```
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSSS  YYY         YYY
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSSS  YYY         YYY
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPrPPPPPPPPP        SSSSSSSSSSSS  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
UUU         UUU  EEE                    TTT        PPP                          SSS        YYY
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP              SSSSSSSSSSSS           YYY
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP              SSSSSSSSSSSS           YYY
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP              SSSSSSSSSSSS           YYY
```

```
**FILE**ID**SATSSS22                                  I 15




  SSSSSSSS    AAAAAA    TTTTTTTTTT    SSSSSSSS    SSSSSSSS    SSSSSSSS    222222      222222
  SSSSSSSS    AAAAAA    TTTTTTTTTT    SSSSSSSS    SSSSSSSS    SSSSSSSS    222222      222222
SS          AA      AA      TT      SS          SS          SS          22      22  22      22
SS          AA      AA      TT      SS          SS          SS          22      22  22      22
SS          AA      AA      TT      SS          SS          SS                  22          22
SS          AA      AA      TT      SS          SS          SS                  22          22
  SSSSSS    AA      AA      TT        SSSSSS      SSSSSS      SSSSSS          22          22
  SSSSSS    AA      AA      TT        SSSSSS      SSSSSS      SSSSSS          22          22
        SS  AAAAAAAAAA      TT              SS          SS          SS      22          22
        SS  AAAAAAAAAA      TT              SS          SS          SS      22          22
        SS  AA      AA      TT              SS          SS          SS    22          22            ....
        SS  AA      AA      TT              SS          SS          SS    22          22            ....
SSSSSSSS    AA      AA      TT      SSSSSSSS    SSSSSSSS    SSSSSSSS    2222222222  2222222222      ....
SSSSSSSS    AA      AA      TT      SSSSSSSS    SSSSSSSS    SSSSSSSS    2222222222  2222222222      ....

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

```
0000           .TITLE  SATSSS22 - SATS SYSTEM SERVI'' TESTS  (SUCC S.C.)
0000           .IDENT  'V04-000'
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000    ;++
0000    ; FACILITY:     SATS SYSTEM SERVICE TESTS
0000    ;
0000    ; ABSTRACT:     The SATSSS22 module tests the execution of the following
0000    ;               VMS system services:
0000    ;
0000    ;               $SETEXV
0000    ;               $SETSFM
0000    ;               $UNWIND
0000    ;
0000    ;
0000    ; ENVIRONMENT:  User, Supervisor and Executive mode image.
0000    ;               Needs CMKRNL privilege and dynamically acquires other
0000    ;               privileges, as needed.
0000    ;
0000    ; AUTHOR: THOMAS L. CAFARELLA,          CREATION DATE: MMM, 1978
0000    ;         PAUL D. FAY (DISPSERV & TESTSERV MACROS)
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-002 LDJ0002         Larry D. Jones,         11-Mar-1981
0000    ;               Modified to match a VMS change in exception stack frame size
0000    ;               for VMS version 3.0.
0000    ;
0000    ;       V03-001 LDJ0001         Larry D. Jones,         17-Sep-1980
0000    ;               Modified to conform to new build command procedures.
0000    ;**
0000    ;--
```

```
          0000         .SBTTL  DECLARATIONS
          0000   ;
          0000   ; MACRO LIBRARY CALLS
          0000   ;
          0000         .LIBRARY /SYS$LIBRARY:STARLET.MLB/
          0000         $CHFDEF                                 ; Condition handler frame definitions
          0000         $PRDEF                                  ; processor register definitions
          0000         $PRVDEF                                 ; privilege definitions
          0000         $PSLDEF                                 ; PSL definitions
          0000         $SFDEF                                  ; Stack frame definitions
          0000         $SHR_MESSAGES UETP,116,<<TEXT,INFO>> ; UETP$_TEXT definition
          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        ''      ''    ''   ''
          0000   ;
          0000   ; MACROS
          0000   ;
          0000         .MACRO SEVT MODE,PRINT,?LAB1
          0000         .LIST ME
          0000   ;+
          0000   ;
          0000   ; The next section will declare 2 mode exception handlers.
          0000   ; A primary and secondary handler will be
          0000   ; set using the $SETEXV_G system service.
          0000   ;
          0000   ;-
          0000         .NLIST ME
          0000         .LIST MEB
          0000                 CLRL    R5                      ; set init. handler type
          0000                 MOVL    #PSL$C_'MODE,W^SET+SETEXV$_ACMODE ; set access mode
          0000   LAB1:
          0000                 PUSHL   #0                      ; push dummy parameter
          0000                 CALLS   #1,W^REG_SAVE           ; save register snapshot
          0000                 MOVL    R5,W^SET+SETEXV$_VECTOR ; set vector type
          0000                 MOVAL   W^PRE_'MODE'_PRI[R5],-
          0000                         W^SET+SETEXV$_PRVHND    ; set previous handler
          0000                 MOVL    W^'MODE'_HANTAB[R5],-
          0000                         W^SET+SETEXV$_ADDRES    ; set handler address
          0000                 $SETEXV_G W^SET                 ; declare the handler
          0000         .IF IDN,PRINT,NO
          0000                 FAIL_CHECKNP SS$_NORMAL         ; check for success
          0000         .IFF
          0000                 FAIL_CHECK SS$_NORMAL           ; check for success
          0000         .ENDC
          0000         .LIST MEB
          0000                 INCL    W^CURRENT_TC            ; increment step number
          0000                 AOBLEQ  #1,R5,LAB1              ; do all 1 types
          0000                 STEP=STEP+1                     ; bump the step # variable
          0000         .LIST ME
          0000   ;+
0000    ;
0000    ; An exception will now be caused to check the handlers.
0000    ;
0000    ;-
0000            .NLIST ME
0000            .LIST MEB
0000                    BISB2   #2,W^FLAG1                  ; set excep. should occur
0000                    CHMU    #0                          ; cause an exception
0000                    BRB     MODE'_END                   ; go on
0000    MODE'_HANTAB:
0000                    .ADDRESS MODE'_PRIM                 ; handler address table
0000                    .ADDRESS MODE'_SEC
0000    ;
0000    PRE_'MODE'_PRI:
0000                    .LONG   0                           ; previous handler table
0000    PRE_'MODE'_SEC:
0000                    .LONG   0
0000    ;
0000            .LIST ME
0000    ;+
0000    ;
0000    ; test the mode primary exception handler
0000    ;
0000    ;-
0000            .NLIST ME
0000            .LIST MEB
0000    MODE'_PRIM:
0000                    .WORD   0
0000                    NEXT_TEST
0000            .LIST MEB
0000                    INCB    W^FLAG1                     ; set excep. did occur
0000            .IF IDN,PRINT,NO
0000                    BSBW    EXCEP_CHECKNP
0000            .IFF
0000                    BSBW    EXCEP_CHECK                 ; check primary handler
0000            .ENDC
0000                    MOVL    #SS$_RESIGNAL,R0            ; and resignal
0000                    DECB    W^FLAG1                     ; reset excep. did occur
0000                    RET
0000            .LIST ME
0000    ;+
0000    ;
0000    ; test the mode secondary handler and clean up the exception
0000    ;
0000    ;-
0000            .NLIST ME
0000            .LIST MEB
0000    MODE'_SEC:
0000                    .WORD   0
0000                    NEXT_TEST
0000            .LIST MEB
0000                    INCB    W^FLAG1                     ; set excep. did occur
0000            .IF IDN,PRINT,NO
0000                    BSBW EXCEP_CHECKNP
0000            .IFF
0000                    BSBW    EXCEP_CHECK                 ; check secondary handler
0000            .ENDC
```

```
    0000   172                     $UNWIND_S                        ; clean the stack
    0000   173                     MOVL    #SS$_CONTINUE,R0         ; and resignal
    0000   174                     BICB2   #3,W^FLAG1               ; clear excep. did & should FLAG1
    0000   175                     RET
    0000   176             .LIST ME
    0000   177  ;+
    0000   178  ;
    0000   179  ; the mode last chance handler can not be tested because
    0000   180  ; it will always force an exit from the process.
    0000   181  ;
    0000   182  ; reset the mode primary handler
    0000   183  ;
    0000   184  ;-
    0000   185             .NLIST ME
    0000   186             .LIST MEB
    0000   187 MODE'_END:
    0000   188                     NEXT_TEST
    0000   189             .LIST MEB
    0000   190                     $SETEXV_S #0,@W^PRE_'MODE'_PRI,#PSL$C_'MODE ; reset the handlers
    0000   191             .IF IDN,PRINT,NO
    0000   192                     FAIL_CHECKNP SS$_NORMAL          ; check for success
    0000   193             .IFF
    0000   194                     FAIL_CHECK SS$_NORMAL
    0000   195             .ENDC
    0000   196             .LIST ME
    0000   197  ;+
    0000   198  ;
    0000   199  ; reset the mode secondary handler
    0000   200  ;
    0000   201  ;-
    0000   202             .NLIST ME
    0000   203             .LIST MEB
    0000   204                     NEXT_TEST
    0000   205             .LIST MEB
    0000   206                     $SETEXV_S #1,@W^PRE_'MODE'_SEC,#PSL$C_'MODE'
    0000   207             .IF IDN,PRINT,NO
    0000   208                     FAIL_CHECKNP SS$_NORMAL ; check for success
    0000   209             .IFF
    0000   210                     FAIL_CHECK SS$_NORMAL
    0000   211             .ENDC
    0000   212             .ENDM SEVT
```

```
                              00000000   214          .PSECT  RODATA,RD,NOWRT,NOEXE,LONG
                                  0000   215  ;
                                  0000   216  TEST_MOD_NAME:
        32 32 53 53 53 54 41 53 00' 0000   217          .ASCIC  /SATSSS22/              ; needed for SATSMS message
                              08  0000
                                  0009   218  TEST_MOD_NAME_D:
53 53 53 54 41 53 00000011'010E0000' 0009   219          .ASCID /SATSSS22/               ; module name
                           32 32  0017
                                  0019   220  TEST_MOD_BEGIN:
                  6E 75 67 65 62 00' 0019   221          .ASCIC  /begun/
                              05  0019
                                  001F   222  TEST_MOD_SUCC:
   6C 75 66 73 73 65 63 63 75 73 00' 001F   223          .ASCIC  /successful/
                              0A  001F
                                  002A   224  TEST_MOD_FAIL:
               64 65 6C 69 61 66 00' 002A   225          .ASCIC  /failed/
                              06  002A
                                  0031   226  SETEXV:
               56 58 45 54 45 53 00' 0031   227          .ASCIC  /SETEXV/
                              06  0031
                                  0038   228  SETSFM:
               4D 46 53 54 45 53 00' 0038   229          .ASCIC  /SETSFM/
                              06  0038
                                  003F   230  UNWIND:
               44 4E 49 57 4E 55 00' 003F   231          .ASCIC  /UNWIND/
                              06  003F
                                  0046   232  CS1:
21 20 74 73 65 54 0000004E'010E0000' 0046   233          .ASCID  \Test !AC service name !AC step !UL failed.\
6E 20 65 63 69 76 72 65 73 20 43 41  0054
70 65 74 73 20 43 41 21 20 65 6D 61  0060
2E 64 65 6C 69 61 66 20 4C 55 21 20  006C
                                  0078   234  CS2:
74 63 65 70 78 45 00000080'010E0000' 0078   235          .ASCID  \Expected !AS = !XL received !AS = !XL\
4C 58 21 20 3D 20 53 41 21 20 64 65  0086
41 21 20 64 65 76 69 65 63 65 72 20  0092
            4C 58 21 20 3D 20 53  009E
                                  00A5   236  CS3:
74 63 65 70 78 45 000000AD'010E0000' 00A5   237          .ASCID  \Expected !AS!UB = !XL received !AS!UB = !XL\
20 3D 20 42 55 21 53 41 21 20 64 65  00B3
64 65 76 69 65 63 65 72 20 4C 58 21  00BF
58 21 20 3D 20 42 55 21 53 41 21 20  00CB
                              4C  00D7
                                  00D8   238  CS4:
65 70 78 65 6E 55 000000E0'010E0000' 00D8   239          .ASCID  \Unexpected !AS mode exception occured in !AC step !UL.\
64 6F 6D 20 53 41 21 20 64 65 74 63  00E6
20 6E 6F 69 74 70 65 63 78 65 20 65  00F2
21 20 6E 69 20 64 65 72 75 63 63 6F  00FE
2E 4C 55 21 20 70 65 74 73 20 43 41  010A
                                  0116   240  CS5:
77 20 65 64 6F 4D 0000011E'010E0000' 0116   241          .ASCID  \Mode was !AS.\
            2E 53 41 21 20 73 61  0124
                                  012B   242  CS6:
72 69 75 71 65 52 00000133'010E0000' 012B   243          .ASCID  \Required !AS mode exception did'nt occur in !AC step !UL.\
20 65 64 6F 6D 20 53 41 21 20 64 65  0139
69 64 20 6E 6F 69 74 70 65 63 78 65  0145
69 20 72 75 63 63 6F 20 74 6E 27 64  0151
21 20 70 65 74 73 20 43 41 21 20 6E  015D
```

```
                                2E 4C 55  0169
                                          016C      244 UM:
          72 65 73 75 00000174'010E0000'  016C      245          .ASCID  \user\
                                          0178      246 SM:
       72 65 70 75 73 00000180'010E0000'  0178      247          .ASCID  \super\
                                          0185      248 EM:
74 75 63 65 78 65 0000018D'010E0000'      0185      249          .ASCID  \executive\
                                65 76 69  0193
                                          0196      250 KM:
6C 65 6E 72 65 6B 0000019E'010E0000'      0196      251          .ASCID  \kernel\
                                          01A4      252 EXP:
73 75 74 61 74 73 000001AC'010E0000'      01A4      253          .ASCID  \status\
                                          01B2      254 STACK:
                50 53 000001BA'010E0000'  01B2      255          .ASCID  \SP\
                                          01BC      256 RETPC:
6E 72 75 74 65 72 000001C4'010E0000'      01BC      257          .ASCID  \return PC\
                                43 50 20  01CA
                                          01CD      258 ARGLST:
                               00000001   01CD      259          .LONG   1                          ; super mode setup arg list
                               00000838'  01D1      260          .ADDRESS SUPER_MODE
                                          01D5      261 MSGVEC:                                     ; PUTMSG message vector
                               00000003   01D5      262          .LONG   3
                               00741133   01D9      263          .LONG   UETP$_TEXT
                               00000001   01DD      264          .LONG   1
                               00000107'  01E1      265          .ADDRESS MESSAGE
```

```
                                  01E5   ;
                                  01E5          .SBTTL  R/W PSECT
                              00000000          .PSECT  RWDATA,RD,WRT,NOEXE,LONG
                                  0000   ;
                                  0000   TPID:
                        00000000  0000          .LONG   0                       ; PID for this process
                                  0004   CURRENT_TC:
                        00000000  0004          .LONG   0                       ; ptr to current test case
                                  0008          .ALIGN LONG
                                  0008   REG_SAVE_AREA:
                        00000044  0008          .BLKL   15                      ; register save area
                                  0044   MOD_MSG_CODE:
                        007480D9  0044          .LONG   UETP$_SATSMS            ; test module message code for putmsg
                                  0048   TMN_ADDR:
                        00000000' 0048          .ADDRESS TEST_MOD_NAME
                                  004C   TMD_ADDR:
                        00000019' 004C          .ADDRESS TEST_MOD_BEGIN
                                  0050   PRVPRT:
                              00  0050          .BYTE   0                       ; protection return byte for SETPRT
                                  0051   PRIVMASK:
               00000000 00000000  0051          .QUAD   0                       ; priv. mask
                                  0059   CHM_CONT:
                        00000000  0059          .LONG   0                       : change mode continue address
                                  005D   RETADR:
                        00000065  005D          .BLKL   2                       ; returned address's from SETPRT
                                  0065   STATUS:
                        00000000  0065          .LONG   0
                                  0069   MODE:
                        00000000  0069          .LONG   0
                                  006D   SET:
                                  006D          $SETEXV 0,0,0,0                 ; SETEXV parameter list
                                  0081   SET1:
                                  0081          $SETSFM 0                       ; SETSFM parameter list
                                  0089   UNW:
                                  0089          $UNWIND DEPTH,0                 ; UNWIND parameter list
                                  0095   REG:
74 73 69 67 65 72 0000009D'010E0000' 0095       .ASCID  \register R\
                     52 20 72 65  00A3
                                  00A7   REGNUM:
                        00000000  00A7          .LONG   0                       ; register number
                                  00AB   PRVHND1:
                        00000000  00AB          .LONG   0                       ; previous handler address
                                  00AF   MSGL:
                        00000050  00AF          .LONG   80                      ; buffer desc.
                        000000B7' 00B3          .ADDRESS BUF
                                  00B7   BUF:
                        00000107  00B7          .BLKB   80
                                  0107   MESSAGEL:
                        00000000  0107          .LONG   0                       ; message desc.
                        000000B7' 010B          .ADDRESS BUF
                                  010F   SERV_NAME:
                        00000000  010F          .LONG   0                       ; service name pointer
                                  0113   MSGVEC1:
                        00000004  0113          .LONG   4                       ; PUTMSG message vector for exit
                        00000000  0117          .LONG   0
                        00000002  011B          .LONG   2
                        00000127  011F          .BLKL   2
```

```
              0127   323 FLAG1:                          ; flag byte
          00  0127   324         .BYTE   0               ; BIT0 = 0 exception did'nt occur
              0128   325                                 ; BIT0 = 1 exception occured
              0128   326                                 ; BIT1 = 0 exception should'nt have occured
              0128   327                                 ; BIT1 = 1 exception should have occured
              0128   328 DEPTH:
    00000001  0128   329         .LONG   1               ; unwind depth indicator
```

```
00000000          .PSECT  SATSSS22,RD,WRT,EXE,LONG
    0000          .SBTTL  SATSSS22
    0000    ;++
    0000    ; FUNCTIONAL DESCRIPTION:
    0000    ;
    0000    ;         After performing some initial housekeeping, such as
    0000    ; printing the module begin message and acquiring needed privileges,
    0000    ; the system services are tested in each of their normal conditions.
    0000    ; Detected failures are identified and  an error message is printed
    0000    ; on the terminal.  Upon completion of the test a success or fail
    0000    ; message is printed on the terminal.
    0000    ;
    0000    ; CALLING SEQUENCE:
    0000    ;
    0000    ;         $ RUN SATSSS22  ...  (DCL COMMAND)
    0000    ;
    0000    ; INPUT PARAMETERS:
    0000    ;
    0000    ;         none
    0000    ;
    0000    ; IMPLICIT INPUTS:
    0000    ;
    0000    ;         none
    0000    ;
    0000    ; OUTPUT PARAMETERS:
    0000    ;
    0000    ;         none
    0000    ;
    0000    ; IMPLICIT OUTPUTS:
    0000    ;
    0000    ;         Messages to SYS$OUTPUT are the only output from SATSSS22.
    0000    ;         They are of the form:
    0000    ;
    0000    ;                 %UETP-S-SATSMS, TEST MODULE SATSSS22 BEGUN ... (BEGIN MSG)
    0000    ;                 %UETP-S-SATSMS, TEST MODULE SATSSS22 SUCCESSFUL ... (END MSG)
    0000    ;                 %UETP-E-SATSMS, TEST MODULE SATSSS22 FAILED ... (END MSG)
    0000    ;                 %UETP-I-TEXT, ... (VARIABLE INFORMATION ABOUT A TEST MODULE FAILURE)
    0000    ;
    0000    ; COMPLETION CODES:
    0000    ;
    0000    ;         The SATSSS22 routine terminates with a $EXIT to the
    0000    ;         operating system with a status code defined by UETP$_SATSMS.
    0000    ;
    0000    ; SIDE EFFECTS:
    0000    ;
    0000    ;         none
    0000    ;
    0000    ;--
    0000
    0000
    0000
    0000              TEST_START SATSSS22             ; let the test begin
```

```
                            0000
                       0000 0000                .ENTRY SATSSS22,0
           0004'CF       D4 0002                CLRL    W^CURRENT_TC
                00       DD 0006                PUSHL   #0
           0000'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
           0009'CF       7F 001A                PUSHAQ  W^TEST_MOD_NAME_D
       00000000'GF   01  FB 001E                CALLS   #1,G^SYS$SETPRN
              102D       30 0025                BSBW    W^MOD_MSG_PRINT
     004C'CF    001F'CF  DE 0028                MOVAL   W^TEST_MOD_SUCC,W^TMD_ADDR
  0044'CF    03    00    01 F0 002F             INSV    #SUCCESS,#0,#3,W^MOD_MSG_CODE
                00       DD 0036                PUSHL   #0
        096E'CF      01  FB 0038                CALLS #1,W^REG_SAVE
                            003D         STP0:
                            003D              .SBTTL  SETEXV TESTS
                            003D         ;+
                            003D         ;
                            003D         ; $SETEXV tests
                            003D         ;
                            003D         ; test user mode
                            003D         ;
                            003D         ;-
                            003D              $CMKRNL_S W^SETUP_SUPER,W^ARGLST        ; declare CHMS handler
          5E     10'     C0 004C              ADDL2   S^#EXE$C_CMSTKSZ+16,SP          ; fix the user stackpointer
          5D     5E      D0 004F              MOVL    SP,FP                           ; and user framepointer
     0E41'CF     00      FB 0052              CALLS   #0,W^ERLBUF_DUMP                ; dump any errors that occured at ke
  010F'CF    0031'CF     DE 0057              MOVAL   W^SETEXV,W^SERV_NAME            ; set service name
  0069'CF    016C'CF     DE 005E              MOVAL   W^UM,W^MODE                     ; set the mode
                            0065              SEVT    USER,YES                        ; do the user mode tests
                            0065
                            0065         ;+
                            0065         ;
                            0065         ; The next section will declare 2 USER exception handlers.
                            0065         ; A primary and secondary handler will be
                            0065         ; set using the $SETEXV_G system service.
                            0065         ;
                            0065         ;-
                55       D4 0065                CLRL    R5                            ; set init. handler type
       0079'CF  03       D0 0067                MOVL    #PSL$C_USER,W^SET+SETEXV$_ACMODE ; set access USER
                            006C         30000$:
                00       DD 006C                PUSHL   #0                            ; push dummy parameter
       096E'CF  01       FB 006E                CALLS   #1,W^REG_SAVE                 ; save register snapshot
       0071'CF  55       D0 0073                MOVL    R5,W^SET+SETEXV$_VECTOR ; set vector type
  007D'CF    00B5'CF45   DE 0078                MOVAL   W^PRE_USER_PRI[R5],-
  0075'CF    00AD'CF45   D0 0080                MOVL    W^USER_HANTAB[R5],-
00000000'GF    006D'CF   FA 0088                CALLG   W^SET,G^SYS$SETEXV
         00000000'8F     DD 0091                PUSHL   #SS$_NORMAL
       0978'CF  01       FB 0097                CALLS   #1,W^REG_CHECK
           0004'CF       D6 009C                INCL    W^CURRENT_TC                  ; increment step number
        C8 55   01       F3 00A0                AOBLEQ  #1,R5,30000$                  ; do all 1 types
                            00A4              .LIST ME
                            00A4         ;+
                            00A4         ;
                            00A4         ; An exception will now be caused to check the handlers.
                            00A4         ;
                            00A4         ;-
        0127'CF   02    88  00A4                      BISB2   #2,W^FLAG1              ; set excep. should occur
              00    BF  00A9                      CHMU    #0                      ; cause an exception
              5E    11  00AB                      BRB     USER_END                ; go on
                        00AD       USER_HANTAB:
               000000BD' 00AD                     .ADDRESS USER_PRIM              ; handler address table
               000000DE' 00B1                     .ADDRESS USER_SEC
                        00B5       PRE_USER_PRI:
               00000000  00B5                     .LONG   0                       ; previous handler table
                        00B9       PRE_USER_SEC:
               00000000  00B9                     .LONG   0
                        00BD              .LIST ME
                        00BD       ;+
                        00BD       ;
                        00BD       ; test the USER primary exception handler
                        00BD       ;
                        00BD       ;-
                        00BD       USER_PRIM:
                    0000 00BD                     .WORD   0
                        00BF              .LIST ME
                        00BF       STP2:
        0004'CF   02    D0  00BF                      MOVL    #2,W^CURRENT_TC
              00    DD  00C4                      PUSHL   #0
        096E'CF   01    FB  00C6                      CALLS   #1,W^REG_SAVE
            0127'CF     96  00CB                      INCB    W^FLAG1                 ; set excep. did occur
               0EDF     30  00CF                      BSBW    EXCEP_CHECK             ; check primary handler
    50   00000000'8F    D0  00D2                      MOVL    #SS$_RESIGNAL,R0        ; and resignal
            0127'CF     97  00D9                      DECB    W^FLAG1                 ; reset excep. did occur
                    04  00DD                      RET
                        00DE              .LIST ME
                        00DE       ;+
                        00DE       ;
                        00DE       ; test the USER secondary handler and clean up the exception
                        00DE       ;
                        00DE       ;-
                        00DE       USER_SEC:
                    0000 00DE                     .WORD   0
                        00E0              .LIST ME
                        00E0       STP3:
        0004'CF   03    D0  00E0                      MOVL    #3,W^CURRENT_TC
              00    DD  00E5                      PUSHL   #0
        096E'CF   01    FB  00E7                      CALLS   #1,W^REG_SAVE
            0127'CF     96  00EC                      INCB    W^FLAG1                 ; set excep. did occur
               0EBE     30  00F0                      BSBW    EXCEP_CHECK             ; check secondary handler
              00    DD  00F3                      PUSHL   #0
              00    DD  00F5                      PUSHL   #0
    00000000'GF   02    FB  00F7                      CALLS   #2,G^SYS$UNWIND
    50   00000000'8F    D0  00FE                      MOVL    #SS$_CONTINUE,R0        ; and resignal
        0127'CF   03    8A  0105                      BICB2   #3,W^FLAG1              ; clear excep. did & should FLAG1
                    04  010A                      RET
                        010B              .LIST ME
                        010B       ;+
                        010B       ;
                        010B       ; the USER last chance handler can not be tested because
                        010B       ; it will always force an exit from the process.
                        010B       ;
                        010B       ; reset the USER primary handler
                        010B       ;
```

```
                              010B         ;-
                              010B         USER_END:
                              010B                 .LIST ME
                              010B         STP4:
        0004'CF    04    D0   010B                         MOVL    #4,W^CURRENT_TC
                   00    DD   0110                         PUSHL   #0
        096E'CF    01    FB   0112                         CALLS   #1,W^REG_SAVE
                   00    DD   0117                         PUSHL   #0
                   03    DD   0119                         PUSHL   #PSL$C_USER
             FF96  DF    DF   011B                         PUSHAL  @W^PRE_USER_PRI
                   00    DD   011F                         PUSHL   #0
    00000000'GF    04    FB   0121                         CALLS   #4,G^SYS$SETEXV
      00000000'8F        DD   0128                         PUSHL   #SS$_NORMAL
        0978'CF    01    FB   012E                         CALLS   #1,W^REG_CHECK
                              0133
                              0133         ;+
                              0133         ;
                              0133         ; reset the USER secondary handler
                              0133         ;
                              0133         ;-
                              0133                 .LIST ME
                              0133         STP5:
        0004'CF    05    D0   0133                         MOVL    #5,W^CURRENT_TC
                   00    DD   0138                         PUSHL   #0
        096E'CF    01    FB   013A                         CALLS   #1,W^REG_SAVE
                   00    DD   013F                         PUSHL   #0
                   03    DD   0141                         PUSHL   #PSL$C_USER
             FF72  DF    DF   0143                         PUSHAL  @W^PRE_USER_SEC
                   01    DD   0147                         PUSHL   #1
    00000000'GF    04    FB   0149                         CALLS   #4,G^SYS$SETEXV
      00000000'8F        DD   0150                         PUSHL   #SS$_NORMAL
        0978'CF    01    FB   0156                         CALLS   #1,W^REG_CHECK
                              015B         ;+
                              015B         ;
                              015B         ; test super mode
                              015B         ;
                              015B         ;-
                              015B                 NEXT_TEST
                              015B
                              015B         STP6:
        0004'CF    06    D0   015B                         MOVL    #6,W^CURRENT_TC
                   00    DD   0160                         PUSHL   #0
        096E'CF    01    FB   0162                         CALLS   #1,W^REG_SAVE
  0069'CF   0178'CF      DE   0167                 MOVAL   W^SM,W^MODE                     ; set the mode
                   01    BE   016E                 CHMS    #1                              ; do the super mode tests
                  0000000B    0170                 STEP=STEP+5
                              0170         ;+
                              0170         ;
                              0170         ; test exec mode
                              0170         ;
                              0170         ;-
                              0170                 NEXT_TEST
                              0170
                              0170         STP12:
        0004'CF    0C    D0   0170                         MOVL    #12,W^CURRENT_TC
                   00    DD   0175                         PUSHL   #0
        096E'CF    01    FB   0177                         CALLS   #1,W^REG_SAVE
```

```
     0069'CF   0185'CF   DE  017C    413          MOVAL   W^EM,W^MODE                         ; set the mode
                             0183    414          MODE TO,B10,EXEC,NOREGS                     ; set mode to exec
                             01A0    415          SEVT    EXEC,NO                             ; do the exec mode tests
                             01A0
                             01A0            ;+
                             01A0            ;
                             01A0            ; The next section will declare 2 EXEC exception handlers.
                             01A0            ; A primary and secondary handler will be
                             01A0            ; set using the $SETEXV_G system service.
                             01A0            ;
                             01A0            ;-
                    55   D4  01A0                          CLRL    R5                          ; set init. handler type
          0079'CF   01   D0  01A2                          MOVL    #PSL$C_EXEC,W^SET+SETEXV$_ACMODE ; set access EXEC
                             01A7            30002$:
                    00   DD  01A7                          PUSHL   #0                          ; push dummy parameter
          096E'CF   01   FB  01A9                          CALLS   #1,W^REG_SAVE               ; save register snapshot
          0071'CF   55   D0  01AE                          MOVL    R5,W^SET+SETEXV$_VECTOR ; set vector type
     007D'CF   01F0'CF45  DE  01B3                          MOVAL   W^PRE_EXEC_PRI[R5],-
     0075'CF   01E8'CF45  D0  01BB                          MOVL    W^EXEC_HANTAB[R5],-
  00000000'GF    006D'CF  FA  01C3                          CALLG   W^SET,G^SYS$SETEXV
         00000000'8F     DD  01CC                          PUSHL   #SS$_NORMAL
          0DA7'CF   01   FB  01D2                          CALLS   #1,W^REG_CHECKNP
              0004'CF    D6  01D7                          INCL    W^CURRENT_TC                ; increment step number
         C8 55      01   F3  01DB                          AOBLEQ  #1,R5,30002$                ; do all 1 types
                             01DF                  .LIST ME
                             01DF            ;+
                             01DF            ;
                             01DF            ; An exception will now be caused to check the handlers.
                             01DF            ;
                             01DF            ;-
          0127'CF   02   88  01DF                          BISB2   #2,W^FLAG1                  ; set excep. should occur
                    00   BF  01E4                          CHMU    #0                          ; cause an exception
                    5E   11  01E6                          BRB     EXEC_END                    ; go on
                             01E8            EXEC_HANTAB:
                 000001F8'   01E8                          .ADDRESS EXEC_PRIM                  ; handler address table
                 00000219'   01EC                          .ADDRESS EXEC_SEC
                             01F0            PRE_EXEC_PRI:
                 00000000    01F0                          .LONG   0                           ; previous handler table
                             01F4            PRE_EXEC_SEC:
                 00000000    01F4                          .LONG   0
                             01F8                  .LIST ME
                             01F8            ;+
                             01F8            ;
                             01F8            ; test the EXEC primary exception handler
                             01F8            ;
                             01F8            ;-
                             01F8            EXEC_PRIM:
                     0000    01F8                          .WORD   0
                             01FA                  .LIST ME
                             01FA            STP14:
          0004'CF   0E   D0  01FA                          MOVL    #14,W^CURRENT_TC
                    00   DD  01FF                          PUSHL   #0
          096E'CF   01   FB  0201                          CALLS   #1,W^REG_SAVE
              0127'CF    96  0206                          INCB    W^FLAG1                     ; set excep. did occur
                 0DE6    30  020A                          BSBW    EXCEP_CHECKNP
      50   00000000'8F   D0  020D                          MOVL    #SS$_RESIGNAL,R0            ; and resignal
              0127'CF    97  0214                          DECB    W^FLAG1                     ; reset excep. did occur
```

```
                     04  0218                              RET
                         0219                  .LIST ME
                         0219          ;+
                         0219          ;
                         0219          ; test the EXEC secondary handler and clean up the exception
                         0219          ;
                         0219          ;-
                         0219          EXEC_SEC:
                   0000  0219                              .WORD   0
                         021B                  .LIST ME
                         021B          STP15:
        0004'CF    0F    D0  021B                          MOVL    #15,W^CURRENT_TC
                   00    DD  0220                          PUSHL   #0
        096E'CF    01    FB  0222                          CALLS   #1,W^REG_SAVE
           0127'CF       96  0227                          INCB    W^FLAG1                 ; set excep. did occur
              0DC5       30  022B                          BSBW EXCEP_CHECKNP
                   00    DD  022E                          PUSHL   #0
                   00    DD  0230                          PUSHL   #0
    00000000'GF    02    FB  0232                          CALLS   #2,G^SYS$UNWIND
    50  00000000'8F      D0  0239                          MOVL    #SS$_CONTINUE,R0        ; and resignal
        0127'CF    03    8A  0240                          BICB2   #3,W^FLAG1              ; clear excep. did & should FLAG1
                   04        0245                          RET
                         0246                  .LIST ME
                         0246          ;+
                         0246          ;
                         0246          ; the EXEC last chance handler can not be tested because
                         0246          ; it will always force an exit from the process.
                         0246          ;
                         0246          ; reset the EXEC primary handler
                         0246          ;
                         0246          ;-
                         0246          EXEC_END:
                         0246                  .LIST ME
                         0246          STP16:
        0004'CF    10    D0  0246                          MOVL    #16,W^CURRENT_TC
                   00    DD  024B                          PUSHL   #0
        096E'CF    01    FB  024D                          CALLS   #1,W^REG_SAVE
                   00    DD  0252                          PUSHL   #0
                   01    DD  0254                          PUSHL   #PSL$C_EXEC
            FF96 DF      DF  0256                          PUSHAL  @W^PRE_EXEC_PRI
                   00    DD  025A                          PUSHL   #0
    00000000'GF    04    FB  025C                          CALLS   #4,G^SYS$SETEXV
       00000000'8F       DD  0263                          PUSHL   #SS$_NORMAL
        0DA7'CF    01    FB  0269                          CALLS   #1,W^REG_CHECKNP
                         026E
                         026E          ;+
                         026E          ;
                         026E          ; reset the EXEC secondary handler
                         026E          ;
                         026E          ;-
                         026E                  .LIST ME
                         026E          STP17:
        0004'CF    11    D0  026E                          MOVL    #17,W^CURRENT_TC
                   00    DD  0273                          PUSHL   #0
        096E'CF    01    FB  0275                          CALLS   #1,W^REG_SAVE
                   00    DD  027A                          PUSHL   #0
                   01    DD  027C                          PUSHL   #PSL$C_EXEC
```

```
          FF72 DF     DF   027E                         PUSHAL  @W^PRE_EXEC_SEC
                 01   DD   0282                         PUSHL   #1
    00000000'GF  04   FB   0284                         CALLS   #4,G^SYS$SETEXV
       00000000'8F    DD   028B                         PUSHL   #SS$_NORMAL
       0DA7'CF   01   FB   0291                         CALLS   #1,W^REG_CHECKNP
                           0296               MODE      FROM,B10                        ; back to user mode
       0E41'CF   00   FB   0297               CALLS     #0,W^ERLBUF_DUMP                ; dump any errors
                           029C         ;+
                           029C         ;
                           029C         ; test kernel mode
                           029C         ;
                           029C         ;-
                           029C               NEXT_TEST
                           029C
                           029C         STP18:
       0004'CF   12   D0   029C                         MOVL    #18,W^CURRENT_TC
                 00   DD   02A1                         PUSHL   #0
       096E'CF   01   FB   02A3                         CALLS   #1,W^REG_SAVE
   0069'CF   0196'CF  DE   02A8               MOVAL     W^KM,W^MODE                     ; set the mode
                           02AF               MODE TO,C10,KRNL,NOREGS                   ; get into kernal mode
                           02CC               SEVT      KERNEL,NO,                      ; do the kernal mode tests
                           02CC
                           02CC         ;+
                           02CC         ;
                           02CC         ; The next section will declare 2 KERNEL exception handlers.
                           02CC         ; A primary and secondary handler will be
                           02CC         ; set using the $SETEXV_G system service.
                           02CC         ;
                           02CC         ;-
                 55   D4   02CC                         CLRL    R5                      ; set init. handler type
       0079'CF   00   D0   02CE                         MOVL    #PSL$C_KERNEL,W^SET+SETEXV$_ACMODE ; set access KERNEL
                           02D3         30005$:
                 00   DD   02D3                         PUSHL   #0                      ; push dummy parameter
       096E'CF   01   FB   02D5                         CALLS   #1,W^REG_SAVE           ; save register snapshot
       0071'CF   55   D0   02DA                         MOVL    R5,W^SET+SETEXV$_VECTOR ; set vector type
   007D'CF   031C'CF45  DE   02DF                       MOVAL   W^PRE_KERNEL_PRI[R5],-
   0075'CF   0314'CF45  D0   02E7                       MOVL    W^KERNEL_HANTAB[R5],-
  00000000'GF   006D'CF FA   02EF                       CALLG   W^SET,G^SYS$SETEXV
       00000000'8F    DD   02F8                         PUSHL   #SS$_NORMAL
       0DA7'CF   01   FB   02FE                         CALLS   #1,W^REG_CHECKNP
            0004'CF   D6   0303                         INCL    W^CURRENT_TC            ; increment step number
        C8 55    01   F3   0307                         AOBLEQ  #1,R5,30005$            ; do all 1 types
                           030B               .LIST ME
                           030B         ;+
                           030B         ;
                           030B         ; An exception will now be caused to check the handlers.
                           030B         ;
                           030B         ;-
       0127'CF   02   88   030B                         BISB2   #2,W^FLAG1              ; set excep. should occur
                 00   BF   0310                         CHMU    #0                      ; cause an exception
                 5E   11   0312                         BRB     KERNEL_END              ; go on
                           0314         KERNEL_HANTAB:
             00000324'     0314                         .ADDRESS KERNEL_PRIM            ; handler address table
             00000345'     0318                         .ADDRESS KERNEL_SEC
                           031C         PRE_KERNEL_PRI:
             00000000      031C                         .LONG   0                       ; previous handler table
                           0320         PRE_KERNEL_SEC:
```

```
            00000000  0320                 .LONG   0
                      0324          .LIST ME
                      0324     ;+
                      0324     ;
                      0324     ; test the KERNEL primary exception handler
                      0324     ;
                      0324     ;-
                      0324     KERNEL_PRIM:
                0000  0324                 .WORD   0
                      0326          .LIST ME
                      0326     STP20:
        0004'CF   14   D0  0326                 MOVL    #20,W^CURRENT_TC
                  00   DD  032B                 PUSHL   #0
        096E'CF   01   FB  032D                 CALLS   #1,W^REG_SAVE
           0127'CF     96  0332                 INCB    W^FLAG1                 ; set excep. did occur
              0CBA     30  0336                 BSBW    EXCEP_CHECKNP
50    00000000'8F      D0  0339                 MOVL    #SS$_RESIGNAL,R0        ; and resignal
           0127'CF     97  0340                 DECB    W^FLAG1                 ; reset excep. did occur
                       04  0344                 RET
                      0345          .LIST ME
                      0345     ;+
                      0345     ;
                      0345     ; test the KERNEL secondary handler and clean up the exception
                      0345     ;
                      0345     ;-
                      0345     KERNEL_SEC:
                0000  0345                 .WORD   0
                      0347          .LIST ME
                      0347     STP21:
        0004'CF   15   D0  0347                 MOVL    #21,W^CURRENT_TC
                  00   DD  034C                 PUSHL   #0
        096E'CF   01   FB  034E                 CALLS   #1,W^REG_SAVE
           0127'CF     96  0353                 INCB    W^FLAG1                 ; set excep. did occur
              0C99     30  0357                 BSBW EXCEP_CHECKNP
                  00   DD  035A                 PUSHL   #0
                  00   DD  035C                 PUSHL   #0
     00000000'GF  02   FB  035E                 CALLS   #2,G^SYS$UNWIND
50    00000000'8F      D0  0365                 MOVL    #SS$_CONTINUE,R0        ; and resignal
        0127'CF   03   8A  036C                 BICB2   #3,W^FLAG1              ; clear excep. did & should FLAG1
                       04  0371                 RET
                      0372          .LIST ME
                      0372     ;+
                      0372     ;
                      0372     ; the KERNEL last chance handler can not be tested because
                      0372     ; it will always force an exit from the process.
                      0372     ;
                      0372     ; reset the KERNEL primary handler
                      0372     ;
                      0372     ;-
                      0372     KERNEL_END:
                      0372          .LIST ME
                      0372     STP22:
        0004'CF   16   D0  0372                 MOVL    #22,W^CURRENT_TC
                  00   DD  0377                 PUSHL   #0
        096E'CF   01   FB  0379                 CALLS   #1,W^REG_SAVE
                  00   DD  037E                 PUSHL   #0
                  00   DD  0380                 PUSHL   #PSL$C_KERNEL
              FF96 DF    DF  0382                               PUSHAL  @W^PRE_KERNEL_PRI
                   00    DD  0386                               PUSHL   #0
      00000000'GF  04    FB  0388                               CALLS   #4,G^SYS$SETEXV
        00000000'8F      DD  038F                               PUSHL   #SS$_NORMAL
          0DA7'CF  01    FB  0395                               CALLS   #1,W^REG_CHECKNP
                             039A
                             039A           ;+
                             039A           ;
                             039A           ; reset the KERNEL secondary handler
                             039A           ;
                             039A           ;-
                             039A                    .LIST ME
                             039A           STP23:
          0004'CF  17    D0  039A                               MOVL    #23,W^CURRENT_TC
                   00    DD  039F                               PUSHL   #0
          096E'CF  01    FB  03A1                               CALLS   #1,W^REG_SAVE
                   00    DD  03A6                               PUSHL   #0
                   00    DD  03A8                               PUSHL   #PSL$C_KERNEL
              FF72 DF    DF  03AA                               PUSHAL  @W^PRE_KERNEL_SEC
                   01    DD  03AE                               PUSHL   #1
      00000000'GF  04    FB  03B0                               CALLS   #4,G^SYS$SETEXV
        00000000'8F      DD  03B7                               PUSHL   #SS$_NORMAL
          0DA7'CF  01    FB  03BD                               CALLS   #1,W^REG_CHECKNP
                             03C2    427             MODE       FROM,C10                            ; back to user mode
          0E41'CF  00    FB  03C3    428             CALLS      #0,W^ERLBUF_DUMP                    ; dump any errors
```

```
                             03C8             .SBTTL  SETSFM TESTS
                             03C8       ;+
                             03C8       ;
                             03C8       ; $SETSFM tests
                             03C8       ;
                             03C8       ; test _S disable mode
                             03C8       ;
                             03C8       ;-
                             03C8             NEXT_TEST
                             03C8
                             03C8       STP24:
       0004'CF    18    D0   03C8                     MOVL    #24,W^CURRENT_TC
                  00    DD   03CD                     PUSHL   #0
       096E'CF    01    FB   03CF                     CALLS   #1,W^REG_SAVE
  0069'CF   016C'CF     DE   03D4             MOVAL   W^UM,W^MODE               ; set mode
            0127'CF     94   03DB             CLRB    W^FLAG1                   ; clear flag bits
  010F'CF   0038'CF     DE   03DF             MOVAL   W^SETSFM,W^SERV_NAME      ; set service name
       6D   040E'CF     DE   03E6             MOVAL   W^NOT_ENABLED,(FP)        ; set handler address
                             03E9             $SETSFM_S #0                      ; disable failure mode
                             03F4             FAIL_CHECK SS$_WASCLR             ; check success
        00000000'8F     DD   03F4                     PUSHL   #SS$_WASCLR
       0978'CF    01    FB   03FA                     CALLS   #1,W^REG_CHECK
                             03FF       ;
                             03FF       ; make sure that it's really disabled by forcing an error
                             03FF       ;
                             03FF             $CLREF_S #2000                    ; force an error
                  08    11   040C             BRB     A10                       ; if you got here we're OK
                             040E       NOT_ENABLED:
                      0000   040E             .WORD   0                         ; enter here if illegally enabled
       0F4F'CF    00    FB   0410             CALLS   #0,W^EXCEP_FAIL           ; print a failure message
                        04   0415             RET                               ; go on
                             0416       A10:
                             0416       ;+
                             0416       ;
                             0416       ; test _S & enable
                             0416       ;
                             0416       ;-
                             0416             NEXT_TEST
                             0416
                             0416       STP25:
       0004'CF    19    D0   0416                     MOVL    #25,W^CURRENT_TC
                  00    DD   041B                     PUSHL   #0
       096E'CF    01    FB   041D                     CALLS   #1,W^REG_SAVE
       6D   045A'CF     DE   0422             MOVAL   W^ENABLED,(FP)            ; set the handler address
                             0427             $SETSFM_S #1                      ; test _S & enable mode
                             0430             FAIL_CHECK SS$_WASCLR             ; check success
        00000000'8F     DD   0430                     PUSHL   #SS$_WASCLR
       0978'CF    01    FB   0436                     CALLS   #1,W^REG_CHECK
                             043B       ;
                             043B       ; make sure that its really enabled by forcing an error
                             043B       ;
       0127'CF    02    88   043B             BISB2   #2,W^FLAG1                ; set expecting exception flag
                             0440             $CLREF_S #2000                    ; force an error
    2E 0127'CF    00    E4   044D             BBSC    #0,W^FLAG1,A30            ; check the exception flag and clear it if s
       0F4F'CF    00    FB   0453             CALLS   #0,W^EXCEP_FAIL           ; print exception failure if not set
                  27    11   0458             BRB     A30                       ; get to the next test
                             045A       ENABLED:
```

D 1

```
                0004  045A          .WORD   ^M<R2>                  ; enter here if OK
      0127'CF     96  045C          INCB    W^FLAG1                 ; set exception occured flag
   52    08 AC    D0  0460          MOVL    CHF$L_MCHARGLST(AP),R2  ; get mechanism array pointer
         0C A2    D1  0464          CMPL    CHF$L_MCH_SAVR0(R2),-
     00000000'8F      0467                  #SS$_ILLEFC             ; is this the right error?
              12  13  046C          BEQL    A20                     ; br if OK
         0C A2    DD  046E          PUSHL   CHF$L_MCH_SAVR0(R2)     ; push received
     00000000'8F  DD  0471          PUSHL   #SS$_ILLEFC             ; push expected
         01A4'CF  DF  0477          PUSHAL  W^EXP                   ; push string variable
   0E7E'CF    03  FB  047B          CALLS   #3,W^PRINT_FAIL         ; print the failure
                      0480      A20:
                  04  0480          RET                             ; carry on
                      0481      A30:
   0127'CF    01  8A  0481          BICB2   #1,W^FLAG1              ; clear exception occured flag
                      0486      ;+
                      0486      ;
                      0486      ; test _G disable mode
                      0486      ;
                      0486      ;-
                      0486          NEXT_TEST
                      0486
                      0486      STP26:
   0004'CF    1A  D0  0486                  MOVL    #26,W^CURRENT_TC
              00  DD  048B                  PUSHL   #0
   096E'CF    01  FB  048D                  CALLS   #1,W^REG_SAVE
   6D   04BF'CF   DE  0492          MOVAL   W^NOT_ENABLED1,(FP)     ; set handler address
                      0497          $SETSFM_G       W^SET1          ; test _G & disable
                      04A0          FAIL_CHECK SS$_WASSET           ; check for success
     00000000'8F  DD  04A0                  PUSHL   #SS$_WASSET
   0978'CF    01  FB  04A6                  CALLS   #1,W^REG_CHECK
                      04AB      ;
                      04AB      ; make sure that it really is disabled by forcing an error
                      04AB      ;
   0127'CF    02  8A  04AB          BICB2   #2,W^FLAG1              ; clear expecting exception flag
                      04B0          $CLREF_S #2000                  ; force an error
              08  11  04BD          BRB     A40                     ; if we got here we're OK
                      04BF      NOT_ENABLED1:
                0000  04BF          .WORD   0                       ; enter here if illegally enabled
   0F4F'CF    00  FB  04C1          CALLS   #0,W^EXCEP_FAIL         ; print exception failure message
                  04  04C6          RET                             ; carry on
                      04C7      A40:
                      04C7      ;+
                      04C7      ;
                      04C7      ; test _G & enable mode
                      04C7      ;
                      04C7      ;-
                      04C7          NEXT_TEST
                      04C7
                      04C7      STP27:
   0004'CF    1B  D0  04C7                  MOVL    #27,W^CURRENT_TC
              00  DD  04CC                  PUSHL   #0
   096E'CF    01  FB  04CE                  CALLS   #1,W^REG_SAVE
    6D    0E'AF   DE  04D3          MOVAL   B^ENABLED1,(FP)         ; set handler address
       0085'CF    D6  04D7          INCL    W^SET1+SETSFM$_ENBFLG   ; set mode to enable
                      04DB          $SETSFM_G W^SET1                ; test _G & enable
                      04E4          FAIL_CHECK SS$_WASCLR           ; check success
     00000000'8F  DD  04E4                  PUSHL   #SS$_WASCLR
```

```
      0978'CF    01    FB  04EA                  CALLS    #1,W^REG_CHECK
                           04EF      ;
                           04EF      ; make sure it's enabled by forcing an error
                           04EF      ;
      0127'CF    02    88  04EF                  BISB2    #2,W^FLAG1                ; set expecting exception flag
                           04F4                  $CLREF_S #2000                     ; force an error
   2E 0127'CF    00    E4  0501                  BBSC     #0,W^FLAG1,A60            ; br if OK and clear the flag bit
      0F4F'CF    00    FB  0507                  CALLS    #0,W^EXCEP_FAIL           ; otherwise print exception fail message
              27       11  050C                  BRB      A60                       ; bad news if you got here
                           050E      ENABLED1:
                     0004  050E                  .WORD    ^M<R2>                    ; if you are here we're OK
          0127'CF      96  0510                  INCB     W^FLAG1                   ; set exception occured flag
       52    08 AC     D0  0514                  MOVL     CHF$L_MCHARGLST(AP),R2    ; get mechanism array pointer
             0C A2     D1  0518                  CMPL     CHF$L_MCH_SAVR0(R2),-
       00000000'8F         051B                           #SS$_ILLEFC               ; is it the right error message
              13       13  0520                  BEQL     A60                       ; br if good
             0C A2     DD  0522                  PUSHL    CHF$L_MCH_SAVR0(R2)       ; push received
       00000000'8F     DD  0525                  PUSHL    #SS$_ILLEFC               ; set expected
          01A4'CF      DF  052B                  PUSHAL   W^EXP                     ; set string variable
      0E7E'CF    03    FB  052F                  CALLS    #3,W^PRINT_FAIL           ; print the failure
                           0534      A50:
                       04  0534                  RET                                ; carry on
                           0535      A60:
      0127'CF    01    8A  0535                  BICB2    #1,W^FLAG1                ; clear exception occured flag
```

```
                            053A   541          .SBTTL  UNWIND TESTS
                            053A   542  ;+
                            053A   543  ;
                            053A   544  ; $UNWIND tests
                            053A   545  ;
                            053A   546  ; test level 1 _S
                            053A   547  ;
                            053A   548  ;-
                            053A   549          NEXT_TEST
                            053A
                            053A       STP28:
      0004'CF    1C    D0   053A                    MOVL    #28,W^CURRENT_TC
                 00    DD   053F                    PUSHL   #0
      096E'CF    01    FB   0541                    CALLS   #1,W^REG_SAVE
  0069'CF   016C'CF    DE   0546   550          MOVAL   W^UM,W^MODE                 ; set the mode
  010F'CF   003F'CF    DE   054D   551          MOVAL   W^UNWIND,W^SERV_NAME        ; set service name
      0FDA'CF    5E    D0   0554   552          MOVL    SP,W^WORK2                  ; save the stack pointer
      0FDA'CF    04    C2   0559   553          SUBL2   #4,W^WORK2                  ; compensate for BSBW PC+2 word
        62'AF    00    FB   055E   554          CALLS   #0,B^10$                    ; put a call frame on the stack
                            0562   555 10$:
                     0000   0562   556          .WORD   0
        6D    6A'AF    DE   0564   557          MOVAL   B^20$,(FP)                  ; set the handler address
                 00    BF   0568   558          CHMU    #0                          ; cause an exception
                            056A   559 20$:
                     0004   056A   560          .WORD   ^M<R2>
        52    04 AC    D0   056C   561          MOVL    B^CHF$L_SIGARGLST(AP),R2 ; get signal array address
  04 A2   00000000'8F  D1   0570   562          CMPL    #SS$_UNWIND,B^CHF$L_SIG_NAME(R2) ; check the signal name
                 25    13   0578   563          BEQL    25$                         ; br if its an unwind signal
                 00    DD   057A   564          PUSHL   #0                          ; push dummy parameter
      096E'CF    01    FB   057C   565          CALLS   #1,W^REG_SAVE               ; save a register snapshot
                            0581   566          $UNWIND_S DEPADR=DEPTH,NEWPC=30$ ; try level 1 _S
                            0594   567          FAIL_CHECK SS$_NORMAL               ; check success
      00000000'8F      DD   0594                    PUSHL   #SS$_NORMAL
      0978'CF    01    FB   059A                    CALLS   #1,W^REG_CHECK
                            059F   568 25$:
    50    00000000'8F  D0   059F   569          MOVL    #SS$_CONTINUE,R0            ; signal a continue
                       04   05A6   570          RET                                 ; do your magic now
                            05A7   571 30$:
              0A34     30   05A7   572          BSBW    W^STACK_CHECK               ; check the stack
                            05AA   573  ;+
                            05AA   574  ;
                            05AA   575  ; test level 1 _G
                            05AA   576  ;
                            05AA   577  ;-
                            05AA   578          NEXT_TEST
                            05AA
                            05AA       STP29:
      0004'CF    1D    D0   05AA                    MOVL    #29,W^CURRENT_TC
                 00    DD   05AF                    PUSHL   #0
      096E'CF    01    FB   05B1                    CALLS   #1,W^REG_SAVE
  00000091'EF    03'AF DE   05B6   579          MOVAL   B^30$,UNW+UNWIND$_NEWPC ; set new PC
        C2'AF    00    FB   05BE   580          CALLS   #0,B^10$                    ; put a call frame on the stack
                            05C2   581 10$:
                     0000   05C2   582          .WORD   0
        6D    CA'AF    DE   05C4   583          MOVAL   B^20$,(FP)                  ; set the handler address
                 00    BF   05C8   584          CHMU    #0                          ; cause an exception
                            05CA   585 20$:
```

```
                          0004  05CA              .WORD    ^M<R2>
              52    04 AC   D0  05CC              MOVL     B^CHF$L_SIGARGLST(AP),R2 ; get signal array address
     04 A2   00000000'8F    D1  05D0              CMPL     #SS$_UNWIND,B^CHF$L_SIG_NAME(R2) ; check the signal name
                     21     13  05D8              BEQL     25$                      ; br if its an unwind signal
       0091'CF    03'AF     DE  05DA              MOVAL    B^30$,W^UNW+UNWIND$_NEWPC ; set the return PC
                     00     DD  05E0              PUSHL    #0                       ; push a dummy parameter
          096E'CF    01     FB  05E2              CALLS    #1,W^REG_SAVE            ; save a register snapshot
                                05E7              $UNWIND_G W^UNW                   ; try level 1 _G
                                05F0              FAIL_CHECK SS$_NORMAL             ; check for success
            00000000'8F     DD  05F0                       PUSHL    #SS$_NORMAL
          0978'CF    01     FB  05F6                       CALLS    #1,W^REG_CHECK
                                05FB         25$:
      50    00000000'8F     D0  05FB              MOVL     #SS$_CONTINUE,R0         ; signal a continue
                            04  0602              RET                               ; do your magic now
                                0603         30$:
                09D8        30  0603              BSBW     W^STACK_CHECK            ; check the stack
                                0606         ;+
                                0606         ;
                                0606         ; test level 2 _S
                                0606         ;
                                0606         ;-
                                0606              NEXT_TEST
                                0606
                                0606         STP30:
          0004'CF    1E     D0  0606                       MOVL     #30,W^CURRENT_TC
                     00     DD  060B                       PUSHL    #0
          096E'CF    01     FB  060D                       CALLS    #1,W^REG_SAVE
          0128'CF    02     D0  0612              MOVL     #2,W^DEPTH               ; set the depth
             1B'AF   00     FB  0617              CALLS    #0,B^5$                  ; put a call frame on the stack
                                061B         5$:
                          0000  061B              .WORD    0
             21'AF   00     FB  061D              CALLS    #0,B^10$                 ; put a call frame on the stack
                                0621         10$:
                          0000  0621              .WORD    0
              5D    29'AF   DE  0623              MOVAL    B^20$,(FP)               ; set the handler address
                     00     BF  0627              CHMU     #0                       ; cause an exception
                                0629         20$:
                          0004  0629              .WORD    ^M<R2>
              52    04 AC   D0  062B              MOVL     B^CHF$L_SIGARGLST(AP),R2 ; get signal array address
     04 A2   00000000'8F    D1  062F              CMPL     #SS$_UNWIND,B^CHF$L_SIG_NAME(R2) ; check the signal name
                     25     13  0637              BEQL     25$                      ; br if its an unwind signal
                     00     DD  0639              PUSHL    #0                       ; push a dummy parameter
          096E'CF    01     FB  063B              CALLS    #1,W^REG_SAVE            ; save a register snapshot
                                0640              $UNWIND_S DEPADR=DEPTH,NEWPC=30$ ; try level 1 _S
                                0653              FAIL_CHECK SS$_NORMAL             ; check success
            00000000'8F     DD  0653                       PUSHL    #SS$_NORMAL
          0978'CF    01     FB  0659                       CALLS    #1,W^REG_CHECK
                                065E         25$:
      50    00000000'8F     D0  065E              MOVL     #SS$_CONTINUE,R0         ; signal a continue
                            04  0665              RET
                                0666         30$:
                0975        30  0666              BSBW     W^STACK_CHECK            ; check the stack
                                0669         ;+
                                0669         ;
                                0669         ; test level 2 _G
                                0669         ;
                                0669         ;-
```

```
                              0669   634          NEXT_TEST
                              0669
                              0669       STP31:
        0004'CF    1F    D0   0669                          MOVL    #31,W^CURRENT_TC
                   00    DD   066E                          PUSHL   #0
        096E'CF    01    FB   0670                          CALLS   #1,W^REG_SAVE
   0091'CF    C6'AF      DE   0675   635          MOVAL   B^30$,W^UNW+UNWIND$_NEWPC ; set the new PC
         7F'AF     00    FB   067B   636          CALLS   #0,B^5$                   ; put a stack frame on the stack
                              067F   637 5$:
                 0000         067F   638          .WORD   0
         85'AF     00    FB   0681   639          CALLS   #0,B^10$                  ; put a call frame on the stack
                              0685   640 10$:
                 0000         0685   641          .WORD   0
         6D   8D'AF      DE   0687   642          MOVAL   B^20$,(FP)                ; set the handler address
                   00    BF   068B   643          CHMU    #0                        ; cause an exception
                              068D   644 20$:
                 0004         068D   645          .WORD   ^M<R2>
         52   04 AC      D0   068F   646          MOVL    B^CHF$L_SIGARGLST(AP),R2 ; get signal array address
   04 A2   00000000'8F   D1   0693   647          CMPL    #SS$_UNWIND,B^CHF$L_SIG_NAME(R2) ; check the signal name
                   21    13   069B   648          BEQL    25$                       ; br if its an unwind signal
   0091'CF    C6'AF      DE   069D   649          MOVAL   B^30$,W^UNW+UNWIND$_NEWPC ; set the return PC
                   00    DD   06A3   650          PUSHL   #0                        ; push a dummy parameter
        096E'CF    01    FB   06A5   651          CALLS   #1,W^REG_SAVE             ; save a register snapshot
                              06AA   652          $UNWIND_G W^UNW                   ; try level 1 _G
                              06B3   653          FAIL_CHECK SS$_NORMAL             ; check for success
        00000000'8F      DD   06B3                          PUSHL   #SS$_NORMAL
        0978'CF    01    FB   06B9                          CALLS   #1,W^REG_CHECK
                              06BE   654 25$:
   50   00000000'8F      D0   06BE   655          MOVL    #SS$_CONTINUE,R0          ; set continue
                         04   06C5   656          RET
                              06C6   657 30$:
            0915   30         06C6   658          BSBW    W^STACK_CHECK             ; check the stack
                              06C9   659 ;+
                              06C9   660 ;
                              06C9   661 ; test level 3 _S
                              06C9   662 ;
                              06C9   663 ;-
                              06C9   664          NEXT_TEST
                              06C9
                              06C9       STP32:
        0004'CF    20    D0   06C9                          MOVL    #32,W^CURRENT_TC
                   00    DD   06CE                          PUSHL   #0
        096E'CF    01    FB   06D0                          CALLS   #1,W^REG_SAVE
        0128'CF    03    D0   06D5   665          MOVL    S^#3,W^DEPTH              ; set the depth
         DE'AF     00    FB   06DA   666          CALLS   #0,B^4$                   ; put a frame on the stack
                              06DE   667 4$:
                 0000         06DE   668          .WORD   0
         E4'AF     00    FB   06E0   669          CALLS   #0,B^8$                   ; and an other
                              06E4   670 8$:
                 0000         06E4   671          .WORD   0
         EA'AF     00    FB   06E6   672          CALLS   #0,B^10$                  ; put a call frame on the stack
                              06EA   673 10$:
                 0000         06EA   674          .WORD   0
         6D   F2'AF      DE   06EC   675          MOVAL   B^20$,(FP)                ; set the handler address
                   00    BF   06F0   676          CHMU    #0                        ; cause an exception
                              06F2   677 20$:
                 0004         06F2   678          .WORD   ^M<R2>
        52    04 AC   D0 06F4          MOVL    B^CHF$L_SIGARGLST(AP),R2 ; get signal array address
  04 A2   00000000'8F   D1 06F8          CMPL    #SS$_UNWIND,B^CHF$L_SIG_NAME(R2) ; check the signal name
                 25   13 0700          BEQL    25$                      ; br if its an unwind signal
                 00   DD 0702          PUSHL   #0                       ; push a dummy parameter
        096E'CF  01   FB 0704          CALLS   #1,W^REG_SAVE            ; save a register snapshot
                         0709          $UNWIND_S DEPADR=DEPTH,NEWPC=30$ ; try level 1 _S
                         071C          FAIL_CHECK SS$_NORMAL            ; check success
         00000000'8F  DD 071C                  PUSHL   #SS$_NORMAL
        0978'CF  01   FB 0722                  CALLS   #1,W^REG_CHECK
                         0727      25$:
      50   00000000'8F  D0 0727          MOVL    #SS$_CONTINUE,R0         ; set continue
                 04      072E          RET
                         072F      30$:
             08AC  30    072F          BSBW    W^STACK_CHECK            ; check the stack
                         0732      ;+
                         0732      ;
                         0732      ; test level 3 with _G
                         0732      ;
                         0732      ;-
                         0732          NEXT_TEST
                         0732
                         0732      STP33:
        0004'CF  21   D0 0732                  MOVL    #33,W^CURRENT_TC
                 00   DD 0737                  PUSHL   #0
        096E'CF  01   FB 0739                  CALLS   #1,W^REG_SAVE
     0091'CF    95'AF   DE 073E          MOVAL   B^30$,W^UNW+UNWIND$_NEWPC ; set the new PC
          48'AF  00   FB 0744          CALLS   #0,B^4$                  ; put a frame on the stack
                         0748      4$:
                0000     0748          .WORD   0
          4E'AF  00   FB 074A          CALLS   #0,B^8$                  ; and another
                         074E      8$:
                0000     074E          .WORD   0
          54'AF  00   FB 0750          CALLS   #0,B^10$                 ; put a call frame on the stack
                         0754      10$:
                0000     0754          .WORD   0
        6D    5C'AF   DE 0756          MOVAL   B^20$,(FP)               ; set the handler address
                 00   BF 075A          CHMU    #0                       ; cause an exception
                         075C      20$:
                0004     075C          .WORD   ^M<R2>
        52    04 AC   D0 075E          MOVL    B^CHF$L_SIGARGLST(AP),R2 ; get signal array address
  04 A2   00000000'8F   D1 0762          CMPL    #SS$_UNWIND,B^CHF$L_SIG_NAME(R2) ; check the signal name
                 21   13 076A          BEQL    25$                      ; br if its an unwind signal
     0091'CF    95'AF   DE 076C          MOVAL   B^30$,W^UNW+UNWIND$_NEWPC ; set the return PC
                 00   DD 0772          PUSHL   #0                       ; push a dummy parameter
        096E'CF  01   FB 0774          CALLS   #1,W^REG_SAVE            ; save a register snapshot
                         0779          $UNWIND_G W^UNW                  ; try level 1 _G
                         0782          FAIL_CHECK SS$_NORMAL            ; check for success
         00000000'8F  DD 0782                  PUSHL   #SS$_NORMAL
        0978'CF  01   FB 0788                  CALLS   #1,W^REG_CHECK
                         078D      25$:
      50   00000000'8F  D0 078D          MOVL    #SS$_CONTINUE,R0         ; set continue
                 04      0794          RET
                         0795      30$:
             0846  30    0795          BSBW    W^STACK_CHECK            ; check the stack
                         0798          TEST_END
            004C'CF   DD 0798                  PUSHL   W^TMD_ADDR
            0048'CF   DD 079C                  PUSHL   W^TMN_ADDR
```

```
                         02   DD  07A0                   PUSHL   #2
                    0044'CF   DD  07A2                   PUSHL   W^MOD_MSG_CODE
            00000000'GF   04  FB  07A6                   CALLS   #$$T1,G^LIB$SIGNAL
      0044'CF   01   1C   01  F0  07AD                   INSV    #1,#STS$V_INHIB_MSG,#1,W^MOD_MSG_CODE
                    0044'CF   DD  07B4                   PUSHL   W^MOD_MSG_CODE
            00000000'GF   01  FB  07B8                   CALLS   #1,G^SYS$EXIT
```

```
07BF          .SBTTL SETUP_SUPER ROUTINE
07BF    ;++
07BF    ; FUNCTIONAL DESCRIPTION:
07BF    ;       Routine to declare an initial CHMS handler from user mode.
07BF    ;
07BF    ; CALLING SEQUENCE:
07BF    ;       $CMKRNL_S W^SETUP_SUPER,ARGLST
07BF    ;
07BF    ;               ARGLST = address of a pointer to a one parameter argument list conta
07BF    ;                        the address of the entry mask of the CHMS handler
07BF    ;
07BF    ; INPUT PARAMETERS:
07BF    ;       ARGLST
07BF    ;
07BF    ; IMPLICIT INPUTS
07BF    ;       NONE
07BF    ;
07BF    ; OUTPUT PARAMETERS:
07BF    ;       Declares a change mode handler for super mode which must be
07BF    ;       reset to DCL in the users handler routine when the handler is
07BF    ;       no longer needed.
07BF    ;
07BF    ; IMPLICIT OUTPUTS:
07BF    ;       NONE
07BF    ;
07BF    ; COMPLETION CODES:
07BF    ;       NONE
07BF    ;
07BF    ; SIDE EFFECTS:
07BF    ;       SERV_NAME is left containing a pointer to DCLCMH
07BF    ;
07BF    ; ON ENTRY:
07BF    ;                       ---------            ---------
07BF    ;               KSP => !    0    !    USP => !       !
07BF    ;                      !    0    !           ! USER  !
07BF    ;                      !   AP    !           !       !
07BF    ;                      !   FP    !           ! CALL  !
07BF    ;                      !   PC    !           !       !
07BF    ;                      !    0    !           ! FRAME !
07BF    ;                      !    0    !           !       !
07BF    ;                      !   AP    !           ---------
07BF    ;                      !   FP    !
07BF    ;                      !SRVEXIT  !
07BF    ;                      !   PC    !
07BF    ;                      !  PSL    !
07BF    ;                      ---------
07BF    ;--
```

```
                                07BF   RETURN_PC:
                    00000000    07BF            .LONG   0                               ; storage for user return PC
                                07C3   HANDLER_PC:
                    00000000    07C3            .LONG   0                               ; storage for handler PC
                                07C7   ;
                                07C7   SETUP_SUPER:
                        000C    07C7            .WORD   ^M<R2,R3>
            53    03      DB    07C9            MFPR    #PR$_USP,R3                     ; get the user call frame address
      EE AF    10 A3      D0    07CC            MOVL    SF$L_SAVE_PC(R3),B^RETURN_PC ; get the user return PC
      ED AF    04 AC      D0    07D1            MOVL    4(AP),HANDLER_PC                ; save the handler address
         52    0C AD      D0    07D6            MOVL    SF$L_SAVE_FP(FP),R2             ; get saved FP
            52    00'     C0    07DA            ADDL    S^#EXE$C_CMSTKSZ,R2             ; back over change mode stack frame
         62    EB'AF      9E    07DD            MOVAB   B^20$,(R2)                      ; set return address
                  0A      F0    07E1            INSV    #<<PSL$C_SUPER@PSL$S_CURMOD>+PSL$C_SUPER>,-
                  16            07E3                    #PSL$V_PRVMOD,-
         04 A2    04            07E4                    #PSL$S_CURMOD*2,4(R2)           ; set current and previous mode to super
            50    00'     D0    07E7            MOVL    S^#SS$_NORMAL,R0                ; set correct return code
                          04    07EA            RET                                     ; enter super mode
                                07EB   20$:
                  7E      D4    07EB            CLRL    -(SP)                           ; set up dummy PSL
         F1'AF    6E      FA    07ED            CALLG   (SP),B^30$                      ; create initial call frame
                                07F1   30$:
                        0000    07F1            .WORD   ^M<>                            ; entry mask
                  00      DD    07F3            PUSHL   #0                              ; push a dummy parameter
      096E'CF     01      FB    07F5            CALLS   #1,W^REG_SAVE                   ; save the registers
  0069'CF    0178'CF      DE    07FA            MOVAL   W^SM,W^MODE                     ; set the mode
  010F'CF    0831'CF      DE    0801            MOVAL   W^DCLCMH,W^SERV_NAME            ; set service name
                                0808            $DCLCMH_S @HANDLER_PC,W^PRVHND1,#0 ; set real handler
                                0818            FAIL_CHECKNP SS$_NORMAL                 ; check for success
       00000000'8F        DD    0818                    PUSHL   #SS$_NORMAL
      0DA7'CF     01      FB    081E                    CALLS   #1,W^REG_CHECKNP
       03C00000 8F        DD    0823            PUSHL   #<<PSL$C_USER@PSL$V_CURMOD>-
                                0829                    !<PSL$C_USER@PSL$V_PRVMOD>>; set return to user
            93 AF         DD    0829            PUSHL   RETURN_PC                       ; set the return PC
                          02    082C            REI                                     ; return to user mode
```

```
                               082D   809 .SBTTL SUPER_MODE
                               082D   810 ;++
                               082D   811 ; FUNCTIONAL DESCRIPTION:
                               082D   812 ;       Routine to handle the CHMS instructions.
                               082D   813 ;
                               082D   814 ; CALLING SEQUENCE:
                               082D   815 ;       CHMS    #N
                               082D   816 ;
                               082D   817 ; INPUT PARAMETERS:
                               082D   818 ;           SP=>  CHMS parameter
                               082D   819 ;                 PC
                               082D   820 ;                 PSL
                               082D   821 ;
                               082D   822 ;           The CHMS parameter can be one of the following:
                               082D   823 ;
                               082D   824 ;                 1 = execute the $SETEXV tests
                               082D   825 ;                 2 = execute a $DCLCMH_S to reset the CHMS handler
                               082D   826 ;
                               082D   827 ; OUTPUT PARAMETERS:
                               082D   828 ;       NONE
                               082D   829 ;--
                               082D   830
                               082D   831 WORK:
                    00000000   082D   832         .LONG   0                       ; scratch storage
                               0831   833 DCLCMH:
       48 4D 43 4C 43 44 00'   0831   834         .ASCIC  /DCLCMH/                ; service name
                          06   0831
                               0838   835 SUPER_MODE:
              50   8E     D0   0838   836         MOVL    (SP)+,R0                ; get CHM parameter off the stack
        02    01   50     8F   083B   837         CASEB   R0,#1,#2                ; do the right thing
                               083F   838 10$:
                       0004'   083F   839         .WORD   20$-10$
                       00FD'   0841   840         .WORD   B30-10$
                               0843   841 20$:
                    00000006   0843   842         STEP=6
                               0843   843         SEVT    SUPER,YES               ; do the super tests
                               0843
                               0843           ;+
                               0843           ;
                               0843           ; The next section will declare 2 SUPER exception handlers.
                               0843           ; A primary and secondary handler will be
                               0843           ; set using the $SETEXV_G system service.
                               0843           ;
                               0843           ;-
                   55     D4   0843                   CLRL    R5                      ; set init. handler type
      0079'CF      02     D0   0845                   MOVL    #PSL$C_SUPER,W^SET+SETEXV$_ACMODE ; set access SUPER
                               084A           30007$:
                   00     DD   084A                   PUSHL   #0                      ; push dummy parameter
      096E'CF      01     FB   084C                   CALLS   #1,W^REG_SAVE           ; save register snapshot
      0071'CF      55     D0   0851                   MOVL    R5,W^SET+SETEXV$_VECTOR ; set vector type
  007D'CF    0893'CF45    DE   0856                   MOVAL   W^PRE_SUPER_PRI[R5],-
  0075'CF    088B'CF45    D0   085E                   MOVL    W^SUPER_HANTAB[R5],-
00000000'GF    006D'CF    FA   0866                   CALLG   W^SET,G^SYS$SETEXV
         00000000'8F      DD   086F                   PUSHL   #SS$_NORMAL
      0978'CF      01     FB   0875                   CALLS   #1,W^REG_CHECK
         0004'CF          D6   087A                   INCL    W^CURRENT_TC            ; increment step number
        C8 55      01     F3   087E                   AOBLEQ  #1,R5,30007$            ; do all 1 types
```

```
                        0882            .LIST ME
                        0882    ;+
                        0882    ;
                        0882    ; An exception will now be caused to check the handlers.
                        0882    ;
                        0882    ;-
    0127'CF   02   88   0882                    BISB2   #2,W^FLAG1              ; set excep. should occur
              00   BF   0887                    CHMU    #0                      ; cause an exception
              5E   11   0889                    BRB     SUPER_END               ; go on
                        088B    SUPER_HANTAB:
          0000089B'     088B                    .ADDRESS SUPER_PRIM             ; handler address table
          000008BC'     088F                    .ADDRESS SUPER_SEC
                        0893    PRE_SUPER_PRI:
           00000000     0893                    .LONG   0                       ; previous handler table
                        0897    PRE_SUPER_SEC:
           00000000     0897                    .LONG   0
                        089B            .LIST ME
                        089B    ;+
                        089B    ;
                        089B    ; test the SUPER primary exception handler
                        089B    ;
                        089B    ;-
                        089B    SUPER_PRIM:
               0000     089B                    .WORD   0
                        089D            .LIST ME
                        089D    STP8:
    0004'CF   08   D0   089D                    MOVL    #8,W^CURRENT_TC
              00   DD   08A2                    PUSHL   #0
    096E'CF   01   FB   08A4                    CALLS   #1,W^REG_SAVE
       0127'CF     96   08A9                    INCB    W^FLAG1                 ; set excep. did occur
          0701     30   08AD                    BSBW    EXCEP_CHECK             ; check primary handler
50   00000000'8F   D0   08B0                    MOVL    #SS$_RESIGNAL,R0        ; and resignal
       0127'CF     97   08B7                    DECB    W^FLAG1                 ; reset excep. did occur
                   04   08BB                    RET
                        08BC            .LIST ME
                        08BC    ;+
                        08BC    ;
                        08BC    ; test the SUPER secondary handler and clean up the exception
                        08BC    ;
                        08BC    ;-
                        08BC    SUPER_SEC:
               0000     08BC                    .WORD   0
                        08BE            .LIST ME
                        08BE    STP9:
    0004'CF   09   D0   08BE                    MOVL    #9,W^CURRENT_TC
              00   DD   08C3                    PUSHL   #0
    096E'CF   01   FB   08C5                    CALLS   #1,W^REG_SAVE
       0127'CF     96   08CA                    INCB    W^FLAG1                 ; set excep. did occur
          06E0     30   08CE                    BSBW    EXCEP_CHECK             ; check secondary handler
              00   DD   08D1                    PUSHL   #0
              00   DD   08D3                    PUSHL   #0
00000000'GF   02   FB   08D5                    CALLS   #2,G^SYS$UNWIND
50   00000000'8F   D0   08DC                    MOVL    #SS$_CONTINUE,R0        ; and resignal
    0127'CF   03   8A   08E3                    BICB2   #3,W^FLAG1              ; clear excep. did & should FLAG1
                   04   08E8                    RET
                        08E9            .LIST ME
                        08E9    ;+
                               08E9        ;
                               08E9        ; the SUPER last chance handler can not be tested because
                               08E9        ; it will always force an exit from the process.
                               08E9        ;
                               08E9        ; reset the SUPER primary handler
                               08E9        ;
                               08E9        ;-
                               08E9        SUPER_END:
                               08E9                .LIST ME
                               08E9        STP10:
       0004'CF    0A    D0     08E9                          MOVL     #10,W^CURRENT_TC
                  00    DD     08EE                          PUSHL    #0
       096E'CF    01    FB     08F0                          CALLS    #1,W^REG_SAVE
                  00    DD     08F5                          PUSHL    #0
                  02    DD     08F7                          PUSHL    #PSL$C_SUPER
            FF96 DF     DF     08F9                          PUSHAL   @W^PRE_SUPER_PRI
                  00    DD     08FD                          PUSHL    #0
   00000000'GF    04    FB     08FF                          CALLS    #4,G^SYS$SETEXV
     00000000'8F        DD     0906                          PUSHL    #SS$_NORMAL
       0978'CF    01    FB     090C                          CALLS    #1,W^REG_CHECK
                               0911
                               0911        ;+
                               0911        ;
                               0911        ; reset the SUPER secondary handler
                               0911        ;
                               0911        ;-
                               0911                .LIST ME
                               0911        STP11:
       0004'CF    0B    D0     0911                          MOVL     #11,W^CURRENT_TC
                  00    DD     0916                          PUSHL    #0
       096E'CF    01    FB     0918                          CALLS    #1,W^REG_SAVE
                  00    DD     091D                          PUSHL    #0
                  02    DD     091F                          PUSHL    #PSL$C_SUPER
            FF72 DF     DF     0921                          PUSHAL   @W^PRE_SUPER_SEC
                  01    DD     0925                          PUSHL    #1
   00000000'GF    04    FB     0927                          CALLS    #4,G^SYS$SETEXV
     00000000'8F        DD     092E                          PUSHL    #SS$_NORMAL
       0978'CF    01    FB     0934                          CALLS    #1,W^REG_CHECK
            0031  31    0939                BRW     B70                                 ; carry on
                        093C           B30:
FEEA CF   010F'CF  DE   093C                MOVAL   W^SERV_NAME,W^WORK                  ; save previous service name
010F'CF   FEEA CF  DE   0943                MOVAL   W^DCLCMH,W^SERV_NAME                ; set temp service name
                        094A                $DCLCMH_S @PRVHND1,,#0                      ; reset the CHMS handler to DCL
                        095B                FAIL_CHECK SS$_NORMAL                       ; check for success
     00000000'8F        DD     095B                          PUSHL    #SS$_NORMAL
       0978'CF    01    FB     0961                          CALLS    #1,W^REG_CHECK
010F'CF   FEC3 CF  D0   0966                MOVL    W^WORK,W^SERV_NAME                  ; reset to the previous service name
                        096D           B70:
                  02    096D                REI                                         ; go back to user mode
```

```
                                 096E             .SBTTL REG_SAVE
                                 096E       ;++
                                 096E       ; FUNCTIONAL DESCRIPTION:
                                 096E       ;       Subroutine to save R2-R11 in the register save location.
                                 096E       ;
                                 096E       ; CALLING SEQUENCE:
                                 096E       ;       PUSHL   #0              ; save a dummy parameter
                                 096E       ;       CALLS   #1,W^REG_SAVE   ; save R2-R11
                                 096E       ;
                                 096E       ; INPUT PARAMETERS:
                                 096E       ;       NONE
                                 096E       ;
                                 096E       ; OUTPUT PARAMETERS:
                                 096E       ;       NONE
                                 096E       ;
                                 096E       ;--
                                 096E
                                 096E       REG_SAVE:
                           0FFC  096E             .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
      0008'CF   14 AD   28   28  0970             MOVC3   #4*10,^X14(FP),W^REG_SAVE_AREA  ; save the registers in the program
                             04  0977             RET
                                 0978             .SBTTL  REG_CHECK
                                 0978       ;++
                                 0978       ; FUNCTIONAL DESCRIPTION:
                                 0978       ;       Subroutine to test R0 & R2-R11 for proper content after a service
                                 0978       ;       execution. A snapshot is taken by the REG_SAVE routine at the
                                 0978       ;       beginning of each step and this routine is executed after the
                                 0978       ;       services have been executed.
                                 0978       ;
                                 0978       ; CALLING SEQUENCE:
                                 0978       ;       PUSHL   #SS$_XXXXXX      ; push expected R0 contents
                                 0978       ;       CALLS   #1,W^REG_CHECK  ; execute this routine
                                 0978       ;
                                 0978       ; INPUT PARAMETERS:
                                 0978       ;       expected R0 contents on the stack
                                 0978       ;
                                 0978       ; OUTPUT PARAMETERS:
                                 0978       ;       possible error messages printed using $PUTMSG
                                 0978       ;
                                 0978       ;--
                                 0978
                                 0978       REG_CHECK:
                           0FFC  0978             .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
               50   04 AC   D1  097A             CMPL    4(AP),R0                        ; is this the right fail code?
                        0E   13  097E             BEQL    10$                             ; br if yes
                        50   DD  0980             PUSHL   R0                              ; push received data
                     04 AC   DD  0982             PUSHL   4(AP)                           ; push expected data
                   01A4'CF   DF  0985             PUSHAL  W^EXP                           ; push the string variable
             0E7E'CF    03   FB  0989             CALLS   #3,W^PRINT_FAIL                 ; print the error message
                                 098E       10$:
      0008'CF   14 AD   28   29  098E             CMPC3   #4*10,^X14(FP),W^REG_SAVE_AREA  ; check all but R0
                        22   13  0995             BEQL    20$                             ; br if O.K.
    56   53   00000008'8F   C3  0997             SUBL3   #REG_SAVE_AREA,R3,R6            ; calculate the register number
                   56   04   C6  099F             DIVL2   #4,R6
              7E   56   02   81  09A2             ADDB3   #^X2,R6,-(SP)           ; set number past R0-R1 and save
                   51   03   CA  09A6             BICL2   #3,R1                   ; backup to register boundrys
                   53   03   CA  09A9             BICL2   #3,R3
```

```
                 61   DD  09AC    911          PUSHL   (R1)                          ; push received data
                 63   DD  09AE    912          PUSHL   (R3)                          ; push expected data
            0095'CF   DF  09B0    913          PUSHAL  W^REG                         ; set string pntr param.
   0E7E'CF    04      FB  09B4    914          CALLS   #4,W^PRINT_FAIL               ; print the error message
                          09B9    915 20$:
                      04  09B9    916          RET
```

```
                         09BA         .SBTTL REG_CHECKNP
                         09BA  ;++
                         09BA  ; FUNCTIONAL DESCRIPTION.
                         09BA  ;      Subroutine to test R0 & R2-R11 for proper content after a service
                         09BA  ;      execution without printing it. A snapshot is taken by the REG_SAVE routine a
                         09BA  ;      beginning of each step and this routine is executed after the
                         09BA  ;      services have been executed. This routine collects the error
                         09BA  ;      information in buffer ERLB instead of printing it.
                         09BA  ;
                         09BA  ; CALLING SEQUENCE:
                         09BA  ;      PUSHL   #SS$_XXXXXX      ; push expected R0 contents
                         09BA  ;      CALLS   #1,W^REG_CHECK   ; execute this routine
                         09BA  ;
                         09BA  ; INPUT PARAMETERS:
                         09BA  ;      expected R0 contents on the stack
                         09BA  ;
                         09BA  ; OUTPUT PARAMETERS:
                         09BA  ;      possible error messages logged in buffer ERLB which are printed
                         09BA  ;      using routine ERLBUF_DUMP.
                         09BA  ;
                         09BA  ;      Error packets are in the following form:
                         09BA  ;                   !-----------------!
                         09BA  ;                   !Service name pntr!
                         09BA  ;                   !-----------------!
                         09BA  ;                   !     Step #      !
                         09BA  ;                   !-----------------!
                         09BA  ;                   !Mode name pointer!
                         09BA  ;                   !-----------------!
                         09BA  ;                   !            !    ! long word count
                         09BA  ;                   !-----------------!
                         09BA  ;                   !\/\/\/\/\/\/\/\/\! 3-4 parameter long words
                         09BA  ;
                         09BA  ;--
                         09BA
                         09BA  FLAG:
                    00   09BA         .BYTE 0                  ; error flags are BIT0 = 0 means no errors in the bu
                         09BB                                  ;                 BIT0 = 1 means errors in the buffe
                         09BB  ELBP:
             000009BF'   09BB         .ADDRESS ERLB            ; error log buffer pointer
                         09BF  ERLB:
              00000DA7   09BF         .BLKB   1000             ; error log buffer
                         0DA7  ;
                         0DA7  REG_CHECKNP:
                  0FFC   0DA7         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
      50   04 AC    D1   0DA9         CMPL    4(AP),R0         ; is this the right fail code
              3D    13   0DAD         BEQL    10$              ; br if yes
 FC06 CF      01    88   0DAF         BISB2   #1,FLAG          ; set the error logged flag bit
 52    FC03 CF      D0   0DB4         MOVL    ELBP,R2          ; get the current error log pointer
 82    010F'CF      D0   0DB9         MOVL    W^SERV_NAME,(R2)+ ; save the service name
 82    0004'CF      D0   0DBE         MOVL    W^CURRENT_TC,(R2)+ ; save the step number
 82    0069'CF      D0   0DC3         MOVL    W^MODE,(R2)+     ; save the mode
         82   03    90   0DC8         MOVB    #3,(R2)+         ; save the long word count
         82   50    D0   0DCB         MOVL    R0,(R2)+         ; save received status
    82   04 AC      D0   0DCE         MOVL    4(AP),(R2)+      ; save expected status
 82    01A4'CF      DE   0DD2         MOVAL   W^EXP,(R2)+      ; save the string variable
              62    94   0DD7         CLRB    (R2)             ; set the terminator
 FBDD CF      52    D0   0DD9         MOVL    R2,ELBP          ; reset the buffer pointer
```

```
       C04C'CF    002A'CF   DE  0DDE            MOVAL   W^TEST_MOD_FAIL,W^TMD_ADDR ; set failure message address
  0044'CF    03    00    02   F0  0DE5            INSV    #ERROR,#0,#3,W^MOD_MSG_CODE ; set severity code
                                  0DEC      10$:
    0008'CF    14 AD    28   29  0DEC            CMPC3   #4*10,^X14(FP),W^REG_SAVE_AREA ; check all but R0 and R1
                         4B   13  0DF3            BEQL    20$               ; br if OK
            FBC0 CF      01   88  0DF5            BISB2   #1,FLAG           ; set error logged flag bit
            52    FBBD CF     D0  0DFA            MOVL    ELBP,R2           ; get current error log buf pointer
            82    010F'CF     D0  0DFF            MOVL    W^SERV_NAME,(R2)+ ; save the service name
            82    0004'CF     D0  0E04            MOVL    W^CURRENT_TC,(R2)+ ; save the step number
            82    0069'CF     D0  0E09            MOVL    W^MODE,(R2)+      ; save the mode
                  82     04   90  0E0E            MOVB    S^#4,(R2)+        ; set longword count
             00000008'8F      C3  0E11            SUBL3   #REG_SAVE_AREA,-
                  56     53       0E17            R3,R6                     ; calc reg number
                  56     04   C6  0E19            DIVL2   S^#4,R6           ; make it a longword count
            82    56     02   C1  0E1C            ADDL3   S^#2,R6,(R2)+     ; correct for R0-R1 and save
                  82     61   D0  0E20            MOVL    (R ),(R2)+        ; save received results
                  82     63   D0  0E23            MOVL    (R3),(R2)+        ; save expected results
            82    0095'CF     DE  0E26            MOVAL   W^REG,(R2)+       ; save string variable
                         62   94  0E2B            CLRB    (R2)              ; set the terminator
            FB89 CF      52   D0  0E2D            MOVL    R2,ELBP           ; reset the buffer pointer
       004C'CF    002A'CF     DE  0E32            MOVAL   W^TEST_MOD_FAIL,W^TMD_ADDR      ; set failure message address
  0044'CF    03    00    02   F0  0E39            INSV    #ERROR,#0,#3,W^MOD_MSG_CODE     ; set severity code
                                  0E40      20$:
                              04  0E40            RET                       ; bail out
```

```
                                0E41             .SBTTL  ERLBUF_DUMP
                                0E41   ;++
                                0E41   ; FUNCTIONAL DESCRIPTION:
                                0E41   ;         Routine to check for errors in the error log buffer and
                                0E41   ;         report any that are there.
                                0E41   ;
                                0E41   ; CALLING SEQUENCE:
                                0E41   ;         CALLS #0,W^ERLBUF_DUMP
                                0E41   ;
                                0E41   ; INPUT PARAMETERS:
                                0E41   ;         FLAG bit 0 = 0 for no errors logged
                                0E41   ;         FLAG bit 0 = 1 for errors logged
                                0E41   ;         if errors logged then buffer ERLB must contain legal format errors
                                0E41   ;
                                0E41   ; OUTPUT PARAMETERS:
                                0E41   ;         NONE
                                0E41   ;
                                0E41   ;--
                                0E41
                                0E41   ERLBUF_DUMP:
                          001C  0E41         .WORD   ^M<R2,R3,R4>
         2A FB73 CF        E9   0E43         BLBC    FLAG,30$            ; br if no errors to report
    52      FB73 CF        DE   0E48         MOVAL   ERLB,R2             ; set up buffer pointer
                                0E4D   10$:
                      62   95   0E4D         TSTB    (R2)                ; any more errors?
                      21   13   0E4F         BEQL    30$                 ; br if not
    010F'CF           82   D0   0E51         MOVL    (R2)+,W^SERV_NAME ; reset service name
    0004'CF           82   D0   0E56         MOVL    (R2)+,W^CURRENT_TC ; reset step #
    0069'CF           82   D0   0E5B         MOVL    (R2)+,W^MODE        ; reset the mode
         53           82   9A   0E60         MOVZBL  (R2)+,R3            ; get the longword count
         54           53   D0   0E63         MOVL    R3,R4               ; and save it
                                0E66   20$:
                      82   DD   0E66         PUSHL   (R2)+               ; push a parameter
                   FB 53   F5   0E68         SOBGTR  R3,20$              ; and push them all
    0E7E'CF           54   FB   0E6B         CALLS   R4,W^PRINT_FAIL ; print the failure
                      DB   11   0E70         BRB     10$                 ; do the next one
                                0E72   30$:
FB42 CF     FB49 CF        DE   0E72         MOVAL   ERLB,ELBP           ; reset the buffer pointer
            FB42 CF        94   0E79         CLRB    W^ERLB              ; set fresh terminater
                           04   0E7D         RET                         ; bail out
```

```
                                0E7E         .SBTTL  PRINT_FAIL
                                0E7E    ;++
                                0E7E    ; FUNCTIONAL DESCRIPTION:
                                0E7E    ;       Subroutine to report failures using $PUTMSG
                                0E7E    ;
                                0E7E    ; CALLING SEQUENCE:
                                0E7E    ; Mode  #1      PUSHL EXPECTED  Mode    #2      PUSHL REG_NUMBER
                                0E7E    ;               PUSHL RECEIVED                  PUSHL EXPECTED
                                0E7E    ;               PUSHAL STRING_VAR               PUSHL RECEIVED
                                0E7E    ;               CALLS #3,W^PRINT_FAIL           PUSHAL STRING_VAR
                                0E7E    ;                                               CALLS #4,W^PRINT_FAIL
                                0E7E    ; INPUT PARAMETERS:
                                0E7E    ;       listed above
                                0E7E    ;
                                0E7E    ; OUTPUT PARAMETERS:
                                0E7E    ;       an error message is printed using $PUTMSG
                                0E7E    ;
                                0E7E    ;--
                                0E7E
                                0E7E    PRINT_FAIL:
                          003C  0E7E            .WORD   ^M<R2,R3,R4,R5>
                                0E80            $FAO_S  W^CS1,W^MESSAGEL,W^MSGL,#TEST_MOD_NAME,W^SERV_NAME,W^CURRENT_TC
                                0EA1            $PUTMSG_S W^MSGVEC                      ; print the message
               04     6C    91  0EB2            CMPB    (AP),#4                         ; is this a register message?
                      21    13  0EB5            BEQL    10$                             ; br if yes
                                0EB7            $FAO_S  W^CS2,W^MESSAGEL,W^MSGL,4(AP),8(AP),4(AP),12(AP)
                      25    11  0ED6            BRB     20$                             ; goto output message
                                0ED8    10$:
                                0ED8            $FAO_S  W^CS3,W^MESSAGEL,W^MSGL,4(AP),16(AP),8(AP),4(AP),16(AP),12(AP)
                                0EFD    20$:
                                0EFD
                                0EFD            $PUTMSG_S W^MSGVEC                      ; print the message
         0F22'CF      00    FB  0F0E            CALLS   #0,W^MODE_ID                    ; identify the mode
     004C'CF   002A'CF      DE  0F13            MOVAL   W^TEST_MOD_FAIL,W^TMD_ADDR      ; set failure message address
0044'CF   03      00    02  F0  0F1A            INSV    #ERROR,#0,#3,W^MOD_MSG_CODE     ; set severity code
                            04  0F21            RET
```

```
                         0F22          .SBTTL  MODE_ID
                         0F22  ;++
                         0F22  ; FUNCTIONAL DESCRIPTION:
                         0F22  ;       Subroutine to identify the mode that an exit handler is in.
                         0F22  ;
                         0F22  ; CALLING SEQUENCE:
                         0F22  ;       CALLS   #0,W^MODE_ID
                         0F22  ;
                         0F22  ; INPUT PARAMETERS:
                         0F22  ;       MODE contains an address pointing to an ascii string desc.
                         0F22  ;       of the current CPU mode.
                         0F22  ;
                         0F22  ; OUTPUT PARAMETERS:
                         0F22  ;       NONE
                         0F22  ;
                         0F22  ;--
                         0F22
                         0F22  MODE_ID:
                   003C  0F22          .WORD   ^M<R2,R3,R4,R5>
                         0F24          $FAO_S  W^CS5,W^MESSAGEL,W^MSGL,MODE ; format the error message
                         0F3D          $PUTMSG_S W^MSGVEC                   ; print the mode message
                     04  0F4E          RET
                         0F4F          .SBTTL  EXCEP_FAIL
                         0F4F  ;++
                         0F4F  ; FUCTIONAL DESCRIPTION:
                         0F4F  ;       Subroutine to identify an exception failure.
                         0F4F  ;
                         0F4F  ; CALLING SEQUENCE:
                         0F4F  ;       CALLS #0,W^EXCEP_FAIL
                         0F4F  ;
                         0F4F  ; INPUT PARAMETERS:
                         0F4F  ;       MODE            contains an address pointing to an ascii string desc.
                         0F4F  ;                       of the current CPU mode.
                         0F4F  ;       CURRENT_TC      contains the current test case number.
                         0F4F  ;       FLAG            contains expected or unexpected flag.
                         0F4F  ;
                         0F4F  ; OUTPUT PARAMETERS:
                         0F4F  ;       NONE
                         0F4F  ;
                         0F4F  ;--
                         0F4F
                         0F4F  EXCEP_FAIL:
                   003C  0F4F          .WORD   ^M<R2,R3,R4,R5>
23 FA64 CF    01     E1  0F51          BBC     #1,W^FLAG,10$                      ; br if unexpected exception
                         0F57          $FAO_S  W^CS6,W^MESSAGEL,W^MSGL,W^MODE,-
                         0F57                  #TEST_MOD_NAME,W^CURRENT_TC        ; print missing exception
              21     11  0F78          BRB     20$                                ; and carry on
                         0F7A  10$:
                         0F7A          $FAO_S  W^CS4,W^MESSAGEL,W^MSGL,W^MODE,-
                         0F7A                  #TEST_MOD_NAME,W^CURRENT_TC        ; print unexpected exception
                         0F9B  20$:
                         0F9B          $PUTMSG_S       W^MSGVEC                   ; print the message
                     04  0FAC          RET
```

```
                                0FAD           .SBTTL  EXCEP_CHECK
                                0FAD   ;++
                                0FAD   ; FUNCTIONAL DESCRIPTION:
                                0FAD   ;       Routine to check for proper exception name.
                                0FAD   ;
                                0FAD   ; CALLING SEQUENCE:
                                0FAD   ;       CALLS #0,W^EXCEP_CHECK
                                0FAD   ;
                                0FAD   ; INPUT PARAMETERS:
                                0FAD   ;       NONE
                                0FAD   ;
                                0FAD   ; OUTPUT PARAMETERS:
                                0FAD   ;       Possible error messages.
                                0FAD   ;
                                0FAD   ;--
                                0FAD
                                0FAD   WORK1:
                      00000000  0FAD           .LONG   0                       ; temp storage
                                0FB1   ;
                                0FB1   EXCEP_CHECK:
           F8 AF   52     D0    0FB1           MOVL    R2,B^WORK1              ; save r2
           52   04 AC     D0    0FB5           MOVL    CHF$L_SIGARGLST(AP),R2  ; get signal array pointer
                04 A2     D1    0FB9           CMPL    CHF$L_SIG_NAME(R2),-
          00000000'8F           0FBC                   #SS$_CMODUSER           ; is it the right exception?
                   12     13    0FC1           BEQL    10$                     ; br if yes
                04 A2     DD    0FC3           PUSHL   B^CHF$L_SIG_NAME(R2)    ; push received
          00000000'8F     DD    0FC6           PUSHL   #SS$_CMODUSER           ; push expected
              01A4'CF     DF    0FCC           PUSHAL  W^EXP                   ; push string variable
          FEA9 CF  03     FB    0FD0           CALLS   #3,W^PRINT_FAIL         ; print the error
                                0FD5   10$:
           52   D5 AF     D0    0FD5           MOVL    B^WORK1,R2              ; restore R2
                          05    0FD9           RSB                             ; return
                                0FDA           .SBTTL STACK_CHECK
                                0FDA   ;++
                                0FDA   ; FUNCTIONAL DESCRIPTION:
                                0FDA   ;       Routine to check the stack level.
                                0FDA   ;
                                0FDA   ; CALLING SEQUENCE:
                                0FDA   ;       BSBW W^STACK_CHECK
                                0FDA   ;
                                0FDA   ; INPUT PARAMETERS:
                                0FDA   ;       WORK2 = stack pointer value to check against
                                0FDA   ;
                                0FDA   ; OUTPUT PARAMETERS:
                                0FDA   ;       NONE
                                0FDA   ;
                                0FDA   ;--
                                0FDA
                                0FDA   WORK2:
                      00000000  0FDA           .LONG   0               ; stack save location
                                0FDE   STACK_CHECK:
           F8 AF   5E     D1    0FDE           CMPL    SP,B^WORK2      ; check the level
                   0E     13    0FE2           BEQL    10$             ; br if OK
                   5E     DD    0FE4           PUSHL   SP              ; push received
                F1 AF     DD    0FE6           PUSHL   B^WORK2         ; push expected
              01B2'CF     DF    0FE9           PUSHAL  W^STACK         ; push string variable
          FE8C CF  03     FB    0FED           CALLS   #3,W^PRINT_FAIL ; print the failure
```

```
                          0FF2  10$:
                     05   0FF2          RSB                          ; return
                          0FF3          .SBTTL  EXCEP_CHECKNP
                          0FF3  ;++
                          0FF3  ; FUNCTIONAL DESCRIPTION:
                          0FF3  ;       Routine to check for proper exception name without printing.
                          0FF3  ;
                          0FF3  ; CALLING SEQUENCE:
                          0FF3  ;       CALLS #0,W^EXCEP_CHECKNP
                          0FF3  ;
                          0FF3  ; INPUT PARAMETERS:
                          0FF3  ;       NONE
                          0FF3  ;
                          0FF3  ; OUTPUT PARAMETERS:
                          0FF3  ;       Possible output to ERLB.
                          0FF3  ;
                          0FF3  ;--
                          0FF3
                          0FF3  ;
                          0FF3  EXCEP_CHECKNP:
      FFB5 CF      52  D0 0FF3          MOVL    R2,W^WORK1              ; save R2
        DE AF      53  D0 0FF8          MOVL    R3,B^WORK2              ; save R3
        53     04 AC   D0 0FFC          MOVL    CHF$L_SIGARGLST(AP),R3  ; get signal array pointer
               04 A3   D1 1000          CMPL    CHF$L_SIG_NAME(R3),-
       00000000'8F        1003                  #SS$_CMODUSER           ; is it the right exception?
               41  13     1008          BEQL    10$                     ; br if yes
      F9AB CF      01  88 100A          BISB2   #1,W^FLAG               ; set the error logged flag bit
      52    F9A8 CF    D0 100F          MOVL    ELBP,R2                 ; get current error log pointer
      82    010F'CF    D0 1014          MOVL    W^SERV_NAME,(R2)+       ; save the service name
      82    0004'CF    D0 1019          MOVL    W^CURRENT_TC,(R2)+      ; save the step number
      82    0069'CF    D0 101E          MOVL    W^MODE,(R2)+            ; save the mode
           82      03  90 1023          MOVB    S^#3,(R2)+              ; save the long word count
        82     04 A3   D0 1026          MOVL    CHF$L_SIG_NAME(R3),(R2)+ ; save received name
    82    00000000'8F  D0 102A          MOVL    #SS$_CMODUSER (R2)+     ; save expected name
      82    01A4'CF    DE 1031          MOVAL   W^EXP,(R2)+             ; save string variable
                   62  94 1036          CLRB    (R2)                    ; set the terminator
      F97E CF      52  D0 1038          MOVL    R2,W^ELBP               ; reset the buffer pointer
  004C'CF     002A'CF  DE 103D          MOVAL   W^TEST_MOD_FAIL,W^TMD_ADDR ; set failure message adr
0044'CF   03   00  02  F0 1044          INSV    #ERROR,#0,#3,W^MOD_MSG_CODE ; set severity code
                          104B  10$:
      52    FF5E CF    D0 104B          MOVL    W^WORK1,R2              , restore R2
        53     87 AF   D0 1050          MOVL    B^WORK2,R3              ; restore R3
                       05 1054          RSB                             ; return
```

```
                    1055 MOD_MSG_PRINT:
                    1055 ;
                    1055 ;  ***********************************************************************
                    1055 ;  *                                                                     *
                    1055 ;  *  PRINTS THE TEST MODULE BEGUN/SUCCESSFUL/FAILED MESSAGES            *
                    1055 ;  *       (USING THE PUTMSG MACRO).                                     *
                    1055 ;  *                                                                     *
                    1055 ;  ***********************************************************************
                    1055 ;
                    1055         PUTMSG  <MOD_MSG_CODE,#2,TMN_ADDR,TMD_ADDR> ; PRINT MSG
                05  1070         RSB                                 ; ... AND RETURN TO CALLER
                    1071 ;
                    1071 CHMRTN:
                    1071 ; ************************************************************************
                    1071 ; *                                                                      *
                    1071 ; *      CHANGE MODE ROUTINE. THIS ROUTINE GETS CONTROL WHENEVER         *
                    1071 ; *      A CMKRNL, CMEXEC, OR CMSUP SYSTEM SERVICE IS ISSUED             *
                    1071 ; *      BY THE MODE MACRO ('TO' OPTION).  IT MERELY DOES                *
                    1071 ; *      A JUMP INDIRECT ON A FIELD SET UP BY MODE. IT HAS               *
                    1071 ; *      THE EFFECT OF RETURNING TO THE END OF THE MODE                  *
                    1071 ; *      MACRO EXPANSION.                                                *
                    1071 ; *                                                                      *
                    1071 ; ************************************************************************
                    1071 ;
              0000  1071         .WORD   0                   ; ENTRY MASK
00000059'FF     17  1073         JMP     @CHM_CONT           ; RETURN TO MODE MACRO IN NEW MODE
                    1079 ;
                    1079 ; *      RET INSTR WILL BE ISSUED IN EXPANSION OF 'MODE FROM, ....' MACRO
                    1079 ;
                    1079         .END    SATSSS22
```

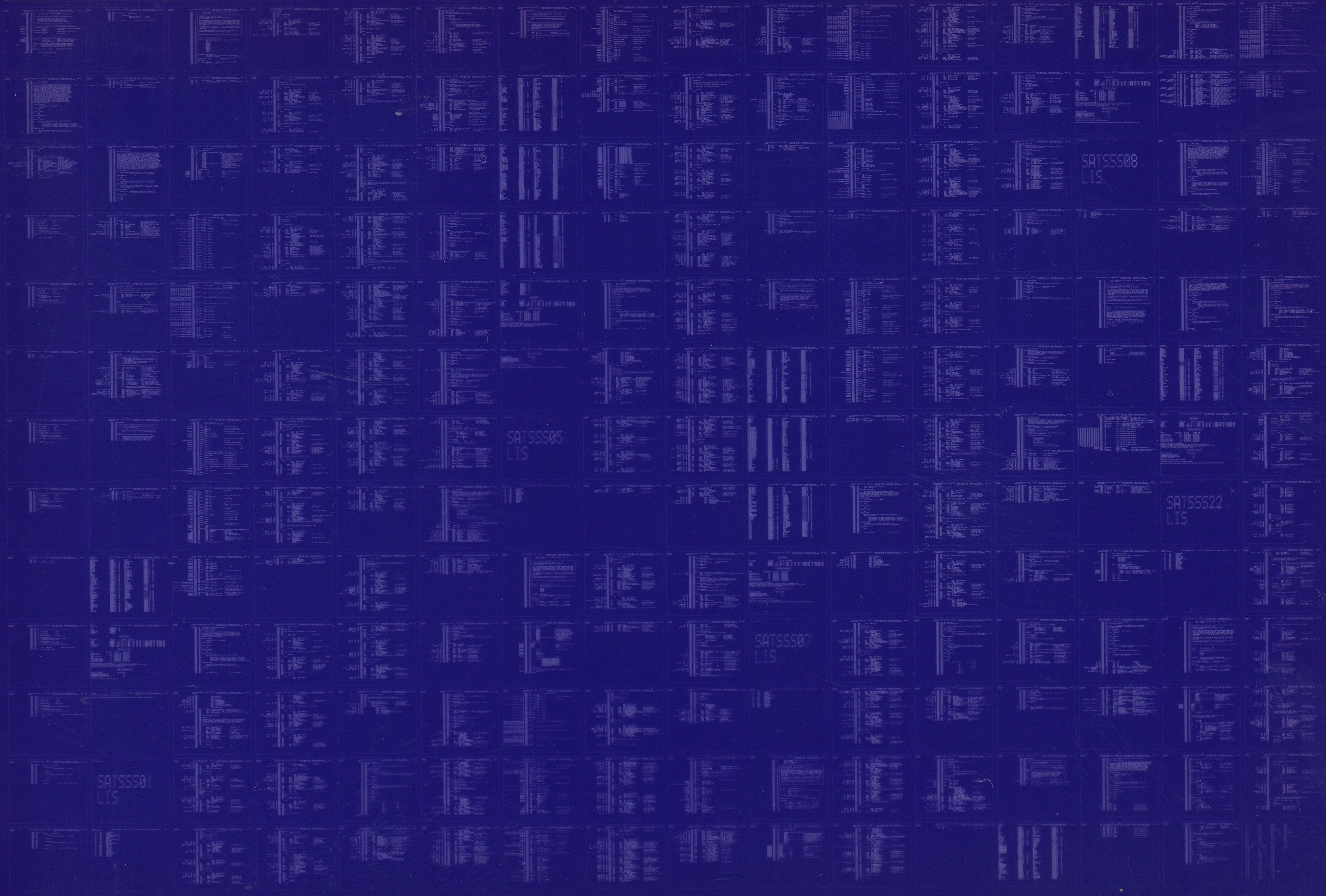
```
                                                                  M  2
SATSSS22                          - SATS SYSTEM SERVICE TESTS  (SUCC S.C.) 16-SEP-1984 00:48:27  VAX/VMS Macro V04-00        Page  41
Symbol table                                                                5-SEP-1984 04:30:12  [UETPSY.SRC]SATSSS22.MAR;1        (3)

$$ARGS                          = 00000002                   MODE                            00000069 R     03
$$T1                            = 00000004                   MODE_ID                         00000F22 R     04
$$T2                            = 00000006                   MOD_MSG_CODE                    00000044 R     03
A10                               00000416 R     04          MOD_MSG_PRINT                   00001055 R     04
A20                               00000480 R     04          MSGL                            000000AF R     03
A30                               00000481 R     04          MSGVEC                          000001D5 R     02
A40                               000004C7 R     04          MSGVEC1                         00000113 R     03
A50                               00000534 R     04          NOT_ENABLED                     0000040E R     04
A60                               00000535 R     04          NOT_ENABLED1                    000004BF R     04
ARGLST                            000001CD R     02          PR$_USP                       = 00000003
B10                               00000297 R     04          PRE_EXEC_PRI                    000001F0 R     04
B30                               0000093C R     04          PRE_EXEC_SEC                    000001F4 R     04
B70                               0000096D R     04          PRE_KERNEL_PRI                  0000031C R     04
BUF                               000000B7 R     03          PRE_KERNEL_SEC                  00000320 R     04
C10                               000003C3 R     04          PRE_SUPER_PRI                   00000893 R     04
CHF$L_MCHARGLST                 = 00000008                   PRE_SUPER_SEC                   00000897 R     04
CHF$L_MCH_SAVR0                 = 0000000C                   PRE_USER_PRI                    000000B5 R     04
CHF$L_SIGARGLST                 = 00000004                   PRE_USER_SEC                    000000B9 R     04
CHF$L_SIG_NAME                  = 00000004                   PRINT_FAIL                      00000E7E R     04
CHMRTN                            00001071 R     04          PRIVMASK                        00000051 R     03
CHM_CONT                          00000059 R     03          PRVHND1                         000000AB R     03
CS1                               00000046 R     02          PRVPRT                          00000050 R     03
CS2                               00000078 R     02          PSL$C_EXEC                    = 00000001
CS3                               000000A5 R     02          PSL$C_KERNEL                  = 00000000
CS4                               000000D8 R     02          PSL$C_SUPER                   = 00000002
CS5                               00000116 R     02          PSL$C_USER                    = 00000003
CS6                               0000012B R     02          PSL$S_CURMOD                  = 00000002
CURRENT_TC                        00000004 R     03          PSL$V_CURMOD                  = 00000018
DCLCMH                            00000831 R     04          PSL$V_PRVMOD                  = 00000016
DEPTH                             00000128 R     03          REG                             00000095 R     03
ELBP                              000009BB R     04          REGNUM                          000000A7 R     03
EM                                00000185 R     02          REG_CHECK                       00000978 R     04
ENABLED                           0000045A R     04          REG_CHECKNP                     00000DA7 R     04
ENABLED1                          0000050E R     04          REG_SAVE                        0000096E R     04
ERLB                              000009BF R     04          REG_SAVE_AREA                   00000008 R     03
ERLBUF_DUMP                       00000E41 R     04          RETADR                          0000005D R     03
ERROR                           = 00000002                   RETPC                           000001BC R     02
EXCEP_CHECK                       00000FB1 R     04          RETURN_PC                       000007BF R     04
EXCEP_CHECKNP                     00000FF3 R     04          SATSSS22                        00000000 RG    04
EXCEP_FAIL                        00000F4F R     04          SERV_NAME                       0000010F R     03
EXE$C_CMSTKSZ                     ********    X  04          SET                             0000006D R     03
EXEC_END                          00000246 R     04          SET1                            00000081 R     03
EXEC_HANTAB                       000001E8 R     04          SETEXV                          00000031 R     02
EXEC_PRIM                         000001F8 R     04          SETEXV$_ACMODE                = 0000000C
EXEC_SEC                          00000219 R     04          SETEXV$_ADDRES                = 00000008
EXP                               000001A4 R     02          SETEXV$_NARGS                 = 00000004
FLAG                              000009BA R     04          SETEXV$_PRVHND                = 00000010
FLAG1                             00000127 R     03          SETEXV$_VECTOR                = 00000004
HANDLER_PC                        000007C3 R     04          SETSFM                          00000038 R     02
INFO                            = 00000003                   SETSFM$_ENBFLG                = 00000004
KERNEL_END                        00000372 R     04          SETSFM$_NARGS                 = 00000001
KERNEL_HANTAB                     00000314 R     04          SETUP_SUPER                     000007C7 R     04
KERNEL_PRIM                       00000324 R     04          SEVERE                        = 00000004
KERNEL_SEC                        00000345 R     04          SF$L_SAVE_FP                  = 0000000C
KM                                00000196 R     02          SF$L_SAVE_PC                  = 00000010
LIB$SIGNAL                        ********    X  04          SHR$K_SHRDEF                  = 00000001
MESSAGEL                          00000107 R     03          SHR$_TEXT                     = 00001130
```

```
                                                    N 2
SATSSS22                        - SATS SYSTEM SERVICE TESTS  (SUCC S.C.) 16-SEP-1984 00:48:27  VAX/VMS Macro V04-00        Page  42       SA
Symbol table                                                              5-SEP-1984 04:30:12  [UETPSY.SRC]SATSSS22.MAR;1         (3)       VO

SM                                00000178 R     02         SYS$PUTMSG                    ********  GX    04
SS$_CMODUSER                      ********    X  04         SYS$SETEXV                    ********  GX    04
SS$_CONTINUE                      ********    X  04         SYS$SETPRN                    ********  GX    04
SS$_ILLEFC                        ********    X  04         SYS$SETSFM                    ********  GX    04
SS$_NORMAL                        ********    X  04         SYS$UNWIND                    ********  GX    04
SS$_RESIGNAL                      ********    X  04         SYS$WAKE                      ********  GX    04
SS$_UNWIND                        ********    X  04         TEST_MOD_BEGIN                00000019 R      02
SS$_WASCLR                        ********    X  04         TEST_MOD_FAIL                 0000002A R      02
SS$_WASSET                        ********    X  04         TEST_MOD_NAME                 00000000 R      02
STACK                             000001B2 R     02         TEST_MOD_NAME_D               00000009 R      02
STACK_CHECK                       00000FDE R     04         TEST_MOD_SUCC                 0000001F R      02
STATUS                            00000065 R     03         TMD_ADDR                      0000004C R      03
STEP                            = 0000000B                  TMN_ADDR                      00000048 R      03
STP0                              0000003D R     04         TPID                          00000000 R      03
STP10                             000008E9 R     04         UETP$_SATSMS                = 007480D9
STP11                             00000911 R     04         UETP$_TEXT                  = 00741133
STP12                             00000170 R     04         UM                            0000016C R      02
STP14                             000001FA R     04         UNW                           00000089 R      03
STP15                             0000021B R     04         UNWIND                        0000003F R      02
STP16                             00000246 R     04         UNWIND$_DEPADR              = 00000004
STP17                             0000026E R     04         UNWIND$_NARGS               = 00000002
STP18                             0000029C R     04         UNWIND$_NEWPC               = 00000008
STP2                              000000BF R     04         USER_END                      0000010B R      04
STP20                             00000326 R     04         USER_HANTAB                   000000AD R      04
STP21                             00000347 R     04         USER_PRIM                     000000BD R      04
STP22                             00000372 R     04         USER_SEC                      000000DE R      04
STP23                             0000039A R     04         WARNING                     = 00000000
STP24                             000003C8 R     04         WORK                          0000082D R      04
STP25                             00000416 R     04         WORK1                         00000FAD R      04
STP26                             00000486 R     04         WORK2                         00000FDA R      04
STP27                             000004C7 R     04
STP28                             0000053A R     04
STP29                             000005AA R     04
STP3                              000000E0 R     04
STP30                             00000606 R     04
STP31                             00000669 R     04
STP32                             000006C9 R     04
STP33                             00000732 R     04
STP4                              0000010B R     04
STP5                              00000133 R     04
STP6                              0000015B R     04
STP8                              0000089D R     04
STP9                              000008BE R     04
STS$V_INHIB_MSG                 = 0000001C
SUCCESS                         = 00000001
SUPER_END                         000008E9 R     04
SUPER_HANTAB                      0000088B R     04
SUPER_MODE                        00000838 R     04
SUPER_PRIM                        0000089B R     04
SUPER_SEC                         000008BC R     04
SYS$CLREF                         ********   GX  04
SYS$CMEXEC                        ********   GX  04
SYS$CMKRNL                        ********   GX  04
SYS$DCLCMH                        ********   GX  04
SYS$EXIT                          ********   GX  04
SYS$FAO                           ********    X  04
SYS$HIBER                         ********   GX  04
```

```
                                                      +----------------+
                                                      ! 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                          000001E5  (   485.)  02 (  2.)  NOPIC    USR    CON    REL    LCL NOSHR NOEXE   RD   NOWRT NOVEC LONG
RWDATA                          0000012C  (   300.)  03 (  3.)  NOPIC    USR    CON    REL    LCL NOSHR NOEXE   RD     WRT NOVEC LONG
SATSSS22                        00001079  ( 4217.)  04 (  4.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC LONG

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

Phase                      Page faults    CPU Time       Elapsed Time
-----                      -----------    --------       ------------
Initialization                      29    00:00:00.08    00:00:00.47
Command processing                 107    00:00:00.75    00:00:02.73
Pass 1                             419    00:00:12.66    00:00:31.35
Symbol table sort                    0    00:00:00.91    00:00:01.62
Pass 2                             306    00:00:04.35    00:00:09.90
Symbol table output                 24    00:00:00.16    00:00:00.35
Psect synopsis output                2    00:00:00.03    00:00:00.03
Cross-reference output               0    00:00:00.00    00:00:00.00
Assembler run totals               889    00:00:18.95    00:00:46.46

The working set limit was 1800 pages.
96225 bytes (188 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 579 non-local and 54 local symbols.
1262 source lines were read in Pass 1, producing 33 object records in Pass 2.
60 pages of virtual memory were used to define 51 macros.

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

Macro library name                                Macros defined
------------------                                --------------
_$255$DUA28:[SYSLIB]STARLET.MLB;2                       35
_$255$DUA28:[SHRLIB]UETP.MLB;1                          12
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                           0
_$255$DUA28:[SYSLIB]STARLET.MLB;2                        0
TOTALS (all libraries)                                  47

768 GETS were required to define 47 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:SATSSS22/OBJ=OBJ$:SATSSS22 MSRC$:SATSSS22/UPDATE=(ENH$:SATSSS22)+EXECML$/LIB+SHRLIB$:UETP/LIB
```

0421 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY


SATSSS08
LIS

SATSSS05
LIS

SATSSS22
LIS

SATSSS07
LIS

SATSSS01
LIS

0422 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SATSSS35
LIS

SATSSS26
LIS

SATSSS38
LIS

SATSSS30
LIS

SATSSS37
LIS

SATSSS39
LIS

SATSSS36
LIS