```
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSSS  YYY         YYY
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSSS  YYY         YYY
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        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
```

FILEID**SATSSS26

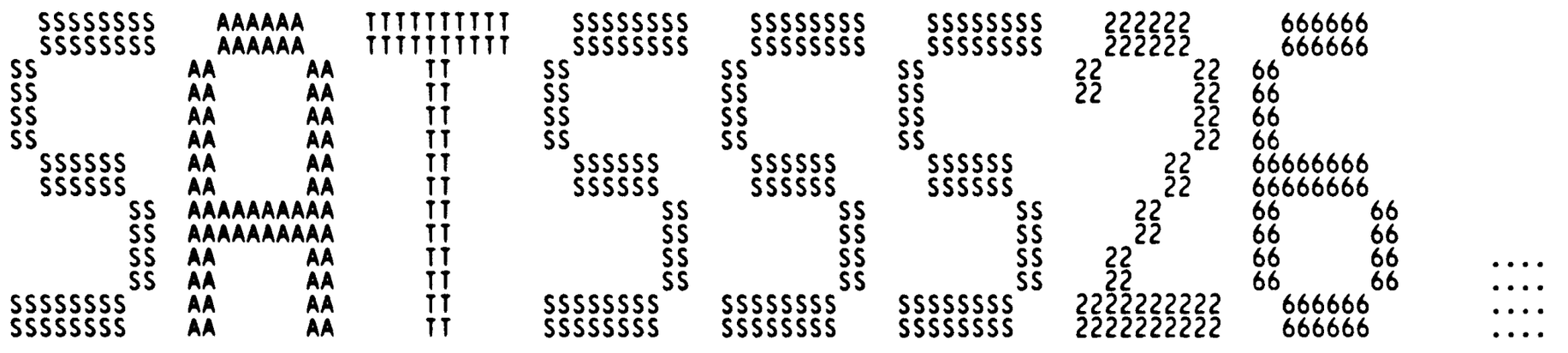

Table of contents

```
0000      1          .TITLE  SATSSS26 - SATS SYSTEM SERVICE TESTS  (SUCC S.C.)
0000      2          .IDENT  'V04-000'
0000      3
0000      4  ;
0000      5  ;*****************************************************************************
0000      6  ;*                                                                           *
0000      7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000      8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000      9  ;*  ALL RIGHTS RESERVED.                                                     *
0000     10  ;*                                                                           *
0000     11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000     12  ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE            *
0000     13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER       *
0000     14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000     15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
0000     16  ;*  TRANSFERRED.                                                             *
0000     17  ;*                                                                           *
0000     18  ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
0000     19  ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
0000     20  ;*  CORPORATION.                                                             *
0000     21  ;*                                                                           *
0000     22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
0000     23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000     24  ;*                                                                           *
0000     25  ;*                                                                           *
0000     26  ;*****************************************************************************
0000     27  ;
0000     28
0000     29  ;++
0000     30  ; FACILITY:      SATS SYSTEM SERVICE TESTS
0000     31  ;
0000     32  ; ABSTRACT:      The SATSSS26 module tests the execution of the following
0000     33  ;                VMS system services:
0000     34  ;
0000     35  ;                $DCLAST
0000     36  ;                $SETAST
0000     37  ;                $CANTIM
0000     38  ;                $SETIMR
0000     39  ;                $SETPRA
0000     40  ;
0000     41  ;
0000     42  ; ENVIRONMENT:  User, Supervisor, Executive, and Kernal mode image.
0000     43  ;                Needs CMKRNL privilege and dynamically acquires other
0000     44  ;                privileges, as needed.
0000     45  ;
0000     46  ; AUTHOR: THOMAS L. CAFARELLA,          CREATION DATE: MMM, 1978
0000     47  ;         PAUL D. FAY (DISPSERV & TESTSERV MACROS)
0000     48  ;
0000     49  ; MODIFIED BY:
0000     50  ;
0000     51  ;       V03-003 LDJ0002         Larry D. Jones,         29-Apr-1981
0000     52  ;               Modified to do all AST checking in non-user mode correctly.
0000     53  ;
0000     54  ;       V03-002 RNP0001         Robert N. Perron,       6-APR-1981
0000     55  ;               Modified to correct hang problem and correct error reporting.
0000     56  ;
0000     57  ;       V03-001 LDJ0001         Larry D. Jones,         17-Sep-1980
```

```
                                F 3
0000     58 ;
0000     59 ;**                    Modified to conform to new build command procedures.
0000     60 ;--
```

```
                     0000     62          .SBTTL  DECLARATIONS
                     0000     63 ;
                     0000     64 ; MACRO LIBRARY CALLS
                     0000     65 ;
                     0000     66          $PRDEF                                  ; processor register definitions
                     0000     67          $PRVDEF                                 ; privilege definitions
                     0000     68          $PSLDEF                                 ; PSL definitions
                     0000     69          $SFDEF                                  ; Stack Frame definitions
                     0000     70          $SHR_MESSAGES UETP,116,<<TEXT,INFO>> ; UETP$_TEXT definition
                     0000     71          $STSDEF                                 ; STS definitions
                     0000     72          $UETPDEF                                ; UETP message definitions
                     0000     73 ;
                     00C0     74 ; Equated symbols
                     0000     75 ;
00000000             0000     76 WARNING          = 0                             ; warning severity value for msgs
00000001             0000     77 SUCCESS          = 1                             ; success      "     "   "   "
00000002             0000     78 ERROR            = 2                             ; error        "     "   "   "
00000003             0000     79 INFO             = 3                             ; information  "     "   "   "
00000004             0000     80 SEVERE           = 4                             ; fatal        "     "   "   "
                     0000     81
```

```
                                                     H 3
                                00000000     83          .PSECT  RODATA,RD,NOWRT,NOEXE,LONG
                                    0000     84 ;
                                    0000     85 DUMMY:                                   ; dummy handler routine address
                                    0000     86 TEST_MOD_NAME:
        36 32 53 53 53 54 41 53 00' 0000     87          .ASCIC  /SATSSS26/              ; needed for SATSMS message
                                    08   0000
                                    0009     88 TEST_MOD_NAME_D:
53 53 53 54 41 53 00000011'010E0000' 0009     89          .ASCID  /SATSSS26/              ; module name
                                 36 32   0017
                                    0019     90 TEST_MOD_BEGIN:
              6E 75 67 65 62 00' 0019     91          .ASCIC  /begun/
                                    05   0019
                                    001F     92 TEST_MOD_SUCC:
   6C 75 66 73 73 65 63 63 75 73 00' 001F     93          .ASCIC  /successful/
                                    0A   001F
                                    002A     94 TEST_MOD_FAIL:
           64 65 6C 69 61 66 00' 002A     95          .ASCIC  /failed/
                                    06   002A
                                    0031     96 DCLCMH:
           48 4D 43 4C 43 44 00' 0031     97          .ASCIC  /DCLCMH/
                                    06   0031
                                    0038     98 DCLAST:
           54 53 41 4C 43 44 00' 0038     99          .ASCIC  /DCLAST/
                                    06   0038
                                    003F    100 SETAST:
           54 53 41 54 45 53 00' 003F    101          .ASCIC  /SETAST/
                                    06   003F
                                    0046    102 SETIMR:
           52 4D 49 54 45 53 00' 0046    103          .ASCIC  /SETIMR/
                                    06   0046
                                    004D    104 CANTIM:
           4D 49 54 4E 41 43 00' 004D    105          .ASCIC  /CANTIM/
                                    06   004D
                                    0054    106 SETPRA:
           41 52 50 54 45 53 00' 0054    107          .ASCIC  /SETPRA/
                                    06   0054
                                    005B    108 CS1:
21 20 74 73 65 54 00000063'010E0000' 005B    109          .ASCID  \Test !AC service name !AC step !UL failed.\
6E 20 65 63 69 76 72 65 73 20 43 41  0069
70 65 74 73 20 43 41 21 20 65 6D 61  0075
2E 64 65 6C 69 61 66 20 4C 55 21 20  0081
                                    008D    110 CS2:
74 63 65 70 78 45 00000095'010E0000' 008D    111          .ASCID  \Expected !AS = !XL received !AS = !XL\
4C 58 21 20 3D 20 53 41 21 20 64 65  009B
41 21 20 64 65 76 69 65 63 65 72 20  00A7
              4C 58 21 20 3D 20 53   00B3
                                    00BA    112 CS3:
74 63 65 70 78 45 000000C2'010E0000' 00BA    113          .ASCID  \Expected !AS!UB = !XL received !AS!UB = !XL\
20 3D 20 42 55 21 53 41 21 20 64 65  00C8
64 65 76 69 65 63 65 72 20 4C 58 21  00D4
58 21 20 3D 20 42 55 21 53 41 21 20  00E0
                                 4C  00EC
                                    00ED    114 CS4:
65 70 78 65 6E 55 000000F5'010E0000' 00ED    115          .ASCID  \Unexpected AST occured.\
63 63 6F 20 54 53 41 20 64 65 74 63  00FB
                  2E 64 65 72 75     0107
                                    010C    116 CS5:
```

```
                                                                I 3
77 20 65 64 6F 4D 00000114'010E0000' 010C    117          .ASCID  \Mode was !AS.\
                  2E 53 41 21 20 73 61  011A
                                        0121    118 CS6:
72 69 75 71 65 52 00000129'010E0000' 0121    119          .ASCID  \Required AST did not occur.\
6E 20 64 69 64 20 54 53 41 20 64 65  012F
         2E 72 75 63 63 6F 20 74 6F  013B
                                        0144    120 CS7:
65 70 78 65 6E 55 0000014C'010E0000' 0144    121          .ASCID  \Unexpected timer AST occured.\
41 20 72 65 6D 69 74 20 64 65 74 63  0152
   2E 64 65 72 75 63 63 6F 20 54 53  015E
                                        0169    122 CS8:
65 70 78 65 6E 55 00000171'010E0000' 0169    123          .ASCID  \Unexpected AST!\
         21 54 53 41 20 64 65 74 63  0177
                                        0180    124 ASTPRM:
61 70 20 54 53 41 00000188'010E0000' 0180    125          .ASCID  \AST parameter\
                  72 65 74 65 6D 61 72  018E
                                        0195    126 EXP:
73 75 74 61 74 73 0000019D'010E0000' 0195    127          .ASCID  \status\
                                        01A3    128 ADR:
73 65 72 64 64 61 000001AB'010E0000' 01A3    129          .ASCID  \address\
                                     73 01B1
                                        01B2    130 MS:
      65 64 6F 6D 000001BA'010E0000' 01B2    131          .ASCID  \mode\
                                        01BE    132 UM:
      72 65 73 75 000001C6'010E0000' 01BE    133          .ASCID  \user\
                                        01CA    134 SM:
   72 65 70 75 73 000001D2'010E0000' 01CA    135          .ASCID  \super\
                                        01D7    136 EM:
74 75 63 65 78 65 000001DF'010E0000' 01D7    137          .ASCID  \executive\
                           65 76 69  01E5
                                        01E8    138 KM:
6C 65 6E 72 65 6B 000001F0'010E0000' 01E8    139          .ASCID  \kernel\
                                        01F6    140 ARGLST:
                           00000001  01F6    141          .LONG   1                       ; super mode setup arg list
                           000009B2' 01FA    142          .ADDRESS SUPER_MODE
                                        01FE    143 MSGVEC:                             ; PUTMSG message vector
                           00000003  01FE    144          .LONG   3
                           00741133  0202    145          .LONG   UETP$_TEXT
                           00000001  0206    146          .LONG   1
                           0000011F' 020A    147          .ADDRESS MESSAGEL
                                        020E    148 ONE_SEC:
                           00989680  020E    149          .LONG   10*1000*1000            ; one second
                                        0212    150 DELTA_5_SEC:
                     FFFFFFFF FD050F80  0212    151          .LONG   -10*1000*1000*5,-1      ; 5 second delta time
                                        021A    152 DELTA_4_SEC:
                     FFFFFFFF FD9DA600  021A    153          .LONG   -10*1000*1000*4,-1      ; 4 second delta time
                                        0222    154 DELTA_1_SEC:
                     FFFFFFFF FF676980  0222    155          .LONG   -10*1000*1000,-1        ; 1 second delta time
```

```
                     022A   157 ;
                     022A   158         .SBTTL  R/W PSECT
                00000000    159         .PSECT  RWDATA,RD,WRT,NOEXE,LONG
                     0000   160 ;
                     0000   161 TPID:
            00000000 0000   162         .LONG   0                       ; PID for this process
                     0004   163 CURRENT_TC:
            00000000 0004   164         .LONG   0                       ; ptr to current test case
                     0008   165 PENDING_TC:
            00000000 0008   166         .LONG   0                       ; test case prior to AST delivery
                     000C   167         .ALIGN LONG
                     000C   168 REG_SAVE_AREA:
            00000048 000C   169         .BLKL   15                      ; register save area
                     0048   170 MOD_MSG_CODE:
            007480D9 0048   171         .LONG   UETP$_SATSMS            ; test module message code for putmsg
                     004C   172 TMN_ADDR:
            00000000' 004C  173         .ADDRESS TEST_MOD_NAME
                     0050   174 TMD_ADDR:
            00000019' 0050  175         .ADDRESS TEST_MOD_BEGIN
                     0054   176 PRVPRT:
                  00 0054   177         .BYTE   0                       ; protection return byte for SETPRT
                     0055   178 PRIVMASK:
      00000000 00000000 0055   179         .QUAD   0                       ; priv. mask
                     005D   180 CHM_CONT:
            00000000 005D   181         .LONG   0                       ; change mode continue address
                     0061   182 RETADR:
            00000069 0061   183         .BLKL   2                       ; returned address's from SETPRT
                     0069   184 DCL:
                     0069   185         $DCLAST AST2,-8,PSL$C_USER      ; DCLAST parameter list
                     0079   186 SET:
                     0079   187         $SETAST 0                       ; SETAST parameter list
                     0081   188 SET1:
                     0081   189         $SETIMR 0,TIME,0,5              ; SETIMR parameter list
                     0095   190 SET2:
                     0095   191         $SETPRA DUMMY+1,PSL$C_KERNEL    ; SETPRA parameter list
                     00A1   192 CAN:
                     00A1   193         $CANTIM 1,0                     ; CANTIM parameter list
                     00AD   194 MODE:
                  00000000  00AD   195         .LONG   0                       ; current mode string pointer
                     00B1   196 REG:
74 73 69 67 65 72 00000089'010E0000' 00B1   197         .ASCID  \register R\
                  52 20 72 65  00BF
                     00C3   198 REGNUM:
                  00000000  00C3   199         .LONG   0                       ; register number
                     00C7   200 MSGL:
                  00000050  00C7   201         .LONG   80                      ; buffer desc.
                  000000CF' 00CB   202         .ADDRESS BUF
                     00CF   203 BUF:
                  0000011F  00CF   204         .BLKB   80
                     011F   205 MESSAGEL:
                  00000000  011F   206         .LONG   0                       ; message length
                  000000CF' 0123   207         .ADDRESS BUF
                     0127   208 SERV_NAME:
                  00000000  0127   209         .LONG   0                       ; service name pointer
                     012B   210 ASTPAR1:
                  FFFFFFFF  012B   211         .LONG   -1                      ; AST 1's parameter
                     012F   212 ASTPAR2:
```

```
                                                K 3
FFFFFFF8  012F   213          .LONG   -8                      ; AST 2's parameter
          0133   214 ASTOK:
      00  0133   215          .BYTE   0                       ; AST's legal flag
          0134   216                                          ; BIT0 = 0 AST's are illegal
          0134   217                                          ; BIT0 = 1 AST's are legal
          0134   218 PRVHND1:
00000000  0134   219          .LONG   0                       ; previous handler address 1
          0138   220 HANDLER_ADR:
00000000  0138   221          .LONG   0                       ; handler address storage
          013C   222 X_SEC:
00000000  013C   223          .LONG   0                       ; scratch good up to 7 sec
          0140   224 TIME:
00000000 00000000 0140 225    .QUAD   0                       ; time parameter for SETIMR & CANTIM
          0148   226 ID:
00000000  0148   227          .LONG   0                       ; timer AST flag
          014C   228 WORK:
00000000  014C   229          .LONG   0                       ; scratch long word
          0150   230 MSGVEC1:                               ; PUTMSG message vector
00000003  0150   231          .LONG   3
00741133  0154   232          .LONG   UETP$_TEXT
00000001  0158   233          .LONG   1
00000000  015C   234          .LONG   0
```

```
00000000   236          .PSECT  SATSSS26,RD,WRT,EXE,LONG
     0000  237          .SBTTL  SATSSS26
     0000  238 ;++
     0000  239 ; FUNCTIONAL DESCRIPTION:
     0000  240 ;
     0000  241 ;       After performing some initial housekeeping, such as
     0000  242 ; printing the module begin message and acquiring needed privileges,
     0000  243 ; the system services are tested in each of their normal conditions.
     0000  244 ; Detected failures are identified and  an error message is printed
     0000  245 ; on the terminal.  Upon completion of the test a success or fail
     0000  246 ; message is printed on the terminal.
     0000  247 ;
     0000  248 ; CALLING SEQUENCE:
     0000  249 ;
     0000  250 ;       $ RUN SATSSS26  ...  (DCL COMMAND)
     0000  251 ;
     0000  252 ; INPUT PARAMETERS:
     0000  253 ;
     0000  254 ;       none
     0000  255 ;
     0000  256 ; IMPLICIT INPUTS:
     0000  257 ;
     0000  258 ;       none
     0000  259 ;
     0000  260 ; OUTPUT PARAMETERS:
     0000  261 ;
     0000  262 ;       none
     0000  263 ;
     0000  264 ; IMPLICIT OUTPUTS:
     0000  265 ;
     0000  266 ;       Messages to SYS$OUTPUT are the only output from SATSSS26.
     0000  267 ;       They are of the form:
     0000  268 ;
     0000  269 ;               %UETP-S-SATSMS, TEST MODULE SATSSS26 BEGUN ... (BEGIN MSG)
     0000  270 ;               %UETP-S-SATSMS, TEST MODULE SATSSS26 SUCCESSFUL ... (END MSG)
     0000  271 ;               %UETP-E-SATSMS, TEST MODULE SATSSS26 FAILED ... (END MSG)
     0000  272 ;               %UETP-I-TEXT, ... (VARIABLE INFORMATION ABOUT A TEST MODULE FAILURE)
     0000  273 ;
     0000  274 ; COMPLETION CODES:
     0000  275 ;
     0000  276 ;       The SATSSS26 routine terminates with a $EXIT to the
     0000  277 ;       operating system with a status code defined by UETP$_SATSMS.
     0000  278 ;
     0000  279 ; SIDE EFFECTS:
     0000  280 ;
     0000  281 ;       none
     0000  282 ;
     0000  283 ;--
     0000  284
     0000  285
     0000  286
     0000  287          TEST_START SATSSS26             ; let the test begin
```

```
                                                M 3
                                0000
                          0000  0000                    .ENTRY SATSSS26,0
                  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
                     1389   30  0025                    BSBW    W^MOD_MSG_PRINT
       0050'CF    001F'CF   DE  0028                    MOVAL   W^TEST_MOD_SUCC,W^TMD_ADDR
0048'CF    03   00   01   F0  002F                    INSV    #SUCCESS,#0,#3,W^MOD_MSG_CODE
                        00  DD  0036                    PUSHL   #0
             0B1A'CF    01  FB  0038                    CALLS #1,W^REG_SAVE
                                003D            STP0:
                                003D  288               $CMKRNL_S W^SETUP_SUPER,W^ARGLST ; declare CHMS handler
                     5E   10'   C0  004C  289           ADDL2   S^#EXE$C_CMSTKSZ+16,SP  ; adjust the user stack pointer
                     5D   5E    D0  004F  290           MOVL    SP,FP                   ; fix the frame pointer
                12A9'CF   00    FB  0052  291           CALLS   #0,W^ERLBUF_DUMP        ; dump any errors that occured at kernal mod
                                0057  292               .SBTTL  AST TESTS
                                0057  293 ;+
                                0057  294 ;
                                0057  295 ; test for user AST's
                                0057  296 ;
                                0057  297 ;          DECLARED ORDER
                                0057  298 ; AST ROUTINE    MODE     AST PARAM
                                0057  299 ; ------------------------------
                                0057  300 ;    AST1        USER         -1
                                0057  301 ;    AST2        USER         -8
                                0057  302 ;-
0AD'CF    01BE'CF   DE  0057  303           MOVAL   W^UM,W^MODE             ; set mode
0127'CF   003F'CF   DE  005E  304           MOVAL   W^SETAST,W^SERV_NAME    ; set service name
                        0065  305           $SETAST_G       W^SET           ; disable user AST's
                        006E  306           FAIL_CHECK SS$_WASSET           ; check for success
      00000000'8F   DD  006E                        PUSHL   #SS$_WASSET
    0B24'CF   01    FB  0074                        CALLS   #1,W^REG_CHECK
0127'CF   0038'CF   DE  0079  307           MOVAL   W^DCLAST,W^SERV_NAME    ; set service name
                        0080  308           $DCLAST_S W^AST1,#-1,#PSL$C_USER ; declare user AST #1
                        0093  309           FAIL_CHECK SS$_NORMAL           ; check for success
      00000000'8F   DD  0093                        PUSHL   #SS$_NORMAL
    0B24'CF   01    FB  0099                        CALLS   #1,W^REG_CHECK
                        009E  310           $DCLAST_G W^DCL                 ; declare user AST #2
                        00A7  311           FAIL_CHECK SS$_NORMAL           ; check for success
      00000000'8F   DD  00A7                        PUSHL   #SS$_NORMAL
    0B24'CF   01    FB  00AD                        CALLS   #1,W^REG_CHECK
0127'CF   003F'CF   DE  00B2  312           MOVAL   W^SETAST,W^SERV_NAME    ; set service name
    0133'CF   01    90  00B9  313           MOVB    #1,W^ASTOK              ; set AST's are legal flag
                        00BE  314           $SETAST_S #1                    ; two AST's for user should occur
                        00C7  315           FAIL_CHECK SS$_NORMAL           ; check for success
      00000000'8F   DD  00C7                        PUSHL   #SS$_NORMAL
    0B24'CF   01    FB  00CD                        CALLS   #1,W^REG_CHECK
                        00D2  316 ;+
                        00D2  317 ;
                        00D2  318 ; test for super AST's
                        00D2  319 ;
                        00D2  320 ;          DECLARED ORDER
                        00D2  321 ; AST ROUTINE    MODE     AST PARAM
```

```
                                                     00D2    322 ; -------------------------------
                                                     00D2    323 ;    AST1        SUPER       -2
                                                     00D2    324 ;    AST2        SUPER       -7
                                                     00D2    325 ;-
                                                     00D2    326         NEXT_TEST
                                                     00D2
                                                     00D2        STP1:
                0004'CF    01    D0    00D2                          MOVL    #1,W^CURRENT_TC
                           00    DD    00D7                          PUSHL   #0
                0B1A'CF    01    FB    00D9                          CALLS   #1,W^REG_SAVE
          00AD'CF    01CA'CF    DE    00DE    327         MOVAL   W^SM,W^MODE             ; set mode
          0127'CF    003F'CF    DE    00E5    328         MOVAL   W^SETAST,W^SERV_NAME    ; set service name
                     0133'CF    94    00EC    329         CLRB    W^ASTOK                 ; AST's are now illegal
                           01    BE    00F0    330         CHMS    #1                      ; test super mode AST's
                                              00F2    331 ;+
                                              00F2    332 ;
                                              00F2    333 ; test for exec AST's
                                              00F2    334 ;
                                              00F2    335 ;          DECLARED ORDER
                                              00F2    336 ; AST ROUTINE   MODE    AST PARAM
                                              00F2    337 ; -------------------------------
                                              00F2    338 ;    AST1        EXEC        -3
                                              00F2    339 ;    AST2        EXEC        -6
                                              00F2    340 ;-
                                              00F2    341         NEXT_TEST
                                              00F2
                                              00F2        STP2:
                0004'CF    02    D0    00F2                          MOVL    #2,W^CURRENT_TC
                           00    DD    00F7                          PUSHL   #0
                0B1A'CF    01    FB    00F9                          CALLS   #1,W^REG_SAVE
          00AD'CF    01D7'CF    DE    00FE    342         MOVAL   W^E1,W^MODE             ; set mode
          0127'CF    003F'CF    DE    0105    343         MOVAL   W^SETAST,W^SERV_NAME    ; set service name
                     0133'CF    94    010C    344         CLRB    W^ASTOK                 ; AST's are now illegal
                                              0110    345         MODE    TO,10$,EXEC,NOREGS      ; set mode to exec
                           00    DD    012D    346         PUSHL   #0                      ; push a dummy parameter
                0B1A'CF    01    FB    012F    347         CALLS   #1,W^REG_SAVE           ; save a register snapshot
                                              0134    348         $SETAST_G W^SET                 ; disable exec AST's
                                              013D    349         FAIL_CHECKNP SS$_WASSET         ; check for success
                 00000000'8F    DD    013D                          PUSHL   #SS$_WASSET
                122D'CF    01    FB    0143                          CALLS   #1,W^REG_CHECKNP
          0127'CF    0038'CF    DE    0148    350         MOVAL   W^DCLAST,W^SERV_NAME    ; set service name
                           00    DD    014F    351         PUSHL   #0                      ; push a dummy parameter
                0B1A'CF    01    FB    0151    352         CALLS   #1,W^REG_SAVE           ; save a register snapshot
                                              0156    353         $DCLAST_S W^AST1,#-3,#PSL$C_EXEC ; declare exec AST #1
                                              0169    354         FAIL_CHECKNP SS$_NORMAL         ; check for success
                 00000000'8F    DD    0169                          PUSHL   #SS$_NORMAL
                122D'CF    01    FB    016F                          CALLS   #1,W^REG_CHECKNP
                     0071'CF    D6    0174    355         INCL    W^DCL+DCLAST$_ASTPRM    ; set new parameter
                     0075'CF    D7    0178    356         DECL    W^DCL+DCLAST$_ACMODE    ; set to exec mode
                           00    DD    017C    357         PUSHL   #0                      ; push a dummy parameter
                0B1A'CF    01    FB    017E    358         CALLS   #1,W^REG_SAVE           ; save a reg snapshot
                                              0183    359         $DCLAST_G W^DCL                 ; declare exec AST #2
                                              018C    360         FAIL_CHECKNP SS$_NORMAL         ; check for success
                 00000000'8F    DD    018C                          PUSHL   #SS$_NORMAL
                122D'CF    01    FB    0192                          CALLS   #1,W^REG_CHECKNP
          0127'CF    003F'CF    DE    0197    361         MOVAL   W^SETAST,W^SERV_NAME    ; set service name
                           00    DD    019E    362         PUSHL   #0                      ; push a dummy parameter
```

```
         0B1A'CF    01    FB  01A0   363         CALLS   #1,W^REG_SAVE           ; save a reg snapshot
         0133'CF    01    90  01A5   364         MOVB    #1,W^ASTOK              ; AST's are now legal
                              01AA   365         $SETAST_S #1                    ; two AST's for exec should occur
                              01B3   366         FAIL_CHECKNP SS$_WASCLR         ; check for success
      00000000'8F         DD  01B3                       PUSHL   #SS$_WASCLR
         122D'CF    01    FB  01B9                       CALLS   #1,W^REG_CHECKNP
                              01BE   367         MODE    FROM,10$                ; set mode back
                              01BF   368 ;+
                              01BF   369 ;
                              01BF   370 ; test for kernel AST's
                              01BF   371 ;
                              01BF   372 ;          DECLARED ORDER
                              01BF   373 ; AST ROUTINE   MODE     AST PARAM
                              01BF   374 ; ---------------------------------
                              01BF   375 ;    AST1       KRNL         -4
                              01BF   376 ;    AST2       KRNL         -5
                              01BF   377 ;-
                              01BF   378         NEXT_TEST
                              01BF
                              01BF       STP3:
         0004'CF    03    D0  01BF                       MOVL    #3,W^CURRENT_TC
                    00    DD  01C4                       PUSHL   #0
         0B1A'CF    01    FB  01C6                       CALLS   #1,W^REG_SAVE
00AD'CF    01E8'CF        DE  01CB   379         MOVAL   W^KM,W^MODE             ; set mode
0127'CF    003F'CF        DE  01D2   380         MOVAL   W^SETAST,W^SERV_NAME    ; set service name
                              01D9   381         MODE    TO,10$,KRNL,NOREGS      ; set mode to kernel
                    00    DD  01F6   382         PUSHL   #0                      ; push a dummy parameter
         0B1A'CF    01    FB  01F8   383         CALLS   #1,W^REG_SAVE           ; save a reg snapshot
                              01FD   384         $SETAST_G W^SET                 ; disable kernel AST's
                              0206   385         FAIL_CHECKNP SS$_WASSET         ; check for success
      00000000'8F         DD  0206                       PUSHL   #SS$_WASSET
         122D'CF    01    FB  020C                       CALLS   #1,W^REG_CHECKNP
0127'CF    0038'CF        DE  0211   386         MOVAL   W^DCLAST,W^SERV_NAME    ; set service name
                    00    DD  0218   387         PUSHL   #0                      ; push a dummy parameter
         0B1A'CF    01    FB  021A   388         CALLS   #1,W^REG_SAVE           ; save a reg snapshot
                              021F   389         $DCLAST_S W^AST1,#-4,#PSL$C_KERNEL
                              0232   390         FAIL_CHECKNP SS$_NORMAL         ; check for success
      00000000'8F         DD  0232                       PUSHL   #SS$_NORMAL
         122D'CF    01    FB  0238                       CALLS   #1,W^REG_CHECKNP
              0071'CF     D6  023D   391         INCL    W^DCL+DCLAST$_ASTPRM    ; set new parameter
              0075'CF     D7  0241   392         DECL    W^DCL+DCLAST$_ACMODE    ; set to kernal mode
                    00    DD  0245   393         PUSHL   #0                      ; push a dummy parameter
         0B1A'CF    01    FB  0247   394         CALLS   #1,W^REG_SAVE           ; save a register snapshot
                              024C   395         $DCLAST_G W^DCL
                              0255   396         FAIL_CHECKNP SS$_NORMAL         ; check for success
      00000000'8F         DD  0255                       PUSHL   #SS$_NORMAL
         122D'CF    01    FB  025B                       CALLS   #1,W^REG_CHECKNP
0127'CF    003F'CF        DE  0260   397         MOVAL   W^SETAST,W^SERV_NAME    ; set service name
                    00    DD  0267   398         PUSHL   #0                      ; push a dummy parameter
         0B1A'CF    01    FB  0269   399         CALLS   #1,W^REG_SAVE           ; save a register snapshot
         0133'CF    01    90  026E   400         MOVB    #1,W^ASTOK              ; set AST's are legal flag
                              0273   401         $SETAST_S #1                    ; two AST's for kernal should occur
                              027C   402         FAIL_CHECKNP SS$_WASCLR         ; check for success
      00000000'8F         DD  027C                       PUSHL   #SS$_WASCLR
         122D'CF    01    FB  0282                       CALLS   #1,W^REG_CHECKNP
                              0287   403         MODE    FROM,10$                ; set mode back
         12A9'CF    00    FB  0288   404         CALLS   #0,W^ERLBUF_DUMP        ; dump any errors
```

```
                                                C 4
                     028D   406          .SBTTL  SETPRA TESTS
                     028D   407 ;+
                     028D   408 ;
                     028D   409 ; SETPRA tests
                     028D   410 ;
                     028D   411 ; test user mode
                     028D   412 ;
                     028D   413 ;-
                     028D   414          NEXT_TEST
                     028D
                     028D          STP4:
     0004'CF    04   D0   028D                  MOVL    #4,W^CURRENT_TC
                00   DD   0292                  PUSHL   #0
     0B1A'CF    01   FB   0294                  CALLS   #1,W^REG_SAVE
0138'CF    0000'CF   DE   0299   415          MOVAL   W^DUMMY,W^HANDLER_ADR   ; set handler address
0127'CF    0054'CF   DE   02A0   416          MOVAL   W^SETPRA,W^SERV_NAME    ; set service name
00AD'CF    01BE'CF   DE   02A7   417          MOVAL   W^UM,W^MODE             ; set mode
                     02AE   418          $SETPRA_S @W^HANDLER_ADR,#PSL$C_KERNEL ; test _S & mode maximizing
                     02BB   419          FAIL_CHECK SS$_NORMAL           ; check for success
      00000000'8F   DD   02BB                  PUSHL   #SS$_NORMAL
     0B24'CF    01   FB   02C1                  CALLS   #1,W^REG_CHECK
                03   DD   02C6   420          PUSHL   #PSL$C_USER             ; set expected mode
     1313'CF    01   FB   02C8   421          CALLS   #1,W^POWER_CHECK        ; check effect
          0138'CF   D6   02CD   422          INCL    W^HANDLER_ADR           ; make address unique
                     02D1   423          $SETPRA_G W^SET2                ; test _G & mode maximizing
                     02DA   424          FAIL_CHECK SS$_NORMAL           ; check for success
      00000000'8F   DD   02DA                  PUSHL   #SS$_NORMAL
     0B24'CF    01   FB   02E0                  CALLS   #1,W^REG_CHECK
                03   DD   02E5   425          PUSHL   #PSL$C_USER             ; set expected mode
     1313'CF    01   FB   02E7   426          CALLS   #1,W^POWER_CHECK        ; check effects
          0138'CF   D6   02EC   427          INCL    W^HANDLER_ADR           ; make address unique
                     02F0   428 ;+
                     02F0   429 ;
                     02F0   430 ; test super mode
                     02F0   431 ;
                     02F0   432 ;-
                     02F0   433          NEXT_TEST
                     02F0
                     02F0          STP5:
     0004'CF    05   D0   02F0                  MOVL    #5,W^CURRENT_TC
                00   DD   02F5                  PUSHL   #0
     0B1A'CF    01   FB   02F7                  CALLS   #1,W^REG_SAVE
00AD'CF    01CA'CF   DE   02FC   434          MOVAL   W^SM,W^MODE             ; set mode
                04   BE   0303   435          CHMS    #4                      ; do the super tests
                     0305   436 ;+
                     0305   437 ;
                     0305   438 ; test exec mode
                     0305   439 ;
                     0305   440 ;-
                     0305   441          NEXT_TEST
                     0305
                     0305          STP6:
     0004'CF    06   D0   0305                  MOVL    #6,W^CURRENT_TC
                00   DD   030A                  PUSHL   #0
     0B1A'CF    01   FB   030C                  CALLS   #1,W^REG_SAVE
00AD'CF    01D7'CF   DE   0311   442          MOVAL   W^EM,W^MODE             ; set mode
                     0318   443          $CMEXEC_S B^10$                 ; get to exec mode
```

```
                                                     D 4
                     22   11  0324    444          BRB     20$
                                 0326    445 10$:
                          0000   0326    446          .WORD   0
                     00   DD    0328    447          PUSHL   #0                      ; push a dummy
           0B1A'CF   01   FB    032A    448          CALLS #1,W^REG_SAVE             ; save a register snapshot
                                 032F    449          $SETPRA_S @W^HANDLER_ADR,#PSL$C_EXEC ; test _S & mode maximizing
                                 033C    450          FAIL_CHECKNP SS$_NORMAL         ; check for success
            00000000'8F   DD    033C                         PUSHL   #SS$_NORMAL
           122D'CF   01   FB    0342                         CALLS   #1,W^REG_CHECKNP
                     04   0347    451          RET                             ; return to user mode
                                 0348    452 20$:
           12A9'CF   00   FB    0348    453          CALLS   #0,W^ERLBUF_DUMP        ; dump any errors
                     01   DD    034D    454          PUSHL   #PSL$C_EXEC             ; set expected mode
           1313'CF   01   FB    034F    455          CALLS   #1,W^POWER_CHECK                ; check effect
                0138'CF   D6    0354    456          INCL    W^HANDLER_ADR           ; make address unique
      0099'CF   0138'CF   D0    0358    457          MOVL    W^HANDLER_ADR,W^SET2+SETPRA$_ASTADR ; set new address
                                 035F    458          $CMEXEC_S B^30$                 ; go back to exec mode
                     1E   11    036B    459          BRB     40$                     ; skip exec routine
                                 036D    460 30$:
                          0000   036D    461          .WORD   0
                     00   DD    036F    462          PUSHL   #0                      ; push a dummy parameter
           0B1A'CF   01   FB    0371    463          CALLS   #1,W^REG_SAVE           ; save a reg snapshot
                                 0376    464          $SETPRA_G W^SET2                ; test _G & mode maximizing
                                 037F    465          FAIL_CHECKNP SS$_NORMAL         ; check for success
            00000000'8F   DD    037F                         PUSHL   #SS$_NORMAL
           122D'CF   01   FB    0385                         CALLS   #1,W^REG_CHECKNP
                     04   038A    466          RET                             ; return to user mode
                                 038B    467 40$:
           12A9'CF   00   FB    038B    468          CALLS #0,W^ERLBUF_DUMP          ; dump any errors
                     01   DD    0390    469          PUSHL   #PSL$C_EXEC             ; set expected mode
           1313'CF   01   FB    0392    470          CALLS   #1,W^POWER_CHECK                ; check effect
                0138'CF   D6    0397    471          INCL    W^HANDLER_ADR           ; make address unique
                                 039B    472 ;+
                                 039B    473 ;
                                 039B    474 ; test kernal mode
                                 039B    475 ;
                                 039B    476 ;-
                                 039B    477          NEXT_TEST
                                 039B
                                 039B        STP7:
           0004'CF   07   D0    039B                         MOVL    #7,W^CURRENT_TC
                     00   DD    03A0                         PUSHL   #0
           0B1A'CF   01   FB    03A2                         CALLS   #1,W^REG_SAVE
      00AD'CF   01E8'CF   DE    03A7    478          MOVAL   W^KM,W^MODE             ; set mode
                                 03AE    479          $CMKRNL_S B^10$                 ; get to kernal mode
                     22   11    03BA    480          BRB     20$                     ; skip over the routine
                                 03BC    481 10$:
                          0000   03BC    482          .WORD   0                       ; entry mask
                     00   DD    03BE    483          PUSHL   #0                      ; push a dummy
           0B1A'CF   01   FB    03C0    484          CALLS #1,W^REG_SAVE             ; save a register snapshot
                                 03C5    485          $SETPRA_S @W^HANDLER_ADR,#PSL$C_KERNEL ; test _S form
                                 03D2    486          FAIL_CHECKNP SS$_NORMAL         ; check success
            00000000'8F   DD    03D2                         PUSHL   #SS$_NORMAL
           122D'CF   01   FB    03D8                         CALLS   #1,W^REG_CHECKNP
                     04   03DD    487          RET                             ; return to user mode
                                 03DE    488 20$:
           12A9'CF   00   FB    03DE    489          CALLS #0,W^ERLBUF_DUMP          ; dump any errors
```

```
                    00    DD  03E3    490          PUSHL   #PSL$C_KERNEL           ; set expected mode
          1313'CF   01    FB  03E5    491          CALLS   #1,W^POWER_CHECK        ; check effect
             0138'CF    D6  03EA    492          INCL    W^HANDLER_ADR           ; make address unique
0099'CF      0138'CF    D0  03EE    493          MOVL    W^HANDLER_ADR,W^SET2+SETPRA$_ASTADR ; set new address
                        03F5    494          $CMKRNL_S B^30$                 ; return to kernel mode
                    17  11  0401    495          BRB     40$                     ; skip over kernel routine
                        0403    496 30$:
                  0000  0403    497          .WORD   0
                        0405    498          $SETPRA_G W^SET2                ; test _G
                        040E    499          FAIL_CHECKNP SS$_NORMAL         ; check success
                                                     PUSHL   #SS$_NORMAL
           00000000'8F  DD  040E
          122D'CF   01  FB  0414                     CALLS   #1,W^REG_CHECKNP
                    04  0419    500          RET                             ; return to user mode
                        041A    501 40$:
          12A9'CF   00  FB  041A    502          CALLS #0,W^ERLBUF_DUMP          ; dump any errors
                    00  DD  041F    503          PUSHL   #PSL$C_KERNEL           ; set expected mode
          1313'CF   01  FB  0421    504          CALLS   #1,W^POWER_CHECK        ; check effect
```

```
                                0426    506          .SBTTL  SETIMR TESTS
                                0426    507 ;+
                                0426    508 ;
                                0426    509 ; SETIMR tests
                                0426    510 ;
                                0426    511 ; test EFN #33, real time, ID=6, 6 sec, _S
                                0426    512 ;
                                0426    513 ;-
                                0426    514          NEXT_TEST
                                0426
                                0426        STP8:
            0004'CF    08   D0  0426                   MOVL    #8,W^CURRENT_TC
                       00   DD  042B                   PUSHL   #0
            0B1A'CF    01   FB  042D                   CALLS   #1,W^REG_SAVE
      00AD'CF   01BE'CF   DE   0432    515          MOVAL   W^UM,W^MODE             ; set the mode
      0127'CF   0046'CF   DE   0439    516          MOVAL   W^SETIMR,W^SERV_NAME    ; set the service name
                                0440    517          $GETTIM_S TIMADR=TIME           ; get the current time
  013C'CF   020E'CF   06   C5   044D    518          MULL3   #6,W^ONE_SEC,W^X_SEC    ; make 6 seconds
      0140'CF   013C'CF   C0   0455    519          ADDL2   W^X_SEC,@^TIME          ; add it to the current time
      0144'CF   00        D8   045C    520          ADWC    #0,@^TIME+4             ; include the carry bit if any
                                0461    521          $SETIMR_S EFN=#33,-
                                0461    522                    DAYTIM=W^TIME,-
                                0461    523                    REQIDT =#6           ; do it
                                0472    524          FAIL_CHECK SS$_NORMAL           ; check for success
             00000000'8F   DD   0472                   PUSHL   #SS$_NORMAL
            0B24'CF    01   FB  0478                   CALLS   #1,W^REG_CHECK
                                047D    525 ;+
                                047D    526 ;
                                047D    527 ; test def EFN, delta time, ID=5, 5 sec, _G
                                047D    528 ;
                                047D    529 ;-
                                047D    530          NEXT_TEST
                                047D
                                047D        STP9:
      0004'CF    09   D0  047D                   MOVL    #9,W^CURRENT_TC
                       00   DD  0482                   PUSHL   #0
      0B1A'CF    01   FB  0484                   CALLS   #1,W^REG_SAVE
0089'CF   0212'CF   DE   0489    531          MOVAL   W^DELTA_5_SEC,W^SET1+SETIMR$_DAYTIM ; set 5 sec delta
                                0490    532          $SETIMR_G W^SETT                ; do it
                                0499    533          FAIL_CHECK SS$_NORMAL           ; check success
             00000000'8F   DD   0499                   PUSHL   #SS$_NORMAL
      0B24'CF    01   FB  049F                   CALLS   #1,W^REG_CHECK
                                04A4    534 ;+
                                04A4    535 ;
                                04A4    536 ; test EFN=4, delta time, ID=4, 4 sec, _S
                                04A4    537 ;
                                04A4    538 ;-
                                04A4    539          NEXT_TEST
                                04A4
                                04A4        STP10:
      0004'CF    0A   D0  04A4                   MOVL    #10,W^CURRENT_TC
                       00   DD  04A9                   PUSHL   #0
      0B1A'CF    01   FB  04AB                   CALLS   #1,W^REG_SAVE
                                04B0    540          $SETIMR_S DAYTIM=W^DELTA_4_SEC,-
                                04B0    541                    EFN    =#4,-
                                04B0    542                    REQIDT =#4           ; do it
                                04C1    543          FAIL_CHECK SS$_NORMAL           ; check success
```

```
         00000000'8F     DD  04C1                          PUSHL   #SS$_NORMAL
            0B24'CF  01  FB  04C7                          CALLS   #1,W^REG_CHECK
                             04CC      ;+
                             04CC      ;
                             04CC      ; test EFN=3, real time, ID=3, 3 sec, _G
                             04CC      ;
                             04CC      ;-
                             04CC              NEXT_TEST
                             04CC
                             04CC      STP11:
            0004'CF  0B  D0  04CC                          MOVL    #11,W^CURRENT_TC
                     00  DD  04D1                          PUSHL   #0
            0B1A'CF  01  FB  04D3                          CALLS   #1,W^REG_SAVE
 013C'CF    020E'CF  03  C5  04D8              MULL3   #3,W^ONE_SEC,W^X_SEC      ; make 3 seconds
     0140'CF    013C'CF  C2  04E0              SUBL2   W^X_SEC,@^TIME            ; deduct it from 6 seconds from now
            0144'CF  00  D9  04E7              SBWC    #0,@^TIME+4               ; subtract carry if any
     0089'CF    0140'CF  DE  04EC              MOVAL   W^TIME,W^SET1+SETIMR$_DAYTIM ; set the time
            0085'CF  03  D0  04F3              MOVL    #3,W^SET1+SETIMR$_EFN     ; set the EFN
            0091'CF  03  D0  04F8              MOVL    #3,W^SET1+SETIMR$_REQIDT ; set the ID
                             04FD              $SETIMR_G W^SET1                  ; do it
                             0506              FAIL_CHECK SS$_NORMAL             ; check success
         00000000'8F     DD  0506                          PUSHL   #SS$_NORMAL
            0B24'CF  01  FB  050C                          CALLS   #1,W^REG_CHECK
                             0511      ;+
                             0511      ;
                             0511      ; test EFN=2, ASTADR, real time, ID=2, 2 sec, _S
                             0511      ;
                             0511      ;-
                             0511              NEXT_TEST
                             0511
                             0511      STP12:
            0004'CF  0C  D0  0511                          MOVL    #12,W^CURRENT_TC
                     00  DD  0516                          PUSHL   #0
            0B1A'CF  01  FB  0518                          CALLS   #1,W^REG_SAVE
     0140'CF    020E'CF  C2  051D              SUBL2   W^ONE_SEC,W^TIME          ; make it 2 sec from now
            0144'CF  00  D9  0524              SBWC    #0,W^TIME+4               ; subtract carry if any
                             0529              $SETAST_S #0                      ; if an error would occur it
                             0532                                                ; it would take longer to print
                             0532                                                ; than 2 sec and would be intr.
                             0532                                                ; so hold back the intr.
                             0532              $SETIMR_S DAYTIM=W^TIME,-
                             0532                      EFN    =#2,-
                             0532                      ASTADR =W^AST4,-
                             0532                      REQIDT =#2                ; do it
                             0545              FAIL_CHECK SS$_NORMAL             ; check success
         00000000'8F     DD  0545                          PUSHL   #SS$_NORMAL
            0B24'CF  01  FB  054B                          CALLS   #1,W^REG_CHECK
                             0550              $SETAST_S #1                      ; OK it's safe for intr.'s
                             0559      ;+
                             0559      ;
                             0559      ; test EFN=1, ASTADR, delta time, ID=1, 1 sec _G
                             0559      ;
                             0559      ;-
                             0559              NEXT_TEST
                             0559
                             0559      STP13:
            0004'CF  0D  D0  0559                          MOVL    #13,W^CURRENT_TC
```

```
                   00   DD  055E                 PUSHL   #0
          0B1A'CF  01   FB  0560                 CALLS   #1,W^REG_SAVE
          0085'CF  01   D0  0565          MOVL   #1,W^SET1+SETIMR$_EFN    ; set EFN
    0089'CF   0222'CF   DE  056A          MOVAL  W^DELTA_1_SEC,W^SET1+SETIMR$_DAYTIM ; set time
    008D'CF   05A6'CF   DE  0571          MOVAL  W^AST3,W^SET1+SETIMR$_ASTADR ; set AST address
          0091'CF  01   D0  0578          MOVL   #1,W^SET1+SETIMR$_REQIDT ; set the ID
                            057D          $SETAST_S #0                    ; if an error would occur it
                            0586                                          ; it would take longer to print
                            0586                                          ; than 1 sec and would be intr.
                            0586                                          ; so hold back the intr.
                            0586          $SETIMR_G W^SET1                ; do it
                            058F          FAIL_CHECK SS$_NORMAL           ; check success
          00000000'8F   DD  058F                 PUSHL   #SS$_NORMAL
          0B24'CF  01   FB  0595                 CALLS   #1,W^REG_CHECK
                            059A          $SETAST_S #1                    ; OK it's safe to intr.
               0092     31  05A3          BRW    NEXT_STEP                ; skip the AST routines
                            05A6     ;+
                            05A6     ;
                            05A6     ; At this time there should be the following TQE's to be serviced
                            05A6     ; in the following order:
                            05A6     ;
                            05A6     ;        FORM    TIME    ID#     HOW SERVICED
                            05A6     ;
                            05A6     ;        _G      1 SEC   1       AST3
                            05A6     ;        _S      2 SEC   2       AST4
                            05A6     ;        _G      3 SEC   3       EFN 3
                            05A6     ;        _S      4 SEC   4       EFN 4
                            05A6     ;        _G      5 SEC   5       EFN 0
                            05A6     ;        _S      6 SEC   6       EFN 0
                            05A6     ;
                            05A6     ; The next 6 steps will service the TQE's and check the results.
                            05A6     ;
                            05A6     ; test servicing of AST TQE started with ID=#1
                            05A6     ;
                            05A6     ;-
                            05A6
                            05A6     AST3:
                      0000  05A6          .WORD  0
 00000008'EF   00000004'EF  D0  05A8      MOVL   CURRENT_TC,PENDING_TC    ; save interrupted test case number
                            05B3          NEXT_TEST
                            05B3
                            05B3          STP14:
          0004'CF  0E   D0  05B3                 MOVL    #14,W^CURRENT_TC
                   00   DD  05B8                 PUSHL   #0
          0B1A'CF  01   FB  05BA                 CALLS   #1,W^REG_SAVE
          0148'CF  01   D0  05BF          MOVL   #1,W^ID                  ; set AST id flag
            01   04 AC  D1  05C4          CMPL   4(AP),#1                 ; correct?
                   0E   13  05C8          BEQL   10$                      ; br if yes
                04 AC   DD  05CA          PUSHL  4(AP)                    ; push received
                   01   DD  05CD          PUSHL  #1                       ; push expected
              0195'CF   DF  05CF          PUSHAL W^EXP                    ; push string variable
          0B88'CF  03   FB  05D3          CALLS  #3,W^PRINT_FAIL          ; print the error
                            05D8     10$:
 00000004'EF   00000008'EF  D0  05D8      MOVL   PENDING_TC,CURRENT_TC    ; restore test case number
                   04       05E3          RET                             ; return
                            05E4     ;+
                            05E4     ;
```

```
                                05E4    630 ; test servicing of AST TQE started with ID #2
                                05E4    631 ;
                                05E4    632 ;-
                                05E4    633 AST4:
                         0000   05E4    634             .WORD   0
00000008'EF   00000004'EF   D0  05E6    635             MOVL    CURRENT_TC,PENDING_TC   ; save interrupted test case number
                                05F1    636             NEXT_TEST
                                05F1
                                05F1        STP15:
                0004'CF    0F   D0  05F1                        MOVL    #15,W^CURRENT_TC
                             00   DD  05F6                        PUSHL   #0
                0B1A'CF    01   FB  05F8                        CALLS   #1,W^REG_SAVE
                01   0148'CF    D1  05FD    637             CMPL    W^ID,#1                 ; has the previous AST occured?
                          0F   13  0602    638             BEQL    10$                     ; br if yes
                     0148'CF   DD  0604    639             PUSHL   W^ID                    ; push received
                          01   DD  0608    640             PUSHL   #1                      ; push expected
                     0195'CF   DF  060A    641             PUSHAL  W^EXP                   ; push the string variable
                0B88'CF    03   FB  060E    642             CALLS   #3,W^PRINT_FAIL         ; print the failure
                                0613    643 10$:
                0148'CF    02   D0  0613    644             MOVL    #2,W^ID                 ; set AST ID flag
                  02   04 AC   D1  0618    645             CMPL    4(AP),#2                ; is this the correct one?
                          0E   13  061C    646             BEQL    20$                     ; br if yes
                        04 AC   DD  061E    647             PUSHL   4(AP)                   ; push received
                          02   DD  0621    648             PUSHL   #2                      ; push expected
                     0195'CF   DF  0623    649             PUSHAL  W^EXP                   ; push string variable
                0B88'CF    03   FB  0627    650             CALLS   #3,W^PRINT_FAIL         ; print the failure
                                062C    651 20$:
00000004'EF   00000008'EF   D0  062C    652             MOVL    PENDING_TC,CURRENT_TC   ; restore test case number
                             04  0637    653             RET                             ; return
                                0638    654 ;+
                                0638    655 ;
                                0638    656 ; test TQE with ID of 3
                                0638    657 ;
                                0638    658 ;-
                                0638    659 NEXT_STEP:
                                0638    660             NEXT_TEST
                                0638
                                0638        STP16:
0004'CF    10   D0  0638                        MOVL    #16,W^CURRENT_TC
             00   DD  063D                        PUSHL   #0
0B1A'CF    01   FB  063F                        CALLS   #1,W^REG_SAVE
                                0644    661             $WAITFR_S EFN=#3                ; wait here for AST3, AST4, & EFN 3
02   0148'CF    D1  064D    662             CMPL    W^ID,#2                 ; did both AST's occur?
          0F   13  0652    663             BEQL    10$                     ; br if yes
     0148'CF   DD  0654    664             PUSHL   W^ID                    ; push received
          02   DD  0658    665             PUSHL   #2                      ; push expected
     0195'CF   DF  065A    666             PUSHAL  W^EXP                   ; push string variable
0B88'CF    03   FB  065E    667             CALLS   #3,W^PRINT_FAIL         ; print the failure
                                0663    668 10$:
                                0663    669 ;+
                                0663    670 ;
                                0663    671 ; wait for TQE with ID of 4
                                0663    672 ;
                                0663    673 ;-
                                0663    674             NEXT_TEST
                                0663
                                0663        STP17:
```

```
                0004'CF    11    D0  0663                        MOVL    #17,W^CURRENT_TC
                           00    DD  0668                        PUSHL   #0
                0B1A'CF    01    FB  066A                        CALLS   #1,W^REG_SAVE
                                     066F    675         $WAITFR_S EFN=#4                    ; wait for #4
                                     0678    676 ;+
                                     0678    677 ;
                                     0678    678 ; wait for TQE with ID of 5
                                     0678    679 ;
                                     0678    680 ;-
                                     0678    681         NEXT_TEST
                                     0678
                                     0678        STP18:
                0004'CF    12    D0  0678                        MOVL    #18,W^CURRENT_TC
                           00    DD  067D                        PUSHL   #0
                0B1A'CF    01    FB  067F                        CALLS   #1,W^REG_SAVE
                                     0684    682         $WAITFR_S EFN=#0                    ; wait for #5
                                     068D    683         $CLREF_S EFN=#0                     ; get ready for #6
                                     0696    684 ;+
                                     0696    685 ;
                                     0696    686 ; wait for TQE with ID of 6
                                     0696    687
                                     0696    688 ;
                                     0696    689 ;-
                                     0696    690         NEXT_TEST
                                     0696
                                     0696        STP19:
                0004'CF    13    D0  0696                        MOVL    #19,W^CURRENT_TC
                           00    DD  069B                        PUSHL   #0
                0B1A'CF    01    FB  069D                        CALLS   #1,W^REG_SAVE
                                     06A2    691         $WAITFR_S EFN=#33                   ; wait for #6
```

```
                        06AB   693          .SBTTL CANTIM TESTS
                        06AB   694 ;+
                        06AB   695 ;
                        06AB   696 ; CANTIM tests
                        06AB   697 ;
                        06AB   698 ; test user mode
                        06AB   699 ;
                        06AB   700 ;-
                        06AB   701          NEXT_TEST
                        06AB
                        06AB        STP20:
    0004'CF    14   DO  06AB                   MOVL    #20,W^CURRENT_TC
               00   DD  06B0                   PUSHL   #0
    0B1A'CF    01   FB  06B2                   CALLS   #1,W^REG_SAVE
0127'CF 004D'CF    DE  06B7   702          MOVAL   W^CANTIM,W^SERV_NAME    ; set the service name
    52 0081'CF    DE  06BE   703          MOVAL   W^SET1,R2               ; set SETIMR param. pointer
       1374'CF    DE  06C3   704          MOVAL   W^ILEGAL_TIM,-
         0C A2         06C7   705                  B^SETIMR$_ASTADR(R2)    ; set AST address
       0222'CF    DE  06C9   706          MOVAL   W^DELTA_1_SEC,-
         08 A2         06CD   707                  B^SETIMR$_DAYTIM(R2)    ; set one sec delta time
               00   DD  06CF   708          PUSHL   #0                      ; push a dummy parameter
    0B1A'CF    01   FB  06D1   709          CALLS   #1,W^REG_SAVE           ; save a register snapshot
                        06D6   710          $SETIMR_G (R2)                  ; make a TQE
                        06DD   711          $CANTIM_S REQIDT=#1             ; scrap it
                        06E8   712          FAIL_CHECK SS$_NORMAL           ; check success
     00000000'8F   DD  06E8                   PUSHL   #SS$_NORMAL
    0B24'CF    01   FB  06EE                   CALLS   #1,W^REG_CHECK
                        06F3   713 ;+
                        06F3   714 ;
                        06F3   715 ; test cancelling 2 TQE's in user mode
                        06F3   716 ;
                        06F3   717 ;-
                        06F3   718          NEXT_TEST
                        06F3
                        06F3        STP21:
0004'CF    15   DO  06F3                   MOVL    #21,W^CURRENT_TC
               00   DD  06F8                   PUSHL   #0
0B1A'CF    01   FB  06FA                   CALLS   #1,W^REG_SAVE
0148'CF    01   DO  06FF   719          MOVL    #1,W^ID                 ; set the proper ID
                        0704   720          $SETIMR_G (R2)                  ; make #1 TQE
                        070B   721          $SETIMR_G (R2)                  ; make #2 TQE
                        0712   722          $CANTIM_G W^CAN                 ; try it
                        071B   723          FAIL_CHECK SS$_NORMAL           ; check for success
     00000000'8F   DD  071B                   PUSHL   #SS$_NORMAL
0B24'CF    01   FB  0721                   CALLS   #1,W^REG_CHECK
  10 A2    03   DO  0726   724          MOVL    #3,B^SETIMR$_REQIDT(R2) ; set ID to 3
                        072A   725          $SETIMR_G (R2)                  ; set a TQE for later
  10 A2    01   DO  0731   726          MOVL    #1,B^SETIMR$_REQIDT(R2) ; reset ID to 1
                        0735   727 ;+
                        0735   728 ;
                        0735   729 ; test super mode
                        0735   730 ;
                        0735   731 ;-
                        0735   732          NEXT_TEST
                        0735
                        0735        STP22:
0004'CF    16   DO  0735                   MOVL    #22,W^CURRENT_TC
```

```
                    00    DD  073A                        PUSHL   #0
          0B1A'CF   01    FB  073C                        CALLS   #1,W^REG_SAVE
00AD'CF   01CA'CF   DE  0741    733             MOVAL   W^SM,W^MODE             ; set the mode
                    05    BE  0748    734             CHMS    #5                      ; do super mode tests
                              074A    735 ;+
                              074A    736 ;
                              074A    737 ; test cancelling 1 exec TQE
                              074A    738 ;
                              074A    739 ;-
                 00000018     074A    740             STEP=STEP+2
                              074A    741             NEXT_TEST
                              074A
                              074A        STP25:
          0004'CF   19    D0  074A                        MOVL    #25,W^CURRENT_TC
                    00    DD  074F                        PUSHL   #0
          0B1A'CF   01    FB  0751                        CALLS   #1,W^REG_SAVE
00AD'CF   01D7'CF   DE  0756    742             MOVAL   W^EM,W^MODE             ; set the mode
                        075D    743             $CMEXEC_S B^10$                 ; get to EXEC mode
                  0071  31  0769    744             BRW A20                         ; skip over EXEC routine
                        076C    745 10$:
                  0000  076C    746             .WORD   ^M<>
          014C'CF   52    D0  076E    747             MOVL    R2,W^WORK               ; save R2
          52    0081'CF   DE  0773    748             MOVAL   W^SET1,R2               ; set SETIMR param pointer
                        0778    749             $SETIMR_G (R2)                  ; make a TQE
                        077F    750             $CANTIM_S REQIDT=#1,-
                        077F    751                       ACMODE=#PSL$C_EXEC    , can it
                        078A    752             FAIL_CHECKNP SS$_NORMAL         ; check for success
           00000000'8F  DD  078A                        PUSHL   #SS$_NORMAL
          122D'CF   01    FB  0790                        CALLS   #1,W^REG_CHECKNP
                        0795    753 ;+
                        0795    754 ;
                        0795    755 ; test cancelling 2 exec TQE
                        0795    756 ;
                        0795    757 ;-
                        0795    758             NEXT_TEST
                        0795
                        0795        STP26:
0004'CF   1A    D0  0795                        MOVL    #26,W^CURRENT_TC
                    00    DD  079A                        PUSHL   #0
0B1A'CF   01    FB  079C                        CALLS   #1,W^REG_SAVE
                        07A1    759             $SETIMR_G (R2)                  ; make a TQE
                        07A8    760             $SETIMR_G (R2)                  ; make another
                    01    D0  07AF    761             MOVL    #PSL$C_EXEC,-
          00A9'CF           07B1    762                     W^CAN+CANTIM$_ACMODE    ; set the mode
                        07B4    763             $CANTIM_G W^CAN                 ; can 2 at once
                        07BD    764             FAIL_CHECKNP SS$_NORMAL         ; check for success
           00000000'8F  DD  07BD                        PUSHL   #SS$_NORMAL
          122D'CF   01    FB  07C3                        CALLS   #1,W^REG_CHECKNP
            10 A2   03    D0  07C8    765             MOVL    #3,B^SETIMR$_REQIDT(R2) ; set ID to 3
                        07CC    766             $SETIMR_G (R2)                  ; set one for later
            10 A2   01    D0  07D3    767             MOVL    #1,B^SETIMR$_REQIDT(R2) ; reset ID to 1
52    014C'CF   D0  07D7    768             MOVL    W^WORK,R2               ; restore R2
                    04    07DC    769             RET                             ; return to user mode
                        07DD    770 A20:
12A9'CF   00    FB  07DD    771             CALLS #0,W^ERLBUF_DUMP          ; dump any errors
                        07E2    772 ;+
                        07E2    773 ;
```

```
                                                07E2   774 ; test cancelling one kernel mode TQE
                                                07E2   775 ;
                                                07E2   776 ;-
                                                07E2   777          NEXT_TEST
                                                07E2
                                                07E2       STP27:
            0004'CF    1B   D0                  07E2                MOVL    #27,W^CURRENT_TC
                       00   DD                  07E7                PUSHL   #0
            0B1A'CF    01   FB                  07E9                CALLS   #1,W^REG_SAVE
      00AD'CF   01E8'CF   DE                    07EE   778          MOVAL   W^KM,W^MODE             ; set the mode
                                                07F5   779          MODE TO,B10,KRNL,NOREGS         ; get to kernel mode
            014C'CF    52   D0                  0812   780          MOVL    R2,W^WORK               ; save R2
            52    0081'CF   DE                  0817   781          MOVAL   W^SET1,R2               ; set SETIMR param pointer
                                                081C   782          $SETIMR_G (R2)                  ; make a kernel TQE
                                                0823   783          $CANTIM_S REQIDT=#1,-
                                                082?   784                   ACMODE=#PSL$C_KERNEL  ; can it
                                                082E   785          FAIL_CHECKNP SS$_NORMAL         ; check for success
             00000000'8F   DD                   082E                PUSHL   #SS$_NORMAL
            122D'CF    01   FB                  0834                CALLS   #1,W^REG_CHECKNP
                                                0839   786 ;+
                                                0839   787 ;
                                                0839   788 ; test cancelling 2 kernel TQE's
                                                0839   789 ;
                                                0839   790 ;-
                                                0839   791          NEXT_TEST
                                                0839
                                                0839       STP28:
            0004'CF    1C   D0                  0839                MOVL    #28,W^CURRENT_T
                       00   DD                  083E                PUSHL   #0
            0B1A'CF    01   FB                  0840                CALLS   #1,W^REG_SAVE
                                                0845   792          $SETIMR_G (R2)                  ; make a kernel TQE
                                                084C   793          $SETIMR_G (R2)                  ; and another
                       00   D0                  0853   794          MOVL    #PSL$C_KERNEL,-
                 00A9'CF                        0855   795                  W^CAN+CANTIM$_ACMODE     ; set access mode
                                                0858   796          $CANTIM_G W^CAN                 ; cancel 2 kernel TQE's
                                                0861   797          FAIL_CHECKNP SS$_NORMAL         ; check for success
             00000000'8F   DD                   0861                PUSHL   #SS$_NORMAL
            122D'CF    01   FB                  0867                CALLS   #1,W^REG_CHECKNP
              10 A2    03   D0                  086C   798          MOVL    #3,SETIMR$_REQIDT(R2)   ; set ID to 3
                                                0870   799          $SETIMR_G (R2)                  ; make a kernel TQE
                                                0877   800 ;+
                                                0877   801 ;
                                                0877   802 ; At this point there should be the following TQE's still
                                                0877   803 ; waiting to be killed:
                                                0877   804 ;
                                                0877   805 ;       1       user    TQE     ID = 3
                                                0877   806 ;       1       super   TQE     ID = 3
                                                0877   807 ;       1       exec    TQE     ID = 3
                                                0877   808 ;       1       kernel  TQE     ID = 3
                                                0877   809 ;
                                                0877   810 ; We will now attempt to cancel all of these TQE's with
                                                0877   811 ; 1 CANTIM service.
                                                0877   812 ;
                                                0877   813 ;-
                                                0877   814          NEXT_TEST
                                                0877
                                                0877       STP29:
```

```
                0004'CF    1D   DO  0877                        MOVL    #29,W^CURRENT_TC
                           00   DD  087C                        PUSHL   #0
                0B1A'CF    01   FB  087E                        CALLS   #1,W^REG_SAVE
                                    0883    815         $CANTIM_S REQIDT= #3,-
                                    0883    816                   ACMODE= #PSL$C_KERNEL ; clean em out!
                                    088E    817         FAIL_CHECKNP SS$_NORMAL         ; check for success
                 00000000'8F    DD  088E                        PUSHL   #SS$_NORMAL
                122D'CF    01   FB  0894                        CALLS   #1,W^REG_CHECKNP
                52    014C'CF    DO  0899    818         MOVL    W^WORK,R2               ; restore R2
                                    089E    819         MODE FROM,B10,NOREGS            ; return to user mode
                12A9'CF    00   FB  089F    820         CALLS   #0,W^ERLBUF_DUMP        ; dump any errors
                                    08A4    821 ;+
                                    08A4    822 ;
                                    08A4    823 ; We now have to wait 1 second to see if any TQE's are still pending
                                    08A4    824 ; that CANTIM's should have canceled.
                                    08A4    825 ;
                                    08A4    826 ;-
        0127'CF    0046'CF    DE   08A4    827         MOVAL   W^SETIMR,W^SERV_NAME    ; set service name
                                    08AB    828         $SETIMR_S DAYTIM=W^DELTA_1_SEC,-
                                    08AB    829                   EFN   = #1            ; wait a second for illegal TQE's
                                    08BA    830         FAIL_CHECK SS$_NORMAL           ; check for success
                 00000000'8F    DD  08BA                        PUSHL   #SS$_NORMAL
                0B24'CF    01   FB  08C0                        CALLS   #1,W^REG_CHECK
                                    08C5    831         $WAITFR_S EFN=#1                ; wait here
                                    08CE    832 ;+
                                    08CE    833 ;
                                    08CE    834 ; reset super mode handler to the original address and
                                    08CE    835 ; dump any errors on the terminal that occured at AST disable time.
                                    08CE    836 ;
                                    08CE    837 ;-
                                    08CE    838 CLEAN_UP:
        0127'CF    0031'CF    DE   08CE    839         MOVAL   W^DCLCMH,W^SERV_NAME    ; set service name
                           03   BE  08D5    840         CHMS    #3                      ; reset the CHMS handler
                12A9'CF    00   FB  08D7    841         CALLS #0,W^ERLBUF_DUMP          ; dump any errors
                                    08DC    842         TEST_END
                     0050'CF    DD  08DC                        PUSHL   W^TMD_ADDR
                     004C'CF    DD  08E0                        PUSHL   W^TMN_ADDR
                           02   DD  08E4                        PUSHL   #2
                     0048'CF    DD  08E6                        PUSHL   W^MOD_MSG_CODE
                00000000'GF     04  FB  08EA                    CALLS   #$$T1,G^LIB$SIGNAL
0048'CF    01    1C    01   F0  08F1                        INSV    #1,#STS$V_INHIB_MSG,#1,W^MOD_MSG_CODE
                     0048'CF    DD  08F8                        PUSHL   W^MOD_MSG_CODE
                00000000'GF     01  FB  08FC                    CALLS   #1,G^SYS$EXIT
```

```
0903    844 .SBTTL  ROUTINES
0903    845 .SBTTL SETUP_SUPER ROUTINE
0903    846 ;++
0903    847 ;
0903    848 ;       Routine to declare an initial CHMS handler from user mode.
0903    849 ;
0903    850 ; FUNCTIONAL DESCRIPTION:
0903    851 ;
0903    852 ; CALLING SEQUENCE:
0903    853 ;
0903    854 ;       $CMKRNL_S W^SETUP_SUPER,ARGLST
0903    855 ;
0903    856 ;               ARGLST = address of a pointer to a one parameter argument list conta
0903    857 ;                        the address of the entry mask of the CHMS handler
0903    858 ;
0903    859 ; INPUT PARAMETERS:
0903    860 ;
0903    861 ;       ARGLST
0903    862 ;
0903    863 ; IMPLICIT INPUTS
0903    864 ;
0903    865 ;       NONE
0903    866 ;
0903    867 ; OUTPUT PARAMETERS:
0903    868 ;
0903    869 ;       Declares a change mode handler for super mode which must be
0903    870 ;       reset to DCL in the users handler routine when the handler is
0903    871 ;       no longer needed.
0903    872 ;
0903    873 ; IMPLICIT OUTPUTS:
0903    874 ;
0903    875 ;       NONE
0903    876 ;
0903    877 ; COMPLETION CODES:
0903    878 ;
0903    879 ;       NONE
0903    880 ;
0903    881 ; SIDE EFFECTS:
0903    882 ;
0903    883 ;       NONE
0903    884 ;
0903    885 ; ON ENTRY:
0903    886 ;
0903    887 ;                   KSP =>  ---------          USP =>  ---------
0903    888 ;                           |   0   |                  |       |
0903    889 ;                           |   0   |                  |  USER |
0903    890 ;                           |  AP   |                  |       |
0903    891 ;                           |  FP   |                  |  CALL |
0903    892 ;                           |  PC   |                  |       |
0903    893 ;                           |   0   |                  | FRAME |
0903    894 ;                           |   0   |                  |       |
0903    895 ;                           |  AP   |                  ---------
0903    896 ;                           |  FP   |
0903    897 ;                           |SRVEXIT|
0903    898 ;                           |  PC   |
0903    899 ;                           |  PSL  |
0903    900 ;--                         ---------
```

```
                                0903   902 RETURN_PC:
                    00000000    0903   903         .LONG   0                       ; storage for user return PC
                                0907   904 HANDLER_PC:
                    00000000    0907   905         .LONG   0                       ; storage for handler PC
                                090B   906 ;
                                090B   907 SETUP_SUPER:
                        000C    090B   908         .WORD   ^M<R2,R3>
                53      03  DB  090D   909         MFPR    #PR$_USP,R3             ; get the user call frame address
EE AF             10    A3  D0  0910   910         MOVL    SF$L_SAVE_PC(R3),B^RETURN_PC ; get the user return PC
ED AF             04    AC  D0  0915   911         MOVL    4(AP),HANDLER_PC        ; save the handler address
                52      0C  AD  D0  091A   912         MOVL    SF$L_SAVE_FP(FP),R2     ; get saved FP
                52      00' C0  091E   913         ADDL    S^#EXE$C_CMSTKSZ,R2     ; back over change mode stack frame
                62    2F'AF 9E  0921   914         MOVAB   B^20$,(R2)              ; set return address
                        0A  F0  0925   915         INSV    #<<PSL$C_SUPER@PSL$S_CURMOD>+PSL$C_SUPER>,-
                        16      0927   916                 #PSL$V_PRVMOD,-
                04 A2   04      0928   917                 #PSL$S_CURMOD*2,4(R2)   ; set current and previous mode to super
                50      00' D0  092B   918         MOVL    S^#SS$_NORMAL,R0        ; set correct return code
                        04      092E   919         RET                             ; enter super mode
                                092F   920 20$:
                        7E  D4  092F   921         CLRL    -(SP)                   ; set up dummy PSL
                35'AF   6E  FA  0931   922         CALLG   (SP),B^30$              ; create initial call frame
                                0935   923 30$:
                        0000    0935   924         .WORD   ^M<>                    ; entry mask
                        00  DD  0937   925         PUSHL   #0                      ; push a dummy parameter
             0B1A'CF    01  FB  0939   926         CALLS   #1,W^REG_SAVE           ; save the registers
                                093E   927         $DCLCMH_S @HANDLER_PC,W^PRVHND1,#0 ; set real handler
                                094E   928         FAIL_CHECKNP SS$_NORMAL         ; check for success
              00000000'8F   DD  094E                        PUSHL   #SS$_NORMAL
            122D'CF     01  FB  0954                        CALLS   #1,W^REG_CHECKNP
             03C00000   8F  DD  0959   929         PUSHL   #<<PSL$C_USER@PSL$V_CURMOD>-
                                095F   930                 !<PSL$C_USER@PSL$V_PRVMOD>>; set return to user
                    A1  AF  DD  095F   931         PUSHL   RETURN_PC               ; set the return PC
                        02  DD  0962   932         REI                             ; return to user mode
                                0963   933         .SBTTL  AST SERVICE
                                0963   934 ;++
                                0963   935 ; FUNCTIONAL DESCRIPTION:
                                0963   936 ;       Routine to service AST's for the DCLAST service tests.
                                0963   937 ;
                                0963   938 ; CALLING SEQUENCE:
                                0963   939 ;       AST delivery
                                0963   94- ;
                                0963   941 ; INPUT PARAMETERS:
                                0963   942 ;       4(AP) = AST parameter
                                0963   943 ;       ASTOK = AST flag byte
                                0963   944 ;       ASTPAR1, ASTPAR2 = expected AST parameters
                                0963   945 ;       ELBP  = error log buffer pointer
                                0963   946 ;
                                0963   947 ; OUTPUT PATAMETERS:
                                0963   948 ;       possible errors logged in the ERLB buffer
                                0963   949 ;
                                0963   950 ;--
                                0963   951
                                0963   952 AST1:
                        001C    0963   953         .WORD   ^M<R2,R3,R4>
             53   012B'CF    D0  0965   954         MOVL    W^ASTPAR1,R3            ; save the parameter
                  012B'CF    D7  096A   955         DECL    W^ASTPAR1               ; set up for the next AST
                04 AC   53    D1  096E   956         CMPL    R3,4(AP)                ; is this a good AST param.?
```

```
                33   13  0972   957          BEQL    C20                     ; br if yes
                11   11  0974   958          BRB     C10                     ; br if no
                         0976   959 AST2:
                     001C 0976   960          .WORD   ^M<R2,R3,R4>
53      012F'CF  D0   0978   961          MOVL    W^ASTPAR2,R3            ; save the parameter
        012F'CF  D6   097D   962          INCL    W^ASTPAR2               ; set up for the next AST
  04 AC      53  D1   0981   963          CMPL    R3,4(AP)                ; is this AST param OK?
                20   13  0985   964          BEQL    C20                     ; br if yes
                         0987   965 C10:
1390'CF    00    FB   0987   966          CALLS   #0,W^STORE_STEP         ; save the step information
54      0C4D'CF  D0   098C   967          MOVL    W^ELBP,R4               ; get error log pointer
        84   03   90   0991   968          MOVB    #3,(R4)+                ; save the longword count
  84      04 AC  D0   0994   969          MOVL    4(AP),(R4)+             ; save received data
        84   53   D0   0998   970          MOVL    R3,(R4)+                ; save expected data
84      0180'CF  DE   099B   971          MOVAL   W^ASTPRM,(R4)+          ; save string variable
                64   D4   09A0   972          CLRL    (R4)                    ; set a new terminator
0C4D'CF    54    D0   09A2   973          MOVL    R4,W^ELBP               ; reset the buffer pointer
                         09A7   974 C20:
  05 0133'CF    E8   09A7   975          BLBS    W^ASTOK,30$             ; should AST have occured?
134B'CF    00    FB   09AC   976          CALLS   #0,W^AST_FAIL           ; report a failure
                         09B1   977 30$:
                04   09B1   978          RET
```

```
                                09B2   980 .SBTTL SUPER_MODE
                                09B2   981 ;++
                                09B2   982 ; FUNCTIONAL DESCRIPTION:
                                09B2   983 ;       Routine to handle the CHMS instructions.
                                09B2   984 ;
                                09B2   985 ; CALLING SEQUENCE:
                                09B2   986 ;       CHMS    #N
                                09B2   987 ;
                                09B2   988 ; INPUT PARAMETERS:
                                09B2   989 ;         SP=>  CHMS parameter
                                09B2   990 ;               PC
                                09B2   991 ;               PSL
                                09B2   992 ;
                                09B2   993 ;         The CHMS parameter can be one of the following:
                                09B2   994 ;
                                09B2   995 ;               1 = execute super mode AST tests
                                09B2   996 ;               2 = execute a $SETAST_G to disable super mode AST's
                                09B2   997 ;               3 = execute a $DCLCMH_S to reset the CHMS handler to DCL
                                09B2   998 ;               4 = execute a $SETPRA_S & $SETPRA_G service
                                09B2   999 ;               5 = execute the $CANTIM_S & $CANTIM_G tests
                                09B2  1000 ;
                                09B2  1001 ; OUTPUT PARAMETERS:
                                09B2  1002 ;       NONE
                                09B2  1003 ;--
                                09B2  1004
                                09B2  1005 SUPER_MODE:
      50    8E    D0            09B2  1006         MOVL    (SP)+,R0                ; get CHM parameter off the stack
05    01    50    8F            09B5  1007         CASEB   R0,#1,#5                ; do the right thing
                                09B9  1008 10$:
                    000A'       09B9  1009         .WORD   20$-10$
                    0000        09BB  1010         .WORD   10$-10$
                    008D'       09BD  1011         .WORD   40$-10$
                    00B3'       09BF  1012         .WORD   50$-10$
                    00FF'       09C1  1013         .WORD   A60-10$
                                09C3  1014 20$:
0127'CF   003F'CF     DE        09C3  1015         MOVAL   W^SETAST,W^SERV_NAME    ; set service name pointer
          007D'CF     D4        09CA  1016         CLRL    W^SET+SETAST$_ENBFLG    ; set parameter list to disable
                                09CE  1017         $SETAST_G W^SET                 ; disable super mode AST's
                                09D7  1018         FAIL_CHECKNP SS$_WASSET         ; check for success
      00000000'8F     DD        09D7                       PUSHL   #SS$_WASSET
     122D'CF    01    FB        09DD                       CALLS   #1,W^REG_CHECKNP
0127'CF   0038'CF     DE        09E2  1019         MOVAL   W^DCLAST,W^SERV_NAME    ; set service name pointer
                                09E9  1020         $DCLAST_S W^AST1,#-2,#PSL$C_SUPER ; declare super AST #1
                                09FC  1021         FAIL_CHECKNP SS$_NORMAL         ; and check for success
      00000000'8F     DD        09FC                       PUSHL   #SS$_NORMAL
     122D'CF    01    FB        0A02                       CALLS   #1,W^REG_CHECKNP
          0071'CF     D6        0A07  1022         INCL    W^DCL+DCLAST$_ASTPRM    ; set new parameter
          0075'CF     D7        0A0B  1023         DECL    W^DCL+DCLAST$_ACMODE    ; set to super mode
                                0A0F  1024         $DCLAST_G W^DCL                 ; declare super AST #2
                                0A18  1025         FAIL_CHECKNP SS$_NORMAL         ; and check for failure
      00000000'8F     DD        0A18                       PUSHL   #SS$_NORMAL
     122D'CF    01    FB        0A1E                       CALLS   #1,W^REG_CHECKNP
0127'CF   003F'CF     DE        0A23  1026         MOVAL   W^SETAST,W^SERV_NAME    ; set service name pointer
     0133'CF    01    90        0A2A  1027         MOVB    #1,W^ASTOK              ; AST's are now legal
                                0A2F  1028         $SETAST_S       #1              ; enable super mode AST's
                                0A38  1029         FAIL_CHECKNP SS$_WASCLR         ; check for success
      00000000'8F     DD        0A38                       PUSHL   #SS$_WASCLR
```

```
                122D'CF    01    FB  0A3E                        CALLS   #1,W^REG_CHECKNP
                     00D3   31    0A43  1030              BRW     A70                     ; carry on
                            0A46  1031  40$:
0127'CF    0031'CF    DE    0A46  1032                    MOVAL   W^DCLCMH,W^SERV_NAME    ; set service name pointer
                            0A4D  1033                    $DCLCMH_S @PRVHND1,,#0          ; reset the CHMS handler for DCL
                            0A5E  1034                    FAIL_CHECK SS$_NORMAL           ; check for success
                 00000000'8F    DD  0A5E                        PUSHL   #SS$_NORMAL
                0B24'CF    01    FB  0A64                        CALLS   #1,W^REG_CHECK
                     00AD   31    0A69  1035              BRW     A70                     ; carry on
                            0A6C  1036  50$:
                            0A6C  1037                    $SETPRA_S @W^HANDLER_ADR,#PSL$C_KERNEL ; check _S and mode maximizing
                            0A79  1038                    FAIL_CHECK SS$_NORMAL           ; check success
                 00000000'8F    DD  0A79                        PUSHL   #SS$_NORMAL
                0B24'CF    01    FB  0A7F                        CALLS   #1,W^REG_CHECK
                           02    DD  0A84  1039          PUSHL   #PSL$C_SUPER            ; set expected mode
                1313'CF    01    FB  0A86  1040          CALLS   #1,W^POWER_CHECK        ; check effect
                     0138'CF    D6  0A8B  1041          INCL    W^HANDLER_ADR           ; make address unique
0099'CF    0138'CF    D0    0A8F  1042                    MOVL    W^HANDLER_ADR,W^SET2+SETPRA$_ASTADR ; set new address
                            0A96  1043                    $SETPRA_G W^SET2                ; check _G and mode maximizing
                            0A9F  1044                    FAIL_CHECK SS$_NORMAL           ; check for success
                 00000000'8F    DD  0A9F                        PUSHL   #SS$_NORMAL
                0B24'CF    01    FB  0AA5                        CALLS   #1,W^REG_CHECK
                           02    DD  0AAA  1045          PUSHL   #PSL$C_SUPER            ; set expected mode
                1313'CF    01    FB  0AAC  1046          CALLS   #1,W^POWER_CHECK        ; check effect
                     0138'CF    D6  0AB1  1047          INCL    W^HANDLER_ADR           ; make address unique
                          0061   31    0AB5  1048          BRW     A70                     ; carry on
                           00000022    0AB8  1049          STEP=34
                                       0AB8  105C  A60:
                                       0AB8  1051          $SETIMR_G (R2)                  ; make a super TQE
                                       0ABF  1052          $CANTIM_S REQIDT=#1,-
                                       0ABF  1053                    ACMODE=#PSL$C_SUPER   ; can it
                                       0ACA  1054          FAIL_CHECK SS$_NORMAL           ; check for success
                 00000000'8F    DD  0ACA                        PUSHL   #SS$_NORMAL
                0B24'CF    01    FB  0AD0                        CALLS   #1,W^REG_CHECK
                                       0AD5  1055  ;+
                                       0AD5  1056  ;
                                       0AD5  1057  ; cancel 2 super TQE's
                                       0AD5  1058  ;
                                       0AD5  1059  ;-
                                       0AD5  1060          NEXT_TEST
                                       0AD5
                                       0AD5        STP35:
                0004'CF    23    DO  0AD5                        MOVL    #35,W^CURRENT_TC
                           00    DD  0ADA                        PUSHL   #0
                0B1A'CF    01    FB  0ADC                        CALLS   #1,W^REG_SAVE
                                       0AE1  1061          $SETIMR_G (R2)                  ; make a super TQE
                                       0AE8  1062          $SETIMR_G (R2)                  ; and another
                           02    DO  0AEF  1063          MOVL    #PSL$C_SUPER,-
                     00A9'CF         0AF1  1064                  W^CAN+CANTIM$_ACMODE      ; set the access mode
                                       0AF4  1065          $CANTIM_G W^CAN                 ; can both of them
                                       0AFD  1066          FAIL_CHECK SS$_NORMAL           ; check for success
                 00000000'8F    DD  0AFD                        PUSHL   #SS$_NORMAL
                0B24'CF    01    FB  0B03                        CALLS   #1,W^REG_CHECK
                  10 A2    03    DO  0B08  1067          MOVL    #3,SETIMR$_REQIDT(R2)   ; set ID to 3
                                       0B0C  1068          $SETIMR_G (R2)                  ; set one for later
                  10 A2    01    DO  0B13  1069          MOVL    #1 SETIMR$_REQIDT(R2)   ; reset ID to 1
                           00    11  0B17  1070          BRB     A70                     ; carry on
```

```
                        0B19  1071 A70:
                 02     0B19  1072         REI                                     ; go back to user mode
                        0B1A  1073         .SBTTL REG_SAVE
                        0B1A  1074 ;++
                        0B1A  1075 ; FUNCTIONAL DESCRIPTION:
                        0B1A  1076 ;       Subroutine to save R2-R11 in the register save location.
                        0B1A  1077 ;
                        0B1A  1078 ; CALLING SEQUENCE:
                        0B1A  1079 ;       PUSHL   #0                  ; save a dummy parameter
                        0B1A  1080 ;       CALLS   #1,W^REG_SAVE       ; save R2-R11
                        0B1A  1081 ;
                        0B1A  1082 ; INPUT PARAMETERS:
                        0B1A  1083 ;       NONE
                        0B1A  1084 ;
                        0B1A  1085 ; OUTPUT PARAMETERS:
                        0B1A  1086 ;       NONE
                        0B1A  1087 ;
                        0B1A  1088 ;--
                        0B1A  1089
                        0B1A  1090 REG_SAVE:
                   0FFC 0B1A  1091         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
      000C'CF   14 AD   28  28 0B1C  1092         MOVC3   #4+10,^X14(FP),W^REG_SAVE_AREA  ; save the registers in the program
                 04   0B23  1093         RET
                      0B24  1094         .SBTTL  REG_CHECK
                      0B24  1095 ;++
                      0B24  1096 ; FUNCTIONAL DESCRIPTION:
                      0B24  1097 ;       Subroutine to test R0 & R2-R11 for proper content after a service
                      0B24  1098 ;       execution. A snapshot is taken by the REG_SAVE routine at the
                      0B24  1099 ;       beginning of each step and this routine is executed after the
                      0B24  1100 ;       services have been executed.
                      0B24  1101 ;
                      0B24  1102 ; CALLING SEQUENCE:
                      0B24  1103 ;       PUSHL   #SS$_XXXXXX     ; push expected R0 contents
                      0B24  1104 ;       CALLS   #1,W^REG_CHECK  ; execute this routine
                      0B24  1105 ;
                      0B24  1106 ; INPUT PARAMETERS:
                      0B24  1107 ;       expected R0 contents on the stack
                      0B24  1108 ;
                      0B24  1109 ; OUTPUT PARAMETERS:
                      0B24  1110 ;       possible error messages printed using $PUTMSG
                      0B24  1111 ;
                      0B24  1112 ;--
                      0B24  1113
                      0B24  1114 REG_CHECK:
                   0FFC 0B24  1115         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
           50   04 AC   D1  0B26  1116         CMPL    4(AP),R0                        ; is this the right fail code?
                    1C   13  0B2A  1117         BEQL    10$                             ; br if yes
                    50   DD  0B2C  1118         PUSHL   R0                              ; push received data
                 04 AC   DD  0B2E  1119         PUSHL   4(AP)                           ; push expected data
              0195'CF   DF  0B31  1120         PUSHAL  W^EXP                           ; push the string variable
        0B88'CF    03   FB  0B35  1121         CALLS   #3,W^PRINT_FAIL                 ; print the error message
   0050'CF    002A'CF   DE  0B3A  1122         MOVAL   W^TEST_MOD_FAIL,W^TMD_ADDR      ; set failure message address
0048'CF    03   00   02   F0  0B41  1123         INSV    #ERROR,#0,#3,W^MOD_MSG_CODE     ; set severity code
                         0B48  1124 10$:
  000C'CF    14 AD   28   29  0B48  1125         CMPC3   #4+10,^X14(FP),W^REG_SAVE_AREA  ; check all but R0
                    36   13  0B4F  1126         BEQL    20$                             ; br if O.K.
 56   53   0000000C'8F   C3  0B51  1127         SUBL3   #REG_SAVE_AREA,R3,R6            ; calculate the register number
```

```
                  56    04    C6  0B59  1128          DIVL2   #4,R6
          00C3'CF   56    02    8'  0B5C  1129          ADDB3   #^X2,R6,W^REGNUM                ; put it in the string
                    51    03    CA  0B62  1130          BICL2   #3,R1                           ; backup to register boundrys
                    53    03    CA  0B65  1131          BICL2   #3,R3
                  00C3'CF   DD    0B68  1132          PUSHL   W^REGNUM                                ; push register number
                        61    DD  0B6C  1133          PUSHL   (R1)                            ; push received data
                        63    DD  0B6E  1134          PUSHL   (R3)                            ; push expected data
                  00B1'CF   DF    0B70  1135          PUSHAL  W^REG                           ; set string pntr param.
              0B88'CF   04    FB  0B74  1136          CALLS   #4,W^PRINT_FAIL                 ; print the error message
          0050'CF   002A'CF   DE  0B79  1137          MOVAL   W^TEST_MOD_FAIL,W^TMD_ADDR      ; set failure message address
      0048'CF   03    00    02  F0  0B80  1138          INSV    #ERROR,#0,#3,W^MOD_MSG_CODE     ; set severity code
                                0B87  1139 20$:
                              04  0B87  1140          RET
                                0B88  1141          .SBTTL  PRINT_FAIL
                                0B88  1142 ;++
                                0B88  1143 ; FUNCTIONAL DESCRIPTION:
                                0B88  1144 ;       Subroutine to report failures using $PUTMSG
                                0B88  1145 ;
                                0B88  1146 ; CALLING SEQUENCE:
                                0B88  1147 ; Mode  #1       PUSHL EXPECTED  Mode    #2      PUSHL REG_NUMBER
                                0B88  1148 ;                PUSHL RECEIVED                   PUSHL EXPECTED
                                0B88  1149 ;                PUSHAL STRING_VAR                PUSHL RECEIVED
                                0B88  1150 ;                CALLS #3,W^PRINT_FAIL            PUSHAL STRING_VAR
                                0B88  1151 ;                                                 CALLS #4,W^PRINT_FAIL
                                0B88  1152 ; Mode  #3       PUSHAL STRING_VAR
                                0B88  1153 ;                CALLS #1,W^PRINT_FAIL
                                0B88  1154 ;
                                0B88  1155 ; INPUT PARAMETERS:
                                0B88  1156 ;       listed above
                                0B88  1157 ;
                                0B88  1158 ; OUTPUT PARAMETERS:
                                0B88  1159 ;       an error message is printed using $PUTMSG
                                0B88  1160 ;
                                0B88  1161 ;--
                                0B88  1162
                                0B88  1163 PRINT_FAIL:
                          003C  0B88  1164          .WORD   ^M<R2,R3,R4,R5>
                                0B8A  1165          $FAO_S  W^CS1,W^MESSAGEL,W^MSGL,#TEST_MOD_NAME,W^SERV_NAME,W^CURRENT_TC
                                0BAB  1166          $PUTMSG_S W^MSGVEC                      ; print the message
                      04    6C    91  0BBC  1167          CMPB    (AP),#4                 ; is this a register message?
                            26    13  0BBF  1168          BEQL    10$                     ; br if yes
                      01    6C    91  0BC1  1169          CMPB    (AP),#1                 ; is this just a message?
                            48    13  0BC4  1170          BEQL    20$                     ; br if yes
                                0BC6  1171          $FAO_S  W^CS2,W^MESSAGEL,W^MSGL,4(AP),8(AP),4(AP),12(AP)
                            40    11  0BE5  1172          BRB     30$                     ; goto output message
                                0BE7  1173 10$:
                                0BE7  1174          $FAO_S  W^CS3,W^MESSAGEL,W^MSGL,4(AP),16(AP),8(AP),4(AP),16(AP),12(AP)
                            19    11  0C0C  1175          BRB     30$                     ; goto output message
                                0C0E  1176 20$:
                  015C'CF     04 AC   D0  0C0E  1177          MOVL    4(AP),W^MSGVEC1+12      ; save string address
                                0C14  1178          $PUTMSG_S W^MSGVEC1                     ; print the message
                            11    11  0C25  1179          BRB     40$                     ; skip the other message
                                0C27  1180 30$:
                                0C27  1181          $PUTMSG_S W^MSGVEC                      ; print the message
                                0C38  1182 40$:
                12E6'CF     00    FB  0C38  1183          CALLS   #0,W^MODE_ID            ; identify the mode
          0050'CF   002A'CF   DE  0C3D  1184          MOVAL   W^TEST_MOD_FAIL,W^TMD_ADDR      ; set failure message address
```

```
0048'CF   03   00   02   F0  0C44  1185        INSV    #ERROR,#0,#3,W^MOD_MSG_CODE      ; set severity code
                         04  0C4B  1186        RET
```

```
                        0C4C  1188          .SBTTL REG_CHECKNP
                        0C4C  1189 ;++
                        0C4C  1190 ; FUNCTIONAL DESCRIPTION:
                        0C4C  1191 ;       Subroutine to test R0 & R2-R11 for proper content after a service
                        0C4C  1192 ;       execution without printing it. A snapshot is taken by the REG_SAVE routine a
                        0C4C  1193 ;       beginning of each step and this routine is executed after the
                        0C4C  1194 ;       services have been executed. This routine collects the error
                        0C4C  1195 ;       information in buffer ERLB instead of printing it.
                        0C4C  1196 ;
                        0C4C  1197 ; CALLING SEQUENCE:
                        0C4C  1198 ;       PUSHL   #SS$_XXXXXX     ; push expected R0 contents
                        0C4C  1199 ;       CALLS   #1,W^REG_CHECK  ; execute this routine
                        0C4C  1200 ;
                        0C4C  1201 ; INPUT PARAMETERS:
                        0C4C  1202 ;       expected R0 contents on the stack
                        0C4C  1203 ;
                        0C4C  1204 ; OUTPUT PARAMETERS:
                        0C4C  1205 ;       possible error messages logged in buffer ERLB which are printed
                        0C4C  1206 ;       using routine ERLBUF_DUMP.
                        0C4C  1207 ;
                        0C4C  1208 ;       Error packets are in the following form:
                        0C4C  1209 ;
                        0C4C  1210 ;               !----------------!
                        0C4C  1211 ;               !Service name pntr!
                        0C4C  1212 ;               !----------------!
                        0C4C  1213 ;               !     Step #     !
                        0C4C  1214 ;               !----------------!
                        0C4C  1215 ;               !Mode name pointer!
                        0C4C  1216 ;               !----------------!
                        0C4C  1217 ;               !              ! ! long word count
                        0C4C  1218 ;               !----------------!
                        0C4C  1219 ;               !\/\/\/\/\/\/\/\! 3-4 parameter long words
                        0C4C  1220 ;
                        0C4C  1221 ;--
                        0C4C  1222
                        0C4C  1223 FLAG:
                     00 0C4C  1224          .BYTE 0                 ; error flags are BIT0 = 0 means no errors in the bu
                        0C4D  1225                                  ;                 BIT0 = 1 means errors in the buffe
                        0C4D  1226 ELBP:
             00000C51' 0C4D  1227          .ADDRESS ERLB           ; error log buffer pointer
                        0C51  1228 ERLB:
             0000122D  0C51  1229          .BLKB   1500            ; error log buffer
                        122D  1230 ;
                        122D  1231 REG_CHECKNP:
                 0FFC   122D  1232          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
               50   04 AC  D1  122F  1233          CMPL    4(AP),R0        ; is this the right fail code
                     2E  13  1233  1234          BEQL    10$             ; br if yes
          1390'CF    00  FB  1235  1235          CALLS #0,W^STORE_STEP   ; store step information
          52   FA0F  CF  D0  123A  1236          MOVL    ELBP,R2         ; get the current error log pointer
               82    03  90  123F  1237          MOVB    #3,(R2)+        ; save the long word count
               82    50  D0  1242  1238          MOVL    R0,(R2)+        ; save received status
                 82  04 AC  D0  1245  1239          MOVL    4(AP),(R2)+     ; save expected status
          82   0195'CF    DE  1249  1240          MOVAL   W^EXP,(R2)+     ; save the string variable
               62    D4  124E  1241          CLRL    (R2)            ; set the terminator
          F9F8  CF    52  D0  1250  1242          MOVL    R2,ELBP         ; reset the buffer pointer
     0050'CF    002A'CF  DE  1255  1243          MOVAL   W^TEST_MOD_FAIL,W^TMD_ADDR ; set failure message address
0048'CF    03   00    02  F0  125C  1244          INSV    #ERROR,#0,#3,W^MOD_MSG_CODE ; set severity code
```

```
                                1263  1245 10$:
000C'CF   14 AD   28    29    1263  1246        CMPC3   #4+10,^X14(FP),W^REG_SAVE_AREA ; check all but R0 and R1
                  3C    13    126A  1247        BEQL    20$                     ; br if OK
          1390'CF 00    FB    126C  1248        CALLS #0,W^STORE_STEP           ; store step information
          52   F9D8 CF  D0    1271  1249        MOVL    ELBP,R2                 ; get current error log buf pointer
               82   04  90    1276  1250        MOVB    S^#4,(R2)+              ; set longword count
           0000000C'8F  C3    1279  1251        SUBL3   #REG_SAVE_AREA,-
               56   53        127F  1252        R3,R6                           ; calc reg number
               56   04  C6    1281  1253        DIVL2   S^#4,R6                 ; make it a longword count
          82   56   02  C1    1284  1254        ADDL3   S^#2,R6,(R2)+           ; correct for R0-R1 and save
               82   61  D0    1288  1255        MOVL    (R1),(R2)+              ; save received results
               82   63  D0    128B  1256        MOVL    (R3),(R2)+              ; save expected results
          82   00B1'CF  DE    128E  1257        MOVAL   W^REG,(R2)+             ; save string variable
                    62  D4    1293  1258        CLRL    (R2)                    ; set the terminator
          F9B3 CF   52  D0    1295  1259        MOVL    R2,ELBP                 ; reset the buffer pointer
     0050'CF   002A'CF  DE    129A  1260        MOVAL   W^TEST_MOD_FAIL,W^TMD_ADDR      ; set failure message address
0048'CF   03   00   02  F0    12A1  1261        INSV    #ERROR,#0,#3,W^MOD_MSG_CODE     ; set severity code
                              12A8  1262 20$:
                          04  12A8  1263        RET                             ; bail out
```

```
                                12A9  1265          .SBTTL  ERLBUF_DUMP
                                12A9  1266 ;++
                                12A9  1267 ; FUNCTIONAL DESCRIPTION:
                                12A9  1268 ;       Routine to check for errors in the error log buffer and
                                12A9  1269 ;       report any that are there.
                                12A9  1270 ;
                                12A9  1271 ; CALLING SEQUENCE:
                                12A9  1272 ;       CALLS #0,W^ERLBUF_DUMP
                                12A9  1273 ;
                                12A9  1274 ; INPUT PARAMETERS:
                                12A9  1275 ;       FLAG bit 0 = 0 for no errors logged
                                12A9  1276 ;       FLAG bit 0 = 1 for errors logged
                                12A9  1277 ;       if errors logged then buffer ERLB must contain legal format errors
                                12A9  1278 ;
                                12A9  1279 ; OUTPUT PARAMETERS:
                                12A9  1280 ;       NONE
                                12A9  1281 ;
                                12A9  1282 ;--
                                12A9  1283
                                12A9  1284 ERLBUF_DUMP:
                         001C   12A9  1285          .WORD   ^M<R2,R3,R4>
  2A F99D CF    E9              12AB  1286          BLBC    FLAG,30$        ; br if no errors to report
52  F99D CF    DE              12B0  1287          MOVAL   ERLB,R2         ; set up buffer pointer
                                12B5  1288 10$:
            62  D5              12B5  1289          TSTL    (R2)            ; any more errors?
            21  13              12B7  1290          BEQL    30$             ; br if not
0127'CF     82  D0              12B9  1291          MOVL    (R2)+,W^SERV_NAME ; reset service name
0004'CF     82  D0              12BE  1292          MOVL    (R2)+,W^CURRENT_TC ; reset step #
00AD'CF     82  D0              12C3  1293          MOVL    (R2)+,W^MODE    ; reset the mode
      53    82  9A              12C8  1294          MOVZBL  (R2)+,R3        ; get the longword count
      54    53  D0              12CB  1295          MOVL    R3,R4           ; and save it
                                12CE  1296 20$:
            82  DD              12CE  1297          PUSHL   (R2)+           ; push a parameter
          FB 53  F5             12D0  1298          SOBGTR  R3,20$          ; and push them all
      F8B0 CF    54  FB         12D3  1299          CALLS   R4,W^PRINT_FAIL ; print the failure
            DB  11              12D8  1300          BRB     10$             ; do the next one
                                12DA  1301 30$:
F96C CF     F973 CF    DE       12DA  1302          MOVAL   W^ERLB,W^ELBP   ; reset the buffer pointer
            F96C CF    D4       12E1  1303          CLRL    W^ERLB          ; set fresh terminater
                       04       12E5  1304          RET                     ; bail out
```

```
                                                M 5
                        12E6  1307          .SBTTL  MODE_ID
                        12E6  1308 ;++
                        12E6  1309 ; FUNCTIONAL DESCRIPTION:
                        12E6  1310 ;       Subroutine to identify the mode that an exit handler is in.
                        12E6  1311 ;
                        12E6  1312 ; CALLING SEQUENCE:
                        12E6  1313 ;       CALLS   #0,W^MODE_ID
                        12E6  1314 ;
                        12E6  1315 ; INPUT PARAMETERS:
                        12E6  1316 ;       MODE contains an address pointing to an ascii string desc.
                        12E6  1317 ;       of the current CPU mode.
                        12E6  1318 ;
                        12E6  1319 ; OUTPUT PARAMETERS:
                        12E6  1320 ;       NONE
                        12E6  1321 ;
                        12E6  1322 ;--
                        12E6  1323
                        12E6  1324 MODE_ID:
                003C    12E6  1325          .WORD   ^M<R2,R3,R4,R5>
                        12E8  1326          $FAO_S  W^CS5,W^MESSAGEL,W^MSGL,MODE ; format the error message
                        1301  1327          $PUTMSG_S W^MSGVEC                   ; print the mode message
                  04    1312  1328          RET
                        1313  1329          .SBTTL POWER_CHECK
                        1313  1330 ;++
                        1313  1331 ; FUNCTIONAL DESCRIPTION:
                        1313  1332 ;       Subroutine to check the effects of the SETPRA system service and
                        1313  1333 ;       report any errors that might occur.
                        1313  1334 ;
                        1313  1335 ; CALLING SEQUENCE:
                        1313  1336 ;       PUSHL   #PSL$C_????                  ; push expected mode
                        1313  1337 ;       CALLS   #1,W^POWER_CHECK
                        1313  1338 ;
                        1313  1339 ; INPUT PARAMETERS:
                        1313  1340 ;       4(AP) = expected access mode
                        1313  1341 ;
                        1313  1342 ; OUTPUT PARAMETERS:
                        1313  1343 ;       NONE
                        1313  1344 ;
                        1313  1345 ;--
                        1313  1346
                        1313  1347 POWER_CHECK:
                              0060  1313  1348          .WORD   ^M<R5,R6>
 56   00000000'EF    DE    1315  1349          MOVAL   CTL$GL_POWERAST,R6           ; set checking address
    0138'CF    66    D1    131C  1350          CMPL    (R6),W^HANDLER_ADR           ; is the address there?
               0F    13    1321  1351          BEQL    10$                          ; br if yes
               66    DD    1323  1352          PUSHL   (R6)                         ; push received data
          0138'CF    DD    1325  1353          PUSHL   W^HANDLER_ADR                ; push expected data
          01A3'CF    DF    1329  1354          PUSHAL  W^ADR                        ; push string variable
     F856 CF    03    FB    132D  1355          CALLS   #3,W^PRINT_FAIL              ; print the error
                        1332  1356 10$:
    04 AC    04 A6    91    1332  1357          CMPB    4(R6),4(AP)                  ; is the mode correct?
               11    13    1337  1358          BEQL    20$                          ; br if yes
       7E    04 A6    9A    1339  1359          MOVZBL  4(R6),-(SP)                  ; push received data
       7E    04 AC    9A    133D  1360          MOVZBL  4(AP),-(SP)                  ; push expected data
          01B2'CF    DF    1341  1361          PUSHAL  W^MS                         ; push string variable
     F83E CF    03    FB    1345  1362          CALLS   #3,W^PRINT_FAIL              ; print the error
                        134A  1363 20$:
```

```
                                N 5
 04  134A  1364          RET                                ; bail out
```

```
                                                                .SBTTL  AST_FAIL
                                                        ;++
                                                        ; FUCTIONAL DESCRIPTION:
                                                        ;       Subroutine to identify an AST failure.
                                                        ;
                                                        ; CALLING SEQUENCE:
                                                        ;       CALLS #0,W^AST_FAIL
                                                        ;
                                                        ; INPUT PARAMETERS:
                                                        ;       MODE            contains an address pointing to an ascii string desc.
                                                        ;                       of the current CPU mode.
                                                        ;       CURRENT_TC      contains the current test case number.
                                                        ;       ASTOK           contains expected or unexpected flag.
                                                        ;
                                                        ; OUTPUT PARAMETERS:
                                                        ;       NONE
                                                        ;
                                                        ;--

                                                        AST_FAIL:
                                                                .WORD   ^M<R2,R3,R4,R5>
    1390'CF    00    FB                                         CALLS #0,W^STORE_STEP                   ; store step information
    52    F8F7 CF    D0                                         MOVL    W^ELBP,R2                       ; get error log buffer pointer
          82   01    90                                         MOVB    #1,(R2)+                        ; save the longword count
 07 0133'CF    01    E1                                         BBC     #1,W^ASTOK,10$                  ; br if unexpected exception
    82    0121'CF    DE                                         MOVAL   W^CS6,(R2)+                     ; save the string variable
                05    11                                         BRB     20$                             ; skip other message
                                                        10$:
    82    00ED'CF    DE                                         MOVAL   W^CS4,(R2)+                     ; save the string variable
                                                        20$:
                62    D4                                         CLRL    (R2)                            ; set a new terminator
F8DA CF         52    D0                                         MOVL    R2,W^ELBP                       ; reset buffer pointer
                04    13                                         RET
                                                                .SBTTL  ILEGAL_TIM
                                                        ;++
                                                        ; FUCTIONAL DESCRIPTION:
                                                        ;       AST routine to report an illegal TQE.
                                                        ;
                                                        ; CALLING SEQUENCE:
                                                        ;       Entered via a SETIMR AST
                                                        ;
                                                        ; INPUT PARAMETERS:
                                                        ;       MODE            contains an address pointing to an ascii string desc.
                                                        ;                       of the current CPU mode.
                                                        ;       CURRENT_TC      contains the current test case number.
                                                        ;       ASTOK           contains expected or unexpected flag.
                                                        ;
                                                        ; OUTPUT PARAMETERS:
                                                        ;       NONE
                                                        ;
                                                        ;--

                                                        ILEGAL_TIM:
                                                                .WORD   ^M<R2,R3,R4,R5>
1390'CF    00    FB                                         CALLS #0,W^STORE_STEP                   ; store step information
52    F8CE CF    D0                                         MOVL    W^ELBP,R2                       ; get erl buf pntr
      82   01    90                                         MOVB    #1,(R2)+                        ; save the longword count
```

```
                                                C 6
82   0144'CF    DE  1383  1423          MOVAL   W^CS7,(R2)+                     ; save the string variable
           62   D4  1388  1424          CLRL    (R2)                            ; set a new terminator
F8BE CF    52   D0  138A  1425          MOVL    R2,W^EL                         ; reset the buffer pntr
                04  138F  1426          RET
                    1390  1427          .SBTTL STORE_STEP
                    1390  1428 ;++
                    1390  1429 ; FUNCTIONAL DESCRIPTION:
                    1390  1430 ;       Routine to store step information in the error log buffer.
                    1390  1431 ;
                    1390  1432 ; CALLING SEQUENCE:
                    1390  1433 ;       CALLS #0,W^STORE_STEP
                    1390  1434 ;
                    1390  1435 ; INPUT PARAMETERS:
                    1390  1436 ;       ELBP = current errlog buffer pointer
                    1390  1437 ;
                    1390  1438 ; OUTPUT PARAMETERS:
                    1390  1439 ;       FLAG = error logged flag
                    1390  1440 ;
                    1390  1441 ;--
                    1390  1442
                    1390  1443 STORE_STEP:
               0004 1390  1444          .WORD   ^M<R2>
F8B5 CF    01   88  1392  1445          BISB2   #1,W^FLAG                       ; set the error logged flag
52   F8B2 CF    D0  1397  1446          MOVL    W^ELBP,R2                       ; get errlog buf pntr
82   0127'CF    D0  139C  1447          MOVL    W^SERV_NAME,(R2)+               ; save the service name
82   0004'CF    D0  13A1  1448          MOVL    W^CURRENT_TC,(R2)+              ; save the step number
82   00AD'CF    D0  13A6  1449          MOVL    W^MODE,(R2)+                    ; save the mode
F89D CF    52   D0  13AB  1450          MOVL    R2,W^ELBP                       ; reset the errlog buf pntr
                04  13B0  1451          RET                                     ; return
```

```
                                13B1  1453 MOD_MSG_PRINT:
                                13B1  1454 ;
                                13B1  1455 ;  ************************************************************************
                                13B1  1456 ;  *                                                                      *
                                13B1  1457 ;  *  PRINTS THE TEST MODULE BEGUN/SUCCESSFUL/FAILED MESSAGES              *
                                13B1  1458 ;  *      (USING THE PUTMSG MACRO).                                       *
                                13B1  1459 ;  *                                                                      *
                                13B1  1460 ;  ************************************************************************
                                13B1  1461 ;
                                13B1  1462          PUTMSG  <MOD_MSG_CODE,#2,TMN_ADDR,TMD_ADDR> ; PRINT MSG
                             05 13CC  1463          RSB                                         ; ... AND RETURN TO CALLER
                                13CD  1464 ;
                                13CD  1465 CHMRTN:
                                13CD  1466 ;  ************************************************************************
                                13CD  1467 ;  *                                                                      *
                                13CD  1468 ;  *     CHANGE MODE ROUTINE. THIS ROUTINE GETS CONTROL WHENEVER           *
                                13CD  1469 ;  *     A CMKRNL, CMEXEC, OR CMSUP SYSTEM SERVICE IS ISSUED              *
                                13CD  1470 ;  *     BY THE MODE MACRO ('TO' OPTION).  IT MERELY DOES                 *
                                13CD  1471 ;  *     A JUMP INDIRECT ON A FIELD SET UP BY MODE. IT HAS                *
                                13CD  1472 ;  *     THE EFFECT OF RETURNING TO THE END OF THE MODE                   *
                                13CD  1473 ;  *     MACRO EXPANSION.                                                 *
                                13CD  1474 ;  *                                                                      *
                                13CD  1475 ;  ************************************************************************
                                13CD  1476 ;
                           0000 13CD  1477          .WORD   0                                   ; ENTRY MASK
                0000005D'FF   17 13CF  1478          JMP     @CHM_CONT                           ; RETURN TO MODE MACRO IN NEW MODE
                                13D5  1479 ;
                                13D5  1480 ;  *     RET INSTR WILL BE ISSUED IN EXPANSION OF 'MODE FROM, ....' MACRO
                                13D5  1481 ;
                                13D5  1482          .END    SATSSS26
```

```
$$ARGS                  =  00000002
$$T1                    =  00000C4
$$T2                    =  00000004
A20                        000007DD  R      04
A60                        00000AB8  R      04
A70                        00000B19  R      04
ADR                        000001A3  R      02
ARGLST                     000001F6  R      02
AST1                       00000963  R      04
AST2                       00000976  R      04
AST3                       000005A6  R      04
AST4                       000005E4  R      04
ASTOK                      00000133  R      03
ASTPAR1                    0000012B  R      03
ASTPAR2                    0000012F  R      03
ASTPRM                     00000180  R      02
AST_FAIL                   0000134B  R      04
B10                        0000089F  R      04
BUF                        000000CF  R      03
C10                        00000987  R      04
C20                        000009A7  R      04
CAN                        000000A1  R      03
CANTIM                     0000004D  R      02
CANTIM$_ACMODE          =  00000008
CANTIM$_NARGS           =  00000002
CANTIM$_REQIDT          =  00000004
CHMRTN                     000013CD  R      04
CHM_CONT                   0000005D  R      03
CLEAN_UP                   000008CE  R      04
CS1                        0000005B  R      02
CS2                        0000008D  R      02
CS3                        000000BA  R      02
CS4                        000000ED  R      02
CS5                        0000010C  R      02
CS6                        00000121  R      02
CS7                        00000144  R      02
CS8                        00000169  R      02
CTL$GL_POWERAST            ********    X    04
CURRENT_TC                 00000004  R      03
DCL                        00000069  R      03
DCLAST                     00000038  R      02
DCLAST$_ACMODE          =  0000000C
DCLAST$_ASTADR          =  00000004
DCLAST$_ASTPRM          =  00000008
DCLAST$_NARGS           =  00000003
DCLCMH                     00000031  R      02
DELTA_1_SEC                00000222  R      02
DELTA_4_SEC                0000021A  R      02
DELTA_5_SEC                00000212  R      02
DUMMY                      00000000  R      02
ELBP                       00000C4D  R      04
EM                         000001D7  R      02
ERLB                       00000C51  R      04
ERLBUF_DUMP                000012A9  R      04
ERROR                   =  00000002
EXE$C_CMSTKSZ              ********    X    04
EXP                        00000195  R      02
FLAG                       00000C4C  R      04
HANDLER_ADR                00000138  R      03
HANDLER_PC                 00000907  R      04
ID                         00000148  R      03
ILEGAL_TIM                 00001374  R      04
INFO                    =  00000003
KM                         000001E8  R      02
LIB$SIGNAL                 ********    X    04
MESSAGEL                   0000011F  R      03
MODE                       000000AD  R      03
MODE_ID                    000012E6  R      04
MOD_MSG_CODE               00000048  R      03
MOD_MSG_PRINT              000013B1  R      04
MS                         000001B2  R      02
MSGL                       000000C7  R      03
MSGVEC                     0000 1FE  R      02
MSGVEC1                    00000150  R      03
NEXT_STEP                  00000638  R      04
ONE_SEC                    0000020E  R      02
PENDING_TC                 00000008  R      03
POWER_CHECK                00001313  R      04
PRV_USP                 =  00000003
PRINT_FAIL                 00000B88  R      04
PRIVMASK                   00000055  R      03
PRVHND1                    00000134  R      03
PRVPRT                     00000054  R      03
PSL$C_EXEC              =  00000001
PSL$C_KERNEL            =  00000000
PSL$C_SUPER             =  00000002
PSL$C_USER              =  00000003
PSL$S_CURMOD            =  00000002
PSL$V_CURMOD            =  00000018
PSL$V_PRVMOD            =  00000016
REG                        000000B1  R      03
REGNUM                     000000C3  R      03
REG_CHECK                  00000B24  R      04
REG_CHECKNP                0000122D  R      04
REG_SAVE                   00000B1A  R      04
REG_SAVE_AREA              0000000C  R      03
RETADR                     00000061  R      03
RETURN_PC                  00000903  R      04
SATSSS26                   00000000  RG     04
SERV_NAME                  00000127  R      03
SET                        00000079  R      03
SET1                       00000081  R      03
SET2                       00000095  R      03
SETAST                     0000003F  R      02
SETAST$_ENBFLG          =  00000004
SETAST$_NARGS           =  00000001
SETIMR                     00000046  R      02
SETIMR$_ASTADR          =  0000000C
SETIMR$_DAYTIM          =  00000008
SETIMR$_EFN             =  00000004
SETIMR$_NARGS           =  00000004
SETIMR$_REQIDT          =  00000010
SETPRA                     00000054  R      02
SETPRA$_ACMODE          =  00000008
```

```
SETPRA$_ASTADR           =  00000004
SETPRA$_NARGS            =  00000002
SETUP_SOPER                 0000090B  R      04
SEVERE                   =  00000004
SF$L_SAVE_FP             =  0000000C
SF$L_SAVE_PC             =  00000010
SHR$K_SHRDEF             =  00000001
SHR$_TEXT                =  00001130
SM                          000001CA  R      02
SS$_NORMAL                  ********    X    04
SS$_WASCLR                  ********    X    04
SS$_WASSET                  ********    X    04
STEP                     =  00000023
STORE_STEP                  00001390  R      04
STP0                        0000003D  R      04
STP1                        000000D2  R      04
STP10                       000004A4  R      04
STP11                       000004CC  R      04
STP12                       00000511  R      04
STP13                       00000559  R      04
STP14                       000005B3  R      04
STP15                       000005F1  R      04
STP16                       00000638  R      04
STP17                       00000663  R      04
STP18                       00000678  R      04
STP19                       00000696  R      04
STP2                        000000F2  R      04
STP20                       000006AB  R      04
STP21                       000006F3  R      04
STP22                       00000735  R      04
STP25                       0000074A  R      04
STP26                       00000795  R      04
STP27                       000007E2  R      04
STP28                       00000839  R      04
STP29                       00000877  R      04
STP3                        000001BF  R      04
STP35                       00000AD5  R      04
STP4                        0000028D  R      04
STP5                        000002F0  R      04
STP6                        00000305  R      04
STP7                        0000039B  R      04
STP8                        00000426  R      04
STP9                        0000047D  R      04
STS$V_INHIB_MSG          =  0000001C
SUCCESS                  =  00000001
SUPER_MODE                  000009B2  R      04
SYS$CANTIM                  ********  GX     04
SYS$CLREF                   ********  GX     04
SYS$CMEXEC                  ********  GX     04
SYS$CMKRNL                  ********  GX     04
SYS$DCLAST                  ********  GX     04
SYS$DCLCMH                  ********  GX     04
SYS$EXIT                    ********  GX     04
SYS$FAO                     ********   X     04
SYS$GETTIM                  ********  GX     04
SYS$HIBER                   ********  GX     04
SYS$PUTMSG                  ********  GX     04
SYS$SETAST                  ********  GX     04
SYS$SETIMR                  ********  GX     04
SYS$SETPRA                  ********  GX     04
SYS$SETPRN                  ********  GX     04
SYS$WAITFR                  ********  GX     04
SYS$WAKE                    ********  GX     04
TEST_MOD_BEGIN              00000019  R      02
TEST_MOD_FAIL               0000002A  R      02
TEST_MOD_NAME               00000000  R      02
TEST_MOD_NAME_D             00000009  R      02
TEST_MOD_SUCC               0000001F  R      02
TIME                        00000140  R      03
TMD_ADDR                    00000050  R      03
TMN_ADDR                    0000004C  R      03
TPID                        00000000  R      03
UETP$_SATSMS             =  007480D9
UETP$_TEXT               =  00741133
UM                          000001BE  R      02
WARNING                  =  00000000
WORK                        0000014C  R      03
X_SEC                       0000013C  R      03
```

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	0000022A (554.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
RWDATA	00000160 (352.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
SATSSS26	000013D5 (5077.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.07	00:00:00.50
Command processing	134	00:00:00.71	00:00:02.78
Pass 1	380	00:00:12.69	00:00:22.64
Symbol table sort	0	00:00:00.87	00:00:00.91
Pass 2	296	00:00:04.46	00:00:05.74
Symbol table output	23	00:00:00.16	00:00:00.17
Psect synopsis output	2	00:00:00.04	00:00:00.04
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	873	00:00:19.00	00:00:32.78

The working set limit was 1950 pages.
83335 bytes (163 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 561 non-local and 40 local symbols.
1482 source lines were read in Pass 1, producing 36 object records in Pass 2.
65 pages of virtual memory were used to define 61 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]UETP.MLB;1	12
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	46
TOTALS (all libraries)	58

825 GETS were required to define 58 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:SATSSS26/OBJ=OBJ$:SATSSS26 MSRC$:SATSSS26/UPDATE=(ENH$:SATSSS26)+EXECML$/LIB+SHRLIB$:UETP/LIB
```

SATSSS35
LIS

SATSSS26
LIS

SATSSS38
LIS

SATSS530
LIS

SATSSS37
LIS

SATSSS39
LIS

SATSSS36
LIS