A3B    885 ;
                     0A3B    886 ; Test logical name translation too deep (>10)
                     0A3B    887 ;
                     0A3B    888 ;---
                     0A3B    889         NEXT_TEST
                     0A3B
                     0A3B        STP21:
    2063'CF    15    D0   0A3B                MOVL    #21,W^CURRENT_TC
               00    DD   0A40                PUSHL   #0
    0E16'CF    01    FB   0A42                CALLS   #1,W^REG_SAVE
        0008'CF      96   0A47    890         INCB    LOGIC_NAME+4                    ; Bump the logical name by one
                     0A4B    891         $CRELOG_S LOGNAM = D_MBX_LOGIC_NAME,-
                     0A4B    892                   TBLFLG = #2,-
                     0A4B    893                   EQLNAM = D_EQUIV_NAME         ; Create one more level
          07 50      E8   0A5E    894         BLBS    R0,120$                         ; Skip error report if OK
             50      DD   0A61    895         PUSHL   R0                              ; Save error code
    14BF'CF    01    FB   0A63    896         CALLS   #1,NONSUB_SSE                   ; Print the failure
                     0A68    897 120$:   $CREMBX_S CHAN  =MBCHANS,-
                     0A68    898                   LOGNAM=D_LOGIC_NAME           ; should fail
                     0A7F    899         FAIL_CHECK SS$_TOOMANYLNAM                  ; check success
     00000374 8F     DD   0A7F                PUSHL   #SS$_TOOMANYLNAM
    0E20'CF    01    FB   0A85                CALLS   #1,W^REG_CHECK
                     0A8A    900 ;+++
```

```
                                0A8A    901 ;
                                0A8A    902 ; Test no privilege for permanent mailbox
                                0A8A    903 ;
                                0A8A    904 ;---
                                0A8A    905         NEXT_TEST
                                0A8A
                                0A8A        STP22:
2063'CF    16    D0             0A8A                MOVL    #22,W^CURRENT_TC
           00    DD             0A8F                PUSHL   #0
0E16'CF    01    FB             0A91                CALLS   #1,W^REG_SAVE
                                0A96    906         $SETPRV_S ENBFLG=#DISABLE,-
                                0A96    907                   PRVADR=PRM_PRV_MASK           ; disable perm privilege
      07 50    E8                0AA7    908         BLBS    R0,120$                         ; Skip error report if OK
         50    DD                0AAA    909         PUSHL   R0                              ; Save error code
14BF'CF    01    FB             0AAC    910         CALLS   #1,NONSUB_SSE                   ; Print the failure
                                0AB1    911 120$:   $CREMBX_S PRMFLG=#1,-
                                0AB1    912                   CHAN  =MBCHANS
                                0AC4    913         FAIL_CHECK SS$_NOPRIV                   ; check success
           24    DD             0AC4                PUSHL   #SS$_NOPRIV
0E20'CF    01    FB             0AC6                CALLS   #1,W^REG_CHECK
                                0ACB    914 ;+++
                                0ACB    915 ;
                                0ACB    916 ; Test no privilege for temporary mailbox
                                0ACB    917 ;
                                0ACB    918 ;---
                                0ACB    919         NEXT_TEST
                                0ACB
                                0ACB        STP23:
2063'CF    17    D0             0ACB                MOVL    #23,W^CURRENT_TC
           00    DD             0AD0                PUSHL   #0
0E16'CF    01    FB             0AD2                CALLS   #1,W^REG_SAVE
                                0AD7    920         $SETPRV_S ENBFLG=#DISABLE,-
                                0AD7    921                   PRVADR=TMP_PRV_MASK           ; disable temp privilege
      07 50    E8                0AE8    922         BLBS    R0,120$                         ; Skip error report if OK
         50    DD                0AEB    923         PUSHL   R0                              ; Save error code
14BF'CF    01    FB             0AED    924         CALLS   #1,NONSUB_SSE                   ; Print the failure
                                0AF2    925 120$:   $CREMBX_S CHAN  =MBCHANS
                                0B05    926         FAIL_CHECK SS$_NOPRIV                   ; check success
           24    DD             0B05                PUSHL   #SS$_NOPRIV
0E20'CF    01    FB             0B07                CALLS   #1,W^REG_CHECK
                                0B0C    927         $SETPRV_S ENBFLG=#ENABLE,-              ; re-enable privileges
                                0B0C    928                   PRVADR=TMP_PRM_PRV_MASK
      07 50    E8                0B1D    929         BLBS    R0,220$                         ; Skip error report if OK
         50    DD                0B20    930         PUSHL   R0                              ; Save error code
14BF'CF    01    FB             0B22    931         CALLS   #1,NONSUB_SSE                   ; Print the failure
                                0B27    932 220$:
                                0B27    933
           02    BE             0B27    934         CHMS    #2                              ; reset super mode handler to DCL
                                0B29    935         TEST_END
      269F'CF    DD             0B29                PUSHL   W^TMD_ADDR
      26A3'CF    DD             0B2D                PUSHL   W^TMN_ADDR
           02    DD             0B31                PUSHL   #2
      2594'CF    DD             0B33                PUSHL   W^MOD_MSG_CODE
00000000'GF    04    FB         0B37                CALLS   #$$T1,G^LIB$SIGNAL
2594'CF    01    1C    01       F0  0B3E            INSV    #1,#SFS$V_INHIB_MSG,#1,W^MOD_MSG_CODE
      2594'CF    DD             0B45                PUSHL   W^MOD_MSG_CODE
00000000'GF    01    FB         0B49                CALLS   #1,G^SYS$EXIT
```

```
0B50  .SBTTL  ROUTINES
0B50  .SBTTL SETUP_SUPER ROUTINE
0B50  ;++
0B50  ;
0B50  ;       Routine to declare an initial CHMS handler from user mode.
0B50  ;
0B50  ; FUNCTIONAL DESCRIPTION:
0B50  ;
0B50  ; CALLING SEQUENCE:
0B50  ;
0B50  ;       $CMKRNL_S W^SETUP_SUPER,ARGLST
0B50  ;
0B50  ;               ARGLST = address of a pointer to a one parameter argument list conta
0B50  ;                        the address of the entry mask of the CHMS handler
0B50  ;
0B50  ; INPUT PARAMETERS:
0B50  ;
0B50  ;       ARGLST
0B50  ;
0B50  ; IMPLICIT INPUTS
0B50  ;
0B50  ;       NONE
0B50  ;
0B50  ; OUTPUT PARAMETERS:
0B50  ;
0B50  ;       Declares a change mode handler for super mode which must be
0B50  ;       reset to DCL in the users handler routine when the handler is
0B50  ;       no longer needed.
0B50  ;
0B50  ; IMPLICIT OUTPUTS:
0B50  ;
0B50  ;       NONE
0B50  ;
0B50  ; COMPLETION CODES:
0B50  ;
0B50  ;       NONE
0B50  ;
0B50  ; SIDE EFFECTS:
0B50  ;
0B50  ;       NONE
0B50  ;
0B50  ; ON ENTRY:
0B50  ;                        ---------          ---------
0B50  ;               KSP => !    0    !    USP => !         !
0B50  ;                      !    0    !          !  USER   !
0B50  ;                      !   AP    !          !         !
0B50  ;                      !   FP    !          !  CALL   !
0B50  ;                      !   PC    !          !         !
0B50  ;                      !    0    !          !  FRAME  !
0B50  ;                      !    0    !          !         !
0B50  ;                      !   AP    !          ---------
0B50  ;                      !   FP    !
0B50  ;                      !SRVEXIT  !
0B50  ;                      !   PC    !
0B50  ;                      !  PSL    !
0B50  ;                      ---------
0B50  ;--
```

```
                     0B50   995 RETURN_PC:
            00000000 0B50   996         .LONG   0                       ; storage for user return PC
                     0B54   997 HANDLER_PC:
            00000000 0B54   998         .LONG   0                       ; storage for handler PC
                     0B58   999 ;
                     0B58  1000 SETUP_SUPER:
                000C 0B58  1001         .WORD   ^M<R2,R3>
      53    03   DB 0B5A  1002         MFPR    #PR$_USP,R3             ; get the user call frame address
EE AF   10 A3   D0 0B5D  1003         MOVL    SF$L_SAVE_PC(R3),B^RETURN_PC ; get the user return PC
ED AF   04 AC   D0 0B62  1004         MOVL    4(AP),HANDLER_PC        ; save the handler address
   52   0C AD   D0 0B67  1005         MOVL    SF$L_SAVE_FP(FP),R2     ; get saved FP
   52   00'  C0 0B6B  1006         ADDL    S^#EXE$C_CMSTKSZ,R2     ; back over change mode stack frame
   62  7C'AF   9E 0B6E  1007         MOVAB   B^20$,(R2)              ; set return address
            0A   F0 0B72  1008         INSV    #<<PSL$C_SUPER@PSL$S_CURMOD>+PSL$C_SUPER>,-
            16      0B74  1009                 #PSL$V_PRVMOD,-
   04 A2    04      0B75  1010                 #PSL$S_CURMOD*2,4(R2)   ; set current and previous mode to super
      50    01   D0 0B78  1011         MOVL    S^#SS$_NORMAL,R0        ; set correct return code
            04   0B7B  1012         RET                             ; enter super mode
                     0B7C  1013 20$:
            7E   D4 0B7C  1014         CLRL    -(SP)                   ; set up dummy PSL
   82'AF    6E   FA 0B7E  1015         CALLG   (SP),B^30$              ; create initial call frame
                     0B82  1016 30$:
                0000 0B82  1017         .WORD   ^M<>                    ; entry mask
            00   DD 0B84  1018         PUSHL   #0                      ; push a dummy parameter
 0E16'CF    01   FB 0B86  1019         CALLS   #1,W^REG_SAVE           ; save the registers
                     0B8B  1020         $DCLCMH_S @HANDLER_PC,W^PRVHND1,#0 ; set real handler
                     0B9B  1021         FAIL_CHECKNP SS$_NORMAL         ; check for success
            01   DD 0B9B                         PUSHL   #SS$_NORMAL
 1443'CF    01   FB 0B9D                         CALLS   #1,W^REG_CHECKNP
  03C00000  8F   DD 0BA2  1022         PUSHL   #<<PSL$C_USER@PSL$V_CURMOD>-
                     0BA8  1023                 !<PSL$C_USER@PSL$V_PRVMOD>>; set return to user
         A5 AF   DD 0BA8  1024         PUSHL   RETURN_PC               ; set the return PC
            02   0BAB  1025         REI                             ; return
```

```
                        0BAC  1027 .SBTTL SUPER_MODE
                        0BAC  1028 ;++
                        0BAC  1029 ; FUNCTIONAL DESCRIPTION:
                        0BAC  1030 ;       Routine to handle the CHMS instructions.
                        0BAC  1031 ;
                        0BAC  1032 ; CALLING SEQUENCE:
                        0BAC  1033 ;       CHMS    #N
                        0BAC  1034 ;
                        0BAC  1035 ; INPUT PARAMETERS:
                        0BAC  1036 ;         SP=>  CHMS parameter
                        0BAC  1037 ;               PC
                        0BAC  1038 ;               PSL
                        0BAC  1039 ;
                        0BAC  1040 ;         The CHMS parameter can be one of the following:
                        0BAC  1041 ;
                        0BAC  1042 ;               1 = execute $CREMBX (_S and _G form)
                        0BAC  1043 ;               2 = execute a $DCLCMH_S to reset the CHMS handler to DCL
                        0BAC  1044 ;
                        0BAC  1045 ; OUTPUT PARAMETERS:
                        0BAC  1046 ;       NONE
                        0BAC  1047 ;--
                        0BAC  1048
                        0BAC  1049 SUPER_MODE:
     50    8E    D0     0BAC  1050         MOVL    (SP)+,R0                        ; get CHM parameter off the stack
02   01    50    8F     0BAF  1051         CASEB   R0,#1,#2                        ; do the right thing
                        0BB3  1052 10$:
              0004'     0BB3  1053         .WORD   20$-10$
              006C'     0BB5  1054         .WORD   A30-10$
                        0BB7  1055 20$:
          00    DD      0BB7  1056         PUSHL   #0
      0E16'CF   01    FB  0BB9  1057         CALLS   #1,W^REG_SAVE                   ; save the registers
                        0BBE  1058         $CREMBX_S PRMFLG=#0,-
                        0BBE  1059                   CHAN  =W^MBCHANS,-
                        0BBE  1060                   MAXMSG=#MBBUF,-
                        0BBE  1061                   BUFQUO=#BUF_SIZ_S,-
                        0BBE  1062                   ACMODE=#PSL$C_SUPER,-
                        0BBE  1063                   LOGNAM=W^LOGNAMS           ; try _s form
                        0BDF  1064         FAIL_CHECKNP SS$_NORMAL              ; check success
          01    DD      0BDF                       PUSHL   #SS$_NORMAL
      1443'CF   01    FB  0BE1                       CALLS   #1,W^REG_CHECKNP
          1FBB'CF   D4    0BE6  1065         CLRL    W^CRE+CREMBX$_PRMFLG         ; make it temporary
1FC3'CF   00000400 8F   D0  0BEA  1066         MOVL    #MBBUF,W^CRE+CREMBX$_MAXMSG  ; set max. message size
1FC7'CF   00000400 8F   D0  0BF3  1067         MOVL    #BUF_SIZ_G,W^CRE+CREMBX$_BUFQUO ; set the buffer quota
      1FCF'CF   02    D0  0BFC  1068         MOVL    #PSL$C_SUPER,W^CRE+CREMBX$_ACMODE ; set the access mode
  1FD3'CF   0154'CF   DE  0C01  1069         MOVAL   W^LOGNAMG,W^CRE+CREMBX$_LOGNAM ; set the lognam
                        0C08  1070         $CREMBX_G W^CRE                      ; try _g form
                        0C11  1071         FAIL_CHECKNP SS$_NORMAL              ; check success
          01    DD      0C11                       PUSHL   #SS$_NORMAL
      1443'CF   01    FB  0C13                       CALLS   #1,W^REG_CHECKNP
      0CFC'CF   00    FB  0C18  1072         CALLS   #0,W^VERIFY_MBX              ; read,write,verify,delete MBX
                1D    11  0C1D  1073         BRB     A50                          ; get back to user mode
                        0C1F  1074 A30:
  2697'CF   0119'CF   DE  0C1F  1075         MOVAL   W^DCLCMH,W^SERV_NAME         ; set service name pointer
                        0C26  1076         $DCLCMH_S @PRVHND1,,#0               ; reset the CHMS handler for DCL
                        0C35  1077         FAIL_CHECK SS$_NORMAL                ; check for success
          01    DD      0C35                       PUSHL   #SS$_NORMAL
      0E20'CF   01    FB  0C37                       CALLS   #1,W^REG_CHECK
```

```
                0C3C  1078 A50:
 02             0C3C  1079          REI                             ; return to user mode
```

```
                                                B 13
                     0C3D  1081          .SBTTL  ERLBUF_DUMP
                     0C3D  1082 ;++
                     0C3D  1083 ; FUNCTIONAL DESCRIPTION:
                     0C3D  1084 ;       Routine to check for errors in the error log buffer and
                     0C3D  1085 ;       report any that are there.
                     0C3D  1086 ;
                     0C3D  1087 ; CALLING SEQUENCE:
                     0C3D  1088 ;       CALLS #0,W^ERLBUF_DUMP
                     0C3D  1089 ;
                     0C3D  1090 ; INPUT PARAMETERS:
                     0C3D  1091 ;       FLAG bit 0 = 0 for no errors logged
                     0C3D  1092 ;       FLAG bit 0 = 1 for errors logged
                     0C3D  1093 ;       if errors logged then buffer ERLB must contain legal format errors
                     0C3D  1094 ;
                     0C3D  1095 ; OUTPUT PARAMETERS:
                     0C3D  1096 ;       NONE
                     0C3D  1097 ;
                     0C3D  1098 ;--
                     0C3D  1099
                     0C3D  1100 ERLBUF_DUMP:
             001C    0C3D  1101          .WORD   ^M<R2,R3,R4>
  2A 0E62'CF   E9    0C3F  1102          BLBC    FLAG,30$            ; br if no errors to report
52   0E67'CF   DE    0C44  1103          MOVAL   ERLB,R2             ; set up buffer pointer
                     0C49  1104 10$:
          62   D5    0C49  1105          TSTL    (R2)                ; any more errors?
          21   13    0C4B  1106          BEQL    30$                 ; br if not
2697'CF   82   D0    0C4D  1107          MOVL    (R2)+,W^SERV_NAME   ; reset service name
2063'CF   82   D0    0C52  1108          MOVL    (R2)+,W^CURRENT_TC  ; reset step #
2598'CF   82   D0    0C57  1109          MOVL    (R2)+,W^MODE        ; reset the mode
     53   82   9A    0C5C  1110          MOVZBL  (R2)+,R3            ; get the longword count
     54   53   D0    0C5F  1111          MOVL    R3,R4               ; and save it
                     0C62  1112 20$:
          82   DD    0C62  1113          PUSHL   (R2)+               ; push a parameter
        FB 53   F5    0C64  1114          SOBGTR  R3,20$              ; and push them all
    1501'CF   54   FB    0C67  1115          CALLS   R4,W^PRINT_FAIL     ; print the failure
          DB   11    0C6C  1116          BRB     10$                 ; do the next one
                     0C6E  1117 30$:
0E63'CF   0E67'CF   DE    0C6E  1118          MOVAL   W^ERLB,W^ELBP       ; reset the buffer pointer
          0E67'CF   D4    0C75  1119          CLRL    W^ERLB              ; set fresh terminater
                    04    0C79  1120          RET                         ; bail out
```

```
                                0C7A  1122          .SBTTL BUF_CHECK
                                0C7A  1123 ;++
                                0C7A  1124 ; FUNCTIONAL DESCRIPTION:
                                0C7A  1125 ;       Routine to check the contents of a buffer against known good
                                0C7A  1126 ;       data.
                                0C7A  1127 ;
                                0C7A  1128 ; CALLING SEQUENCE:
                                0C7A  1129 ;       CALLS #0,W^BUF_CHECK            ; check buffer
                                0C7A  1130 ;
                                0C7A  1131 ; INPUT PARAMETERS:
                                0C7A  1132 ;       R6 = buffer address
                                0C7A  1133 ;       R7 = good data address
                                0C7A  1134 ;       R8 = byte count
                                0C7A  1135 ;
                                0C7A  1136 ; OUTPUT PARAMETERS:
                                0C7A  1137 ;       NONE
                                0C7A  1138 ;
                                0C7A  1139 ;--
                                0C7A  1140
                                0C7A  1141 BUF_CHECK:
                          03FC  0C7A  1142          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9>
                59    56    D0  0C7C  1143          MOVL    R6,R9                   ; save a copy of the buffer address
             66 67    58    29  0C7F  1144          CMPC3   R8,(R7),(R6)            ; check the buffer
                      76    13  0C83  1145          BEQL    10$                     ; br if good
       0017'CF    53    59    C3  0C85  1146          SUBL3   R9,R3,W^ARGLST1+8       ; get buffer offset
            001B'CF    61    9A  0C8B  1147          MOVZBL  (R1),W^ARGLST1+12       ; get the good data
                                0C90  1148          $GETCHN_S CHAN  =W^MBCHAN,-
                                0C90  1149                    PRIBUF=W^PB           ; get the unit number
       0013'CF  25C8'CF    3C  0CA6  1150          MOVZWL  W^PB+DIB$W_UNIT+8,W^ARGLST1+4   ; get the unit number
            001F'CF    63    9A  0CAD  1151          MOVZBL  (R3),W^ARGLST1+16       ; get the bad data
                                0CB2  1152          $GETMSG_S MSGID=#UETP$_DATAER,-
                                0CB2  1153                    MSGLEN=W^ML,-
                                0CB2  1154                    BUFADR=W^CTRSTR,-
                                0CB2  1155                    FLAGS =#1             ; get the ctrstr
                                0CCB  1156          $FAOL_S W^CTRSTR,W^ML,W^GETBUF,W^ARGLST1 ; make it readable
       0DF5'CF    00    FB  0CE2  1157          CALLS   #0,W^STORE_STEP         ; store error info.
52     0E63'CF    D0  0CE7  1158          MOVL    W^ELBP,R2               ; get error log buffer pointer
            82    01    90  0CEC  1159          MOVB    #1,(R2)+                ; save longword count
82     258C'CF    DE  0CEF  1160          MOVAL   W^ML,(R2)+              ; push desc. address
                      62    D4  0CF4  1161          CLRL    (R2)                    ; make a new terminator
0E63'CF    52    D0  0CF6  1162          MOVL    R2,W^ELBP               ; reset buffer pointer
                                0CFB  1163 10$:
                          04  0CFB  1164          RET                             ; return
```

```
                                 0CFC  1166 .SBTTL  VERIFY_MBX
                                 0CFC  1167 ;++
                                 0CFC  1168 ; FUNCTIONAL DESCRIPTION:
                                 0CFC  1169 ;       Routine to write to a mailbox, read the data back, verify the
                                 0CFC  1170 ;       results and delete the mailbox.
                                 0CFC  1171 ;
                                 0CFC  1172 ; CALLING SEQUENCE:
                                 0CFC  1173 ;       CALLS   #1,W^VERIFY_MBX
                                 0CFC  1174 ;
                                 0CFC  1175 ; INPUT PARAMTERS
                                 0CFC  1176 ;
                                 0CFC  1177 ; OUTPUT PARAMETERS:
                                 0CFC  1178 ;       NONE
                                 0CFC  1179 ;
                                 0CFC  1180 ;--
                                 0CFC  1181 VERIFY_MBX:
                          03FC   0CFC  1182         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9>     ; entry mask
                                 0CFE  1183         $QIO_S  CHAN=W^MBCHANS,-
                                 0CFE  1184                 FUNC=#IO$_WRITEVBLK,-
                                 0CFE  1185                 P1  =W^TEST_DATA,-
                                 0CFE  1186                 P2  =#BUF_SIZ_S                 ; write to the mailbox_S
0100 8F    00    2180'CF    00  2C 0D21  1187         MOVC5   #0,W^MBX_BUF,#0,#256,W^MBX_BUF  ; zero the MBX buffer
                       2180'CF
                                 0D2A
                                 0D2D  1188         $QIO_S  CHAN=W^MBCHANS,-
                                 0D2D  1189                 FUNC=#IO$_READVBLK,-
                                 0D2D  1190                 P1  =W^MBX_BUF,-
                                 0D2D  1191                 P2  =#BUF_SIZ_S                 ; read from the mailbox
           56    2180'CF    DE  0D50  1192         MOVAL   W^MBX_BUF,R6                    ; set the MBX buffer
           57    0365'CF    DE  0D55  1193         MOVAL   W^TEST_DATA,R7                  ; find the master data
      58    00000100 8F    D0  0D5A  1194         MOVL    #BUF_SIZ_S,R8
     20FC'CF    2100'CF    B0  0D61  1195         MOVW    W^MBCHANS,W^MBCHAN              ; get the channel number
           FF0D CF    00  FB  0D68  1196         CALLS   #0,W^BUF_CHECK                  ; check the data
                                 0D6D  1197         $DASSGN_S CHAN=W^MBCHANS                ; deassign the channel
                                 0D79  1198         $QIO_S  CHAN=W^MBCHANG,-
                                 0D79  1199                 FUNC=#IO$_WRITEVBLK,-
                                 0D79  1200                 P1  =W^TEST_DATA,-
                                 0D79  1201                 P2  =#BUF_SIZ_G                 ; write to the mailbox_G
0400 8F    00    2180'CF    00  2C 0D9C  1202         MOVC5   #0,W^MBX_BUF,#0,#1024,W^MBX_BUF ; zero the MBX buffer
                       2180'CF
                                 0DA5
                                 0DA8  1203         $QIO_S  CHAN=W^M  HANG,-
                                 0DA8  1204                 FUNC=#IO$_READVBLK,-
                                 0DA8  1205                 P1  =W^MBX_BUF,-
                                 0DA8  1206                 P2  =#BUF_SIZ_G                 ; read from the mailbox
           56    2180'CF    DE  0DCB  1207         MOVAL   W^MBX_BUF,R6                    ; set the MBX buffer
           57    0365'CF    DE  0DD0  1208         MOVAL   W^TEST_DATA,R7                  ; find the master data
      58    00000400 8F    D0  0DD5  1209         MOVL    #BUF_SIZ_G,R8
     20FC'CF    20FE'CF    B0  0DDC  1210         MOVW    W^MBCHANG,W^MBCHAN              ; get the channel number
           FE92 CF    00  FB  0DE3  1211         CALLS   #0,W^BUF_CHECK                  ; check the data
                                 0DE8  1212         $DASSGN_S CHAN=W^MBCHANG                ; deassign the channel
                                 04    0DF4  1213         RET
```

```
                        0DF5  1215          .SBTTL STORE_STEP
                        0DF5  1216 ;++
                        0DF5  1217 ; FUNCTIONAL DESCRIPTION:
                        0DF5  1218 ;        Routine to store step information in the error log buffer.
                        0DF5  1219 ;
                        0DF5  1220 ; CALLING SEQUENCE:
                        0DF5  1221 ;        CALLS #0,W^STORE_STEP
                        0DF5  1222 ;
                        0DF5  1223 ; INPUT PARAMETERS:
                        0DF5  1224 ;        ELBP = current errlog buffer pointer
                        0DF5  1225 ;
                        0DF5  1226 ; OUTPUT PARAMETERS:
                        0DF5  1227 ;        FLAG = error logged flag
                        0DF5  1228 ;
                        0DF5  1229 ;--
                        0DF5  1230
                        0DF5  1231 STORE_STEP:
                  0004  0DF5  1232          .WORD   ^M<R2>
0E62'CF    01     88    0DF7  1233          BISB2   #1,W^FLAG                       ; set the error logged flag
52    0E63'CF     D0    0DFC  1234          MOVL    W^ELBP,R2                       ; get errlog buf pntr
82    2697'CF     D0    0E01  1235          MOVL    W^SERV_NAME,(R2)+               ; save the service name
82    2063'CF     D0    0E06  1236          MOVL    W^CURRENT_TC,(R2)+              ; save the step number
82    2598'CF     D0    0E0B  1237          MOVL    W^MODE,(R2)+                    ; save the mode
0E63'CF    52     D0    0E10  1238          MOVL    R2,W^ELBP                       ; reset the errlog buf pntr
                   04    0E15  1239          RET                                     ; return
```

```
                                                F 13
                        0E16  1241          .SBTTL REG_SAVE
                        0E16  1242 ;++
                        0E16  1243 ; FUNCTIONAL DESCRIPTION:
                        0E16  1244 ;       Subroutine to save R2-R11 in the register save location.
                        0E16  1245 ;
                        0E16  1246 ; CALLING SEQUENCE:
                        0E16  1247 ;       PUSHL   #0                  ; save a dummy parameter
                        0E16  1248 ;       CALLS   #1,W^REG_SAVE       ; save R2-R11
                        0E16  1249 ;
                        0E16  1250 ; INPUT PARAMETERS:
                        0E16  1251 ;       NONE
                        0E16  1252 ;
                        0E16  1253 ; OUTPUT PARAMETERS:
                        0E16  1254 ;       NONE
                        0E16  1255 ;
                        0E16  1256 ;--
                        0E16  1257
                        0E16  1258 REG_SAVE:
                  0FFC  0E16  1259          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
      264F'CF   14 AD 28  0E18  1260          MOVC3   #4*10,^X14(FP),W^REG_SAVE_AREA  ; save the registers in the program
                    04  0E1F  1261          RET
```

```
                                 0E20          .SBTTL  REG_CHECK
                                 0E20   ;++
                                 0E20   ; FUNCTIONAL DESCRIPTION:
                                 0E20   ;       Subroutine to test R0 & R2-R11 for proper content after a service
                                 0E20   ;       execution. A snapshot is taken by the REG_SAVE routine at the
                                 0E20   ;       beginning of each step and this routine is executed after the
                                 0E20   ;       services have been executed.
                                 0E20   ;
                                 0E20   ; CALLING SEQUENCE:
                                 0E20   ;       PUSHL   #SS$_XXXXXX     ; push expected R0 contents
                                 0E20   ;       CALLS   #1,W^REG_CHECK  ; execute this routine
                                 0E20   ;
                                 0E20   ; INPUT PARAMETERS:
                                 0E20   ;       expected R0 contents on the stack
                                 0E20   ;
                                 0E20   ; OUTPUT PARAMETERS:
                                 0E20   ;       possible error messages printed using $PUTMSG
                                 0E20   ;
                                 0E20   ;--
                                 0E20
                                 0E20   REG_CHECK:
                          0FFC   0E20           .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
          50    04 AC      D1    0E22           CMPL    4(AP),R0                            ; is this the right fail code?
                   0E      13    0E26           BEQL    10$                                 ; br if yes
                   50      DD    0E28           PUSHL   R0                                  ; push received data
                04 AC      DD    0E2A           PUSHL   4(AP)                               ; push expected data
              0138'CF      DF    0E2D           PUSHAL  W^EXP                               ; push the string variable
        1501'CF    03      FB    0E31           CALLS   #3,W^PRINT_FAIL                     ; print the error message
                                 0E36   10$:
  264F'CF    14 AD    28   29    0E36           CMPC3   #4*10,^X14(FP),W^REG_SAVE_AREA      ; check all but R0
                      22   13    0E3D           BEQL    20$                                 ; br if O.K.
  56    53   0000264F'8F   C3    0E3F           SUBL3   #REG_SAVE_AREA,R3,R6                ; calculate the register number
                56    04   C6    0E47           DIVL2   #4,R6
          7E    56    02   81    0E4A           ADDB3   #^X2,R6,-(SP)                       ; set number past R0-R1 and save
                51    03   CA    0E4E           BICL2   #3,R1                               ; backup to register boundrys
                53    03   CA    0E51           BICL2   #3,R3
                      61   DD    0E54           PUSHL   (R1)                                ; push received data
                      63   DD    0E56           PUSHL   (R3)                                ; push expected data
              263D'CF      DF    0E58           PUSHAL  W^REG                               ; set string pntr param.
        1501'CF    04      FB    0E5C           CALLS   #4,W^PRINT_FAIL                     ; print the error message
                                 0E61   20$:
                           04    0E61           RET
```

```
                                                H 13

                                        .SBTTL REG_CHECKNP
                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;       Subroutine to test R0 & R2-R11 for proper content after a service
                                ;       execution without printing it. A snapshot is taken by the REG_SAVE routine a
                                ;       beginning of each step and this routine is executed after the
                                ;       services have been executed. This routine collects the error
                                ;       information in buffer ERLB instead of printing it.
                                ;
                                ; CALLING SEQUENCE:
                                ;       PUSHL   #SS$_XXXXXX     ; push expected R0 contents
                                ;       CALLS   #1,W^REG_CHECK  ; execute this routine
                                ;
                                ; INPUT PARAMETERS:
                                ;       expected R0 contents on the stack
                                ;
                                ; OUTPUT PARAMETERS:
                                ;       possible error messages logged in buffer ERLB which are printed
                                ;       using routine ERLBUF_DUMP.
                                ;
                                ;       Error packets are in the following form:
                                ;
                                ;               !-----------------!
                                ;               !Service name pntr!
                                ;               !-----------------!
                                ;               !     Step #      !
                                ;               !-----------------!
                                ;               !Mode name pointer!
                                ;               !-----------------!
                                ;               !               ! ! long word count
                                ;               !-----------------!
                                ;               !\/\/\/\/\/\/\/\/\! 3-4 parameter long words
                                ;
                                ;--

                                FLAG:
                        00      .BYTE 0                         ; error flags are BIT0 = 0 means no errors in the bu
                                ;                                       BIT0 = 1 means errors in the buffe
                                ELBP:
                00000E67'       .ADDRESS ERLB                   ; error log buffer pointer
                                ERLB:
                00001443        .BLKB   1500                    ; error log buffer
                                ;
                                REG_CHECKNP:
                        0FFC    .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
          50    04 AC   D1      CMPL    4(AP),R0                ; is this the right fail code
                        2E      13      BEQL    10$             ; br if yes
        F9A5 CF         00      FB      CALLS   #0,W^STORE_STEP ; store step information
        52      FA0F CF         D0      MOVL    ELBP,R2         ; get the current error log pointer
                82      03      90      MOVB    #3,(R2)+        ; save the long word count
                82      50      D0      MOVL    R0,(R2)+        ; save received status
          82    04 AC           D0      MOVL    4(AP),(R2)+     ; save expected status
        82      0138'CF         DE      MOVAL   W^EXP,(R2)+     ; save the string variable
                        62      D4      CLRL    (R2)            ; set the terminator
        F9F8 CF         52      D0      MOVL    R2,ELBP         ; reset the buffer pointer
    269F'CF     076B'CF         DE      MOVAL   W^TEST_MOD_FAIL,W^TMD_ADDR ; set failure message address
2594'CF 03      00      02      F0      INSV    #ERROR,#0,#3,W^MOD_MSG_CODE ; set severity code
```

```
                              1479  1363 10$:
 264F'CF    14 AD   28    29  1479  1364         CMPC3   #4*10,^X14(FP),W^REG_SAVE_AREA ; check all but R0 and R1
                    3C    13  1480  1365         BEQL    20$                   ; br if OK
            F96E CF 00    FB  1482  1366         CALLS #0,W^STORE_STEP         ; store step information
            52  F9D8 CF   D0  1487  1367         MOVL    ELBP,R2               ; get current error log buf pointer
                82  04    90  148C  1368         MOVB    S^#4,(R2)+            ; set longword count
             0000264F'8F  C3  148F  1369         SUBL3   #REG_SAVE_AREA,-
                56  53        1495  1370         R3,R6                         ; calc reg number
                56  04    C6  1497  1371         DIVL2   S^#4,R6               ; make it a longword count
            82  56  02    C1  149A  1372         ADDL3   S^#2,R6,(R2)+         ; correct for R0-R1 and save
                82  61    D0  149E  1373         MOVL    (R1),(R2)+            ; save received results
                82  63    D0  14A1  1374         MOVL    (R3),(R2)+            ; save expected results
            82  263D'CF   DE  14A4  1375         MOVAL   W^REG,(R2)+           ; save string variable
                    62    D4  14A9  1376         CLRL    (R2)                  ; set the terminator
            F9B3 CF 52    D0  14AB  1377         MOVL    R2,ELBP               ; reset the buffer pointer
       269F'CF  076B'CF   DE  14B0  1378         MOVAL   W^TEST_MOD_FAIL,W^TMD_ADDR      ; set failure message address
  2594'CF   03  00  02    F0  14B7  1379         INSV    #ERROR,#0,#3,W^MOD_MSG_CODE     ; set severity code
                              14BE  1380 20$:
                          04  14BE  1381         RET                           ; bail out
```

```
                                14BF  1383          .SBTTL NONSUB_SSE
                                14BF  1384 ;++
                                14BF  1385 ; FUNCTIONAL DESCRIPTION:
                                14BF  1386 ;       Subroutine to report the failure of a system service which is not the
                                14BF  1387 ;       subject system service.
                                14BF  1388 ;
                                14BF  1389 ; CALLING SEQUENCE:
                                14BF  1390 ;       PUSHL   R0                ; Save the failure status
                                14BF  1391 ;       CALLS   #1,NONSUB_SSE     ; Print the failure message
                                14BF  1392 ;
                                14BF  1393 ; INPUT PARAMETERS:
                                14BF  1394 ;       4(AP) = Status code of failing system service
                                14BF  1395 ;
                                14BF  1396 ; OUTPUT PARAMETERS:
                                14BF  1397 ;       NONE
                                14BF  1398 ;
                                14BF  1399 ;--
                                14BF  1400 NONSUB_SSE:
                       0FFC     14BF  1401          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                14C1  1402          $GETMSG_S MSGID  = 4(AP),-
                                14C1  1403                    MSGLEN = BUFFER_PTR,-
                                14C1  1404                    BUFADR = BUFFER_PTR,-
                                14C1  1405                    FLAGS  = #1                     ; Get just the text of the message
                                14D7  1406          $FAO_S  CTRSTR = NSSSF,-
                                14D7  1407                  OUTLEN = MESSAGE1L,-
                                14D7  1408                  OUTBUF = MSG1L,-
                                14D7  1409                  P1     = #BUFFER_PTR            ; Format the failure message
          27BB'CF    0100 8F    B0  14F0  1410          MOVW    #TEXT_BUFFER,BUFFER_PTR     ; Reset the descriptor length
                    27B3'CF     DF  14F7  1411          PUSHAL  MESSAGE1L                   ; Push the string address
               1501'CF    01    FB  14FB  1412          CALLS   #1,PRINT_FAIL               ; Print the failure
                             04   1500  1413          RET
```

```
                                1501  1415          .SBTTL  PRINT_FAIL
                                1501  1416 ;++
                                1501  1417 ; FUNCTIONAL DESCRIPTION:
                                1501  1418 ;       Subroutine to report failures using $PUTMSG
                                1501  1419 ;
                                1501  1420 ; CALLING SEQUENCE:
                                1501  1421 ; Mode  #1       PUSHL EXPECTED  Mode    #2      PUSHL REG_NUMBER
                                1501  1422 ;                PUSHL RECEIVED                   PUSHL EXPECTED
                                1501  1423 ;                PUSHAL STRING_VAR                PUSHL RECEIVED
                                1501  1424 ;                CALLS #3,W^PRINT_FAIL            PUSHAL STRING_VAR
                                1501  1425 ;                                                 CALLS #4,W^PRINT_FAIL
                                1501  1426 ; Mode  #3       PUSHAL STRING_VAR
                                1501  1427 ;                CALLS #1,W^PRINT_FAIL
                                1501  1428 ;
                                1501  1429 ; INPUT PARAMETERS:
                                1501  1430 ;       listed above
                                1501  1431 ;
                                1501  1432 ; OUTPUT PARAMETERS:
                                1501  1433 ;       an error message is printed using $PUTMSG
                                1501  1434 ;
                                1501  1435 ;--
                                1501  1436
                                1501  1437 PRINT_FAIL:
                         003C   1501  1438          .WORD   ^M<R2,R3,R4,R5>
                                1503  1439          $FAO_S  W^CS1,W^MESSAGEL,W^MSGL,#TEST_MOD_NAME,W^SERV_NAME,W^CURRENT_TC
                                1524  1440          $PUTMSG_S W^MSGVEC                       ; print the message
                   04   6C   91  1535  1441          CMPB    (AP),#4                  ; is this a register message?
                             26  1538  1442          BEQL    10$                      ; br if yes
                   01   6C   91  153A  1443          CMPB    (AP),#1                  ; is this just a message?
                             48  153D  1444          BEQL    20$                      ; br if yes
                                 153F  1445          $FAO_S  W^CS2,W^MESSAGEL,W^MSGL,4(AP),8(AP),4(AP),12(AP)
                             40  155E  1446          BRB     30$                      ; goto output message
                                 1560  1447 10$:
                                 1560  1448          $FAO_S  W^CS3,W^MESSAGEL,W^MSGL,4(AP),16(AP),8(AP),4(AP),16(AP),12(AP)
                             19  1585  1449          BRB     30$                      ; goto output message
                                 1587  1450 20$:
          25B0'CF    04   AC   D0  1587  1451          MOVL    4(AP),W^MSGVEC1+12       ; save string address
                                 158D  1452          $PUTMSG_S W^MSGVEC1              ; print the message
                             11  159E  1453          BRB     40$                      ; skip the other message
                                 15A0  1454 30$:
                                 15A0  1455          $PUTMSG_S W^MSGVEC               ; print the message
                                 15B1  1456 40$:
            15C5'CF    00   FB  15B1  1457          CALLS   #0,W^MODE_ID             ; identify the mode
       269F'CF    076B'CF   DE  15B6  1458          MOVAL   W^TEST_MOD_FAIL,W^TMD_ADDR       ; set failure message address
  2594'CF    03   00   02   F0  15BD  1459          INSV    #ERROR,#0,#3,W^MOD_MSG_CODE      ; set severity code
                         04  15C4  1460          RET
```

```
                15C5  1462          .SBTTL  MODE_ID
                15C5  1463 ;++
                15C5  1464 ; FUNCTIONAL DESCRIPTION:
                15C5  1465 ;        Subroutine to identify the mode that an exit handler is in.
                15C5  1466 ;
                15C5  1467 ; CALLING SEQUENCE:
                15C5  1468 ;        CALLS   #0,W^MODE_ID
                15C5  1469 ;
                15C5  1470 ; INPUT PARAMETERS:
                15C5  1471 ;        MODE contains an address pointing to an ascii string desc.
                15C5  1472 ;        of the current CPU mode.
                15C5  1473 ;
                15C5  1474 ; OUTPUT PARAMETERS:
                15C5  1475 ;        NONE
                15C5  1476 ;
                15C5  1477 ;--
                15C5  1478
                15C5  1479 MODE_ID:
    003C        15C5  1480          .WORD   ^M<R2,R3,R4,R5>
                15C7  1481          $FAO_S  W^CS5,W^MESSAGEL,W^MSGL,MODE ; format the error message
                15DE  1482          $PUTMSG_S W^MSGVEC                   ; print the mode message
      04        15EF  1483          RET
```

```
                                                M 13
                     15F0  1486 MOD_MSG_PRINT:
                     15F0  1487 ;
                     15F0  1488 ;  ***************************************************************
                     15F0  1489 ;  *                                                             *
                     15F0  1490 ;  *  PRINTS THE TEST MODULE BEGUN/SUCCESSFUL/FAILED MESSAGES     *
                     15F0  1491 ;  *       (USING THE PUTMSG MACRO).                             *
                     15F0  1492 ;  *                                                             *
                     15F0  1493 ;  ***************************************************************
                     15F0  1494 ;
                     15F0  1495          PUTMSG  <MOD_MSG_CODE,#2,TMN_ADDR,TMD_ADDR> ; PRINT MSG
                  05 1605  1496          RSB                                         ; ... AND RETURN TO CALLER
                     1606  1497 ;
                     1606  1498 CHMRTN:
                     1606  1499 ;  ***************************************************************
                     1606  1500 ;  *                                                             *
                     1606  1501 ;  *     CHANGE MODE ROUTINE. THIS ROUTINE GETS CONTROL WHENEVER  *
                     1606  1502 ;  *     A CMKRNL, CMEXEC, OR CMSUP SYSTEM SERVICE IS ISSUED     *
                     1606  1503 ;  *     BY THE MODE MACRO ('TO' OPTION).  IT MERELY DOES        *
                     1606  1504 ;  *     A JUMP INDIRECT ON A FIELD SET UP BY MODE. IT HAS       *
                     1606  1505 ;  *     THE EFFECT OF RETURNING TO THE END OF THE MODE          *
                     1606  1506 ;  *     MACRO EXPANSION.                                        *
                     1606  1507 ;  *                                                             *
                     1606  1508 ;  ***************************************************************
                     1606  1509 ;
                0000 1606  1510          .WORD   0                                   ; ENTRY MASK
      1FB3'DF    17 1608  1511          JMP     @CHM_CONT                           ; RETURN TO MODE MACRO IN NEW MODE
                     160C  1512 ;
                     160C  1513 ;  *     RET INSTR WILL BE ISSUED IN EXPANSION OF 'MODE FROM, ....' MACRO
                     160C  1514 ;
                     160C  1515          .END    SATSSS07
```

```
$$ARGS                  = 00000001
$$T1                    = 00000004
$$T2                    = 00000004
A                       = 00000201
A30                       00000C1F R      04
A50                       00000C3C R      04
ARGLST                    00000000 R      02
ARGLST1                   0000000F R      03
BUF                       00000023 R      03
BUF1                      000026B3 R      03
BUFFER                    00000073 R      03
BUFFER_PTR                000027BB R      03
BUF_CHECK                 00000C7A R      04
BUF_SIZ_G               = 00000400
BUF_SIZ_S               = 00000100
CHANNEL_ZERO              000001CB R      02
CHMRTN                    00001606 R      04
CHM_CONT                  00001FB3 R      03
CRE                       00001FB7 R      03
CREATED_FLAG              0000000E R      03
CREMBX                    00000008 R      02
CREMBX$_ACMODE          = 00000018
CREMBX$_BUFQUO          = 00000010
CREMBX$_CHAN            = 00000008
CREMBX$_LOGNAM          = 0000001C
CREMBX$_MAXMSG          = 0000000C
CREMBX$_NARGS           = 00000007
CREMBX$_PRMFLG          = 00000004
CREMBX$_PROMSK          = 00000014
CS1                       0000000F R      02
CS2                       00000041 R      02
CS3                       0000006E R      02
CS4                       000000A1 R      02
CS5                       000000C7 R      02
CTL$GW_CHINDX             ******** X      04
CTRSTR                    00001FD7 R      03
CURRENT_TC                00002063 R      03
DCLCMH                    00000119 R      02
DEL                       00002068 R      03
DELMBX                    00000120 R      02
DELMBX$_CHAN            = 00000004
DELMBX$_NARGS           = 00000001
DIB$K_LENGTH            = 00000074
DIB$W_UNIT              = 0000000C
DIB$W_VPROT             = 00000018
DISABLE                 = 00000000
DSC$K_CLASS_S           = 00000001
DSC$K_DTYPE_T           = 0000000E
D_EQUIV_NAME              000002F5 R      02
D_LOGIC_NAME              000002ED R      02
D_MBX_LOGIC_NAME          000002FD R      02
EXBP                      00000E63 R      04
EM                        00000127 R      02
ENABLE                  = 00000001
EQUIV_NAME                00000009 R      03
ERLB                      00000E67 R      04
ERLBUF_DUMP               00000C3D R      04
ERROR                   = 00000002
EXE$C_CMSTKSZ             ********    X   04
EXP                       00000138 R      02
FLAG                      00000E62 R      04
GETBUF                    00002070 R      03
GRP_PRV_MASK              000001C3 R      02
HANDLER_PC                00000B54 R      04
HIMSG                   = 00002000
INFO                    = 00000003
IO$_READVBLK            = 00000031
IO$_WRITEVBLK           = 00000030
KM                        00000146 R      02
LEN_00_DESCR              000001CD R      02
LEN_255_DESCR             000001D5 R      02
LEN_256_NAME              000001E5 R      02
LEN_63_DESCR              000001DD R      02
LIB$SIGNAL                ********    X   04
LOGIC_NAME                00000004 R      03
LOGNAMG                   00000154 R      02
LOGNAMG1                  00000164 R      02
LOGNAMS                   00000175 R      02
LOGNAMS1                  00000185 R      02
LOMSG                   = 00000001
MBA                       00000305 R      02
MBBUF                   = 00000400
MBCHAN                    000020FC R      03
MBCHAND                   00002102 R      03
MBCHANG                   000020FE R      03
MBCHANS                   00002100 R      03
MBCHAR                    00002104 R      03
MBCHR                     0000210C R      03
MBX_BUF                   00002180 R      03
MBX_LOGIC_NAME            00000000 R      03
MESSAGE1L                 000027B3 R      03
MESSAGEL                  00002580 R      03
ML                        0000258C R      03
MODE                      00002598 R      03
MODE_ID                   000015C5 R      04
MOD_MSG_CODE              00002594 R      03
MOD_MSG_PRINT             000015F0 R      04
MSGT                      00000310 R      02
MSG1L                     000026AB R      03
MSGL                      0000259C R      03
MSGVEC                    0000033A R      02
MSGVEC1                   000025A4 R      03
NONSUB_SSE                000014BF R      04
NSSSF                     000000DC R      02
PB                        000025B4 R      03
PR$_USP                 = 00000003
PRINT_FAIL                00001501 R      04
PRIVMASK                  00002630 R      03
PRM_PRV_MASK              000001B3 R      02
PROT                      0000034A R      02
PRV$V_GRPNAM            = 00000003
PRV$V_PRMMBX            = 0000000B
PRV$V_TMPMBX            = 0000000F
PRVHND1                   00002638 R      03
```

```
PRVPRT                  0000263C R      03
PSL$C_EXEC              =  00000001
PSL$C_KERNEL            =  00000000
PSL$C_SUPER             =  00000002
PSL$C_USER              =  00000003
PSL$S_CURMOD            =  00000002
PSL$V_CURMOD            =  00000018
PSL$V_PRVMOD            =  00000016
REG                     0000263D R      03
REGNUM                  0000268B R      03
REG_CHECK               00000E20 R      04
REG_CHECKNP             00001443 R      04
REG_SAVE                00000E16 R      04
REG_SAVE_AREA           0000264F R      03
RETADR                  0000268F R      03
RETURN_PC               00000B50 R      04
SATSSS07                00000000 RG     04
SERV_NAME               00002697 R      03
SETUP_SUPER             00000B58 R      04
SEVERE                  =  00000004
SF$L_SAVE_FP            =  0000000C
SF$L_SAVE_PC            =  00000010
SHR$R_SHRDEF            =  00000001
SHR$_TEXT               =  00001130
SKIP_DELETE             00000887 R      04
SM                      00000358 R      02
SS$_ACCVIO              =  0000000C
SS$_DEVNOTMBX           =  00000074
SS$_IVCHAN              =  0000013C
SS$_IVLOGNAM            =  00000154
SS$_NOPRIV              =  00000024
SS$_NORMAL              =  00000001
SS$_TOOMANYLNAM         =  00000374
STATUS                  0000269B R      03
STEP                    =  00000017
STORE_STEP              00000DF5 R      04
STP0                    0000003D R      04
STP1                    000000AA R      04
STP10                   00000775 R      04
STP11                   00000799 R      04
STP12                   000007C7 R      04
STP13                   000007FD R      04
STP14                   00000827 R      04
STP15                   00000868 R      04
STP16                   00000887 R      04
STP17                   000008BC R      04
STP18                   00000906 R      04
STP19                   0000099D R      04
STP2                    000001C0 R      04
STP20                   000009D2 R      04
STP21                   00000A3B R      04
STP22                   00000A8A R      04
STP23                   00000ACB R      04
STP3                    00000312 R      04
STP4                    0000047E R      04
STP5                    00000568 R      04
STP6                    000005A3 R      04
STP7                    00000638 R      04
STP8                    000006CD R      04
STP9                    0000074B R      04
STS$V_INHIB_MSG         =  0000001C
SUCCESS                 =  00000001
SUPER_MODE              00000BAC R      04
SYS$ASSIGN              ********  GX    04
SYS$CMEXEC              ********  GX    04
SYS$CMKRNL              ********  GX    04
SYS$CRELOG              ********  GX    04
SYS$CREMBX              ********  GX    04
SYS$DASSGN              ********  GX    04
SYS$DCLCMH              ********  GX    04
SYS$DELMBX              ********  GX    04
SYS$EXIT                ********  GX    04
SYS$FAO                 ********   X    04
SYS$FAOL                ********  GX    04
SYS$GETCHN              ********  GX    04
SYS$GETMSG              ********  GX    04
SYS$HIBER               ********  GX    04
SYS$PUTMSG              ********  GX    04
SYS$QIO                 ********  GX    04
SYS$SETPRN              ********  GX    04
SYS$SETPRV              ********  GX    04
SYS$WAKE                ********  GX    04
SYS_DEV                 00000196 R      02
TEST_DATA               00000365 R      02
TEST_MOD_BEGIN          00000765 R      02
TEST_MOD_FAIL           0000076B R      02
TEST_MOD_NAME           00000772 R      02
TEST_MOD_NAME_D         0000077B R      02
TEST_MOD_SUCC           0000078B R      02
TEXT_BUFFER             =  00000100
TMD_ADDR                0000269F R      03
TMN_ADDR                000026A3 R      03
TMP_PRM_PRV_MASK        000001BB R      02
TMP_PRV_MASK            000001AB R      02
TPID                    000026A7 R      03
UETP$_DATAER            =  00748010
UETP$_SATSMS            =  007480D9
UETP$_TEXT              =  00741133
UM                      00000796 R      02
VERIFY_MBX              00000CFC R      04
WARNING                 =  00000000
ZERO_ADDR_DESCR         000002E5 R      02
```

C 14

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
RODATA	000007A2 (1954.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
RWDATA	000027C3 (10179.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
SATSSS07	0000160C (5644.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.10	00:00:00.35
Command processing	108	00:00:00.66	00:00:02.28
Pass 1	609	00:00:25.28	00:00:36.31
Symbol table sort	0	00:00:02.46	00:00:02.79
Pass 2	309	00:00:06.60	00:00:08.51
Symbol table output	26	00:00:00.19	00:00:00.19
Psect synopsis output	2	00:00:00.02	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1087	00:00:35.32	00:00:50.46

The working set limit was 1950 pages.
168419 bytes (329 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1495 non-local and 38 local symbols.
1515 source lines were read in Pass 1, producing 41 object records in Pass 2.
61 pages of virtual memory were used to define 57 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYSLIB]STARLET.MLB,2	43
_255DUA28:[SHRLIB]UETP.MLB;1	11
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	0
TOTALS (all libraries)	54

1756 GETS were required to define 54 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:SATSSS07/OBJ=OBJ$:SATSSS07 MSRC$:SATSSS07/UPDATE=(ENH$:SATSSS07)+EXECML$/LIB+SHRLIB$:UETP/LIB
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SATSSS08
LIS

SATSSS05
LIS

SATSSS22
LIS

SATSSS07
LIS

SATSSS01
LIS