K 1

```
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEE                    TTT        PPP         PPP
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUU         UUU  EEE                    TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP
```

```
_$


Va
---
000
000
000
7F
7F
7F
7F
7F
7F
7F
7F
7F
```

```
**FILE**ID**SYSTSTMAC                                   L 14
```

```
  SSSSSSSS  YY      YY    SSSSSSSS  TTTTTTTTTT    SSSSSSSS  TTTTTTTTTT  MM      MM    AAAAAA      CCCCCCCC
  SSSSSSSS  YY      YY    SSSSSSSS  TTTTTTTTTT    SSSSSSSS  TTTTTTTTTT  MM      MM    AAAAAA      CCCCCCCC
SS          YY      YY  SS              TT      SS              TT      MMMM  MMMM  AA      AA  CC
SS          YY      YY  SS              TT      SS              TT      MMMM  MMMM  AA      AA  CC
SS            YY  YY    SS              TT      SS              TT      MM  MM  MM  AA      AA  CC
SS            YY  YY    SS              TT      SS              TT      MM  MM  MM  AA      AA  CC
  SSSSSS        YY        SSSSSS        TT        SSSSSS        TT      MM      MM  AA      AA  CC
  SSSSSS        YY        SSSSSS        TT        SSSSSS        TT      MM      MM  AA      AA  CC
        SS      YY              SS      TT              SS      TT      MM      MM  AAAAAAAAAA  CC
        SS      YY              SS      TT              SS      TT      MM      MM  AAAAAAAAAA  CC
        SS      YY              SS      TT              SS      TT      MM      MM  AA      AA  CC              ....
        SS      YY              SS      TT              SS      TT      MM      MM  AA      AA  CC              ....
SSSSSSSS        YY      SSSSSSSS        TT      SSSSSSSS        TT      MM      MM  AA      AA    CCCCCCCC      ....
SSSSSSSS        YY      SSSSSSSS        TT      SSSSSSSS        TT      MM      MM  AA      AA    CCCCCCCC      ....

MM      MM    AAAAAA    RRRRRRRR
MM      MM    AAAAAA    RRRRRRRR
MMMM  MMMM  AA      AA  RR      RR
MMMM  MMMM  AA      AA  RR      RR
MM  MM  MM  AA      AA  RR      RR
MM  MM  MM  AA      AA  RR      RR
MM      MM  AA      AA  RRRRRRRR
MM      MM  AA      AA  RRRRRRRR
MM      MM  AAAAAAAAAA  RR  RR
MM      MM  AAAAAAAAAA  RR  RR
MM      MM  AA      AA  RR    RR
MM      MM  AA      AA  RR    RR
MM      MM  AA      AA  RR      RR
MM      MM  AA      AA  RR      RR
```

```
;
; Version:      'V04-000'
;
;****************************************************************************
;*                                                                          *
;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
;*  ALL RIGHTS RESERVED.                                                    *
;*                                                                          *
;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;*  TRANSFERRED.                                                            *
;*                                                                          *
;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;*  CORPORATION.                                                            *
;*                                                                          *
;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;*                                                                          *
;*                                                                          *
;****************************************************************************
; MODIFIED BY:
;
;       V03-003 LDJ0002         Larry D. Jones,         11-Apr-1983
;               Replaced the PRIV macro with one which uses $SETPRV.
;
;       V03-002 LDJ0001         Larry D. Jones,         16-Jun-1981
;               Modified all references to RTL routines to have G^ address
;               specification for the new linker.
;
;       V03-001 TLC0001         Thomas L. Cafarella,    22-Jan-1981
;               Modified macros COND, DECODE, and TESTSERV to remove trailing
;               comma failures caused by MACRO-32 changes.
;

        .MACRO  $QUOTA  NAME=LISTEND,   VALUE=0
        .BYTE   PQL$_'NAME
        .LONG   VALUE
        .ENDM
```

```
;+
; The $SHR_MESSAGES macro defines facility-specific message codes
;       which are based on the system-wide shared message codes.
;
;       $SHR_MESSAGES   name, code, <<msg,severity>, ... >
;
;       where:
;           "name" is the name of the facility (e.g., COPY)
;           "code" is the corresponding facility code (e.g., 103)
;           "msg" is the name of the shared message (e.g., BEGIN)
;           "severity" is the desired message severity (e.g., 1, 0, 2, 4)
;
;-
        .MACRO  $SHR_MESSAGES   NAME,CODE,MSGCODES
;
        .IF     NDF,SHR$K_SHRDEF  ; issue $SHRDEF if not done yet
                SHR$K_SHRDEF = 1 ; define symbol to indic $SHRDEF done
                $SHRDEF          ; define shared message codes
        .ENDC
        .IRP    MSGPAIR, <'MSGCODES>
                $SHRMSG_COD 'NAME', 'CODE', MSGPAIR
        .ENDR
        .ENDM
        .MACRO  $SHRMSG_COD NAME, CODE, MSG, SEVERITY
;
        .IF     IDN,SEVERITY,WARNING    ; if WARNING, set 0 sev
          'NAME'$_'MSG' = 0             ; set 0 sev (WARNING)
        .IFF
          .IF   IDN,SEVERITY,SUCCESS    ; if SUCCESS, set 1 sev
            'NAME'$_'MSG' = 1           ; set 1 sev (SUCCESS)
          .IFF
            .IF IDN,SEVERITY,ERROR      ; if ERROR, set 2 sev
              'NAME'$_'MSG' = 2         ; set 2 sev (ERROR)
            .IFF
              .IF IDN,SEVERITY,INFO     ; if INFO, set 3 sev
                'NAME'$_'MSG' = 3       ; set 3 sev (INFO)
              .IFF
                .IF IDN,SEVERITY,SEVERE ; if SEVERE, set 4 sev
                  'NAME'$_'MSG' = 4     ; set 4 sev (SEVERE)
                .IFF
                  'NAME'$_'MSG' = 'SEVERITY ; set specified sev
                .ENDC
              .ENDC
            .ENDC
          .ENDC
        .ENDC
         'NAME'$_'MSG' = 'NAME'$_'MSG'+SHR$_'MSG'+<'CODE'@16>
        .ENDM
```

```
;+
;       The $SIGNAL macro prints a message using the
;       LIB$SIGNAL facility.  It accepts a variable
;       number of longword arguments (max. 25) to satisfy
;       the requirements of LIB$SIGNAL.  If you want
;       to print SHR messages under another facility,
;       first use the $SHR_MESSAGES macro to define message
;       codes for the desired facility which are based on the
;       SHR message codes.
;-
;
;   S-FORM of $SIGNAL
;
        .MACRO  $SIGNAL_S ARG1, ARG2, ARG3, ARG4, ARG5, -
                        ARG6, ARG7, ARG8, ARG9, ARG10, -
                        ARG11, ARG12, ARG13, ARG14, ARG15, -
                        ARG16, ARG17, ARG18, ARG19, ARG20, -
                        ARG21, ARG22, ARG23, ARG24, ARG25
;
                $$T1  =  0          ; init number of $SIGNAL arguments
;
        .IRP    $$T2,<'ARG25,'ARG24,'ARG23,'ARG22,-
                   'ARG21,'ARG20,'ARG19,'ARG18,-
                   'ARG17,'ARG16,'ARG15,'ARG14,-
                   'ARG13,'ARG12,'ARG11,'ARG10,-
                   'ARG9, 'ARG8, 'ARG7, 'ARG6, -
                   'ARG5, 'ARG4, 'ARG3, 'ARG2, 'ARG1>
;
        .IF     NB,$$T2         ; gen PUSHL if arg specified
                $$T1 = $$T1+1   ; incr # of $SIGNAL args
                PUSHL   $$T2    ; push $SIGNAL arg onto stack
        .ENDC
        .ENDR
                CALLS   #$$T1,G^LIB$SIGNAL
        .ENDM
;
;   parameter list form of $SIGNAL
;
        .MACRO  $SIGNAL ARG1, ARG2, ARG3, ARG4, ARG5, -
                        ARG6, ARG7, ARG8, ARG9, ARG10, -
                        ARG11, ARG12, ARG13, ARG14, ARG15, -
                        ARG16, ARG17, ARG18, ARG19, ARG20, -
                        ARG21, ARG22, ARG23, ARG24, ARG25
;
                $$T1  =  0          ; init number of $SIGNAL arguments
;
        .IRP    $$T2,<'ARG1,'ARG2,'ARG3,'ARG4,-
                   'ARG5,'ARG6,'ARG7,'ARG8,-
                   'ARG9,'ARG10,'ARG11,'ARG12,-
                   'ARG13,'ARG14,'ARG15,'ARG16,-
                   'ARG17,'ARG18,'ARG19,'ARG20,-
                   'ARG21,'ARG22,'ARG23,'ARG24,'ARG25>
;
        .IF     NB,$$T2         ; add 1 to arg total if arg present
                $$T1 = $$T1+1   ; incr # of $SIGNAL args
        .ENDC
        .ENDR
                .LONG   $$T1    ; gen longword argument total
        .IRP    $$T2,<'ARG1,'ARG2,'ARG3,'ARG4,-
                   'ARG5,'ARG6,'ARG7,'ARG8,-
                   'ARG9,'ARG10,'ARG11,'ARG12,-
                   'ARG13,'ARG14,'ARG15,'ARG16,-
                   'ARG17,'ARG18,'ARG19,'ARG20,-
                   'ARG21,'ARG22,'ARG23,'ARG24,'ARG25>
;
        .IF     NB,$$T2         ; gen longword if arg present
                .LONG     $$T2  ; gen longword with this arg as value
        .ENDC
        .ENDR
        .ENDM
;
;  G-FORM of $SIGNAL
;
        .MACRO  $SIGNAL_G       LISTADR
;
                CALLG   LISTADR,G^LIB$SIGNAL
        .ENDM
```

```
;+
;       The $STOP macro prints a message using the
;       LIB$STOP facility.  It accepts a variable
;       number of longword arguments (max. 25) to satisfy
;       the requirements of LIB$STOP.  If you want
;       to print SHR messages under another facility,
;       first use the $SHR_MESSAGES macro to define message
;       codes for the desired facility which are based on the
;       SHR message codes.
;-
;
;  S-FORM of $STOP
;
        .MACRO  $STOP_S ARG1, ARG2, ARG3, ARG4, ARG5  -
                        ARG6, ARG7, ARG8, ARG9, ARG10, -
                        ARG11, ARG12, ARG13, ARG14, ARG15, -
                        ARG16, ARG17, ARG18, ARG19, ARG20, -
                        ARG21, ARG22, ARG23, ARG24, ARG25
;
                $$T1  =  0          , init number of $STOP arguments
;
        .IRP    $$T2,<'ARG25,'ARG24,'ARG23,'ARG22,-
                    'ARG21,'ARG20,'ARG19,'ARG18,-
                    'ARG17,'ARG16,'ARG15,'ARG14,-
                    'ARG13,'ARG12,'ARG11,'ARG10,-
                    'ARG9, 'ARG8, 'ARG7, 'ARG6, -
                    'ARG5, 'ARG4, 'ARG3, 'ARG2, 'ARG1>
;
        .IF     NB,$$T2         ; gen PUSHL if arg specified
                $$T1 = $$T1+1   ; incr # of $STOP args
                PUSHL   $$T2    ; push $STOP arg onto stack
        .ENDC
        .ENDR
                CALLS   #$$T1,G^LIB$STOP
        .ENDM
;
;  parameter list form of $STOP
;
        .MACRO  $STOP ARG1, ARG2, ARG3, ARG4, ARG5, -
                        ARG6, ARG7, ARG8, ARG9, ARG10, -
                        ARG11, ARG12, ARG13, ARG14, ARG15, -
                        ARG16, ARG17, ARG18, ARG19, ARG20, -
                        ARG21, ARG22, ARG23, ARG24, ARG25
;
                $$T1  =  0      ; init number of $STOP arguments
;
        .IRP    $$T2,<'ARG1,'ARG2,'ARG3,'ARG4,-
                    'ARG5,'ARG6,'ARG7,'ARG8,-
                    'ARG9,'ARG10,'ARG11,'ARG12,-
                    'ARG13,'ARG14,'ARG15,'ARG16,-
                    'ARG17,'ARG18,'ARG19,'ARG20,-
                    'ARG21,'ARG22,'ARG23,'ARG24,'ARG25>
;
        .IF     NB,$$T2         ; add 1 to arg total if arg present
                $$T1 = $$T1+1   ; incr # of $STOP args
        .ENDC
        .ENDR
                .LONG   $$T1    ; gen longword argument total
        .IRP    $$T2,<'ARG1,'ARG2,'ARG3,'ARG4,-
                   'ARG5,'ARG6,'ARG7,'ARG8,-
                   'ARG9,'ARG10,'ARG11,'ARG12,-
                   'ARG13,'ARG14,'ARG15,'ARG16,-
                   'ARG17,'ARG18,'ARG19,'ARG20,-
                   'ARG21,'ARG22,'ARG23,'ARG24,'ARG25>
;
        .IF     NB,$$T2         ; gen longword if arg present
                .LONG   $$T2    ; gen longword with this arg as value
        .ENDC
        .ENDR
        .ENDM
;
;  G-FORM of $STOP
;
        .MACRO  $STOP_G LISTADR
;
                CALLG   LISTADR,G^LIB$STOP
        .ENDM
```

```
; ****************************************************
; *
; *     MACRO COND GENERATES A CONDITION TABLE FOR TEST
; *     MODULES WHICH TEST SUCCESSFUL STATUS CODE RETURNS
; *     FROM SYSTEM SERVICES.
; *
; *     INPUTS:
; *             N -- A CONDITION TABLE NUMBER. WITHIN ANY
; *                  PARTICULAR TEST MODULE, CONDITION
; *                  TABLES ARE NUMBERED SEQUENTIALLLY,
; *                  BEGINNING WITH 1.
; *
; *             CTXT -- SYSTEM SERVICE ARGUMENT CONTEXT.
; *                  IF CONDITION IS A SYSTEM SERVICE ARGUMENT,
; *                  CTXT SPECIFIES ITS CONTEXT WITH KEYWORDS
; *                  AS FOLLOWS:
; *                     BYTE  --  BYTE CONTEXT
; *                     WORD  --  WORD CONTEXT
; *                     LONG  --  LONGWORD CONTEXT
; *                     QUAD  --  QUADWORD CONTEXT
; *                     DESC  --  QUADWORD STRING DESCRIPTOR.
; *                  IF CONDITION EXISTS BUT IS NOT A SYSTEM SERVICE
; *                  ARGUMENT, SPECIFY CTXT WITH THE KEYWORD
; *                  NOTARG.
; *                  IF CONDITION DOES NOT EXIST, SPECIFY
; *                  CTXT WITH THE KEYWORD NULL.  A NULL
; *                  CONDITION TABLE REQUIRES ONLY THE
; *                  N AND CTXT ARGUMENTS TO BE SPECIFIED.
; *
; *             CTITLE -- CONDITION TITLE. IF THE CONDITION
; *                  IS A SYSTEM SERVICE ARGUMENT, CTITLE
; *                  MUST BE THE ARGUMENT NAME; OTHERWISE,
; *                  IT IS ANY TEXT DESCRIBING THE CONDITION,
; *                  ENCLOSED IN ANGLE BRACKETS.
; *
; *             S1 THRU S5 -- UP TO 5 TEXT STRINGS. EACH
; *                  STRING IS BOUNDED BY ANGLE BRACKETS, AND
; *                  DESCRIBES THE PARTICULAR CONDITION FOR
; *                  EACH ELEMENT IN THE CONDITION TABLE.
; *
; *             DUMMY -- A dummy argument.  This argument must be specified
; *                  whenever the CTXT argument is anything but null.  DUMMY
; *                  is a null argument specified by a trailing comma.
; *
; *     OUTPUTS:
; *             (REPLACE N IN CONDN BELOW WITH CONDITION
; *               TABLE NUMBER.)
; *             CONDN_C          ASSIGNED VALUE FOR CONTEXT.
; *             CONDN_T:         CTITLE COUNTED STRING
; *             CONDN_H:         BYTE FIELD CONTAINING HIGHEST
; *                              ELEMENT NUMBER FOR THIS TABLE.
; *             CONDN_TAB:       GROUP OF COUNTED STRINGS
; *                              CONTAINING TEXT FOR S1 THRU S5.
; *             CONDN_E:         LABEL FOR FIRST BYTE
; *                              FOLLOWING CONDITION TABLE;
; *                              IT IS ALSO THE LABEL FOR
; *                             CONDITION VALUES FOR ARG-
; *                             UMENT TYPE CONDITIONS.
; *                             (THE VALUES MUST IMMEDIATELY
; *                             FOLLOW COND TABLE).
; *                 "ARGUMENT NAME": AN ALTERNATE NAME FOR CONDN_E.
; *
        .MACRO  COND    N,CTXT,CTITLE,S1,S2,S3,S4,S5,DUMMY
NOTARG          == 0            ; VALUE FOR NOTARG SPECIFICATION OF CONTEXT PARM
BYTE            == 1            ; VALUE FOR BYTE SPECIFICATION OF CONTEXT PARM
WORD            == 2            ; VALUE FOR WORD SPECIFICATION OF CONTEXT PARM
LONG            == 4            ; VALUE FOR LONG SPECIFICATION OF CONTEXT PARM
QUAD            == 8            ; VALUE FOR QUAD SPECIFICATION OF CONTEXT PARM
DESC            == 16           ; VALUE FOR DESC SPECIFICATION OF CONTEXT PARM
NULL            == 20           ; VALUE FOR NULL SPECIFICATION OF CONTEXT PARM
COND'N'_C       = CTXT
        .IF NE CTXT - NULL      ; IF CONDITION NOT NULL, EXPAND TABLE
        .NARG   $$$COND_A       ; GET NUMBER OF ARGS SPECIFIED
        $$$COND_A = $$$COND_A - 5 ; ADJUST TO GET HIGHEST ELEMENT NO. (MAX 5)
COND'N'_T:      STRING  C,<CTITLE>      ; TITLE FOR THIS CONDITION
COND'N'_H::     .BYTE   $$$COND_A ; PUT HIGH ELT. NO. INTO BYTE FIELD
COND'N'_TAB:
        STGRP   C,<S1>,         -
                  <S2>,         -
                  <S3>,         -
                  <S4>,         -
                  <S5>
COND'N'_E:              ; LABEL END OF CONDITION TABLE
        .IF NE COND'N'_C - NOTARG
CTITLE:                 ; LABEL THE TABLE OF VALUES WHICH MUST FOLLOW
        .ENDC
        .IFF            ; IF CONDITION IS NULL, MAKE SMALL TABLE
COND'N'_T:
COND'N'_TAB:
COND'N'_H::     .BYTE   0       ; MAKE HIGH ELEMENT NUMBER ZERO FOR NULL TABLE
        .ENDC
        .ENDM
```

```
;**************************************************************************
;
        .MACRO  DECODE ?C1C,?C1T,?C1R,?C2C,?C2T,?C2R,?C3C,?C3T,?C3R,?C4C,?C4T,?C4R,-
                CODE,VALID,CP2,CC2,CP3,CC3,CP4,CC4,CP5,CC5,CP6,CC6,DUMMY2
;
;**************************************************************************
;
; !!NOTE!! - When DECODE is invoked, its argument list must be followed by
;            a trailing comma, thus generating an extraneous null argument.
;
MEXIT=0
        .NARG   NARGS
        NARGS = NARGS - 1       ; Decrement for trailing null argument
        .IF GT NARGS-24
        .ERROR NARGS-12 ; EXCESS (>6) TEST PARAMETERS IGNORED.;
        .ENDC
;
        .IF LT NARGS-14
        .ERROR NARGS-12 ; INSUFFICIENT (<2) PARAMETERS SUPPLIED.;
        MEXIT=1
        .MEXIT
        .ENDC
;
        .IF GT CODE-4
        .ERROR CODE ; INVALID (NOT 1-4) ARGUMENT CODE SPECIFIED.;
        MEXIT=1
        .MEXIT
        .ENDC
;
        .IF EQ CODE
        .ERROR CODE ; INVALID (NOT 1-4) ARGUMENT CODE SPECIFIED.;
        MEXIT=1
        .MEXIT
        .ENDC
;
        .IF DIF,CODE,4
        MEXIT=NARGS-<2*<NARGS/2>>
        .IF NE MEXIT
        .ERROR NARGS-12 ; MISSING (ODD NUMBER OF) PARAMETER(S).;
        .ENDC
        .ENDC
;**************************************************************************
;
;       THE FOLLOWING THREE MACROS, LABEL, ADDRESS, AND LABGEN
;               ARE USED ONLY TO GENEARTE UNIQUE LABELS WITHIN THE
;               'DECODE' MACRO.
;
;**************************************************************************
        .MACRO  LABEL N
L'N=.
        .ENDM
        .MACRO  ADDRESS N
        .ADDRESS L'N
        .ENDM
        .MACRO  LABGEN FUNC,N,INC=1
        FUNC    \N
```

```
N=N+INC
        .ENDM
;
;*******************************************************************************
;
;
;       GENERATE CODE HERE FOR ARGUMENT TYPE '1'
;
        .IF     IDN,CODE,1
        BRW     C1C                     ; BRANCH AROUND ARG TABLES
        .IF DF VALID
C1T:    .ADDRESS VALID
        .IFF
C1T:    .ADDRESS C11
        .ENDC
        .IRP    X,<<CP2>,<CP3>,<CP4>,<CP5>,<CP6>>
        .IF NB <X>
        .ADDRESS X
        .ENDC
        .ENDR
        .IF DF VALID
        .ADDRESS VALID
        .IFF
        .ADDRESS C1T
        .ENDC
C1R:
        .IRP    X,<<CC2>,<CC3>,<CC4>,<CC5>,<CC6>>
        .IF     NB <X>
        .LONG   SS$_'X
        .ENDC
        .ENDR
C1C:    MOVAL   C1T,R10         ; PASS BACK PARM TABLE ADDRESS
        MOVZBL  #NARGS-14/2,(R9)+       ; SAVE NO. ARGS TO BE TESTED
        MOVAL   C1R,(R9)+               ; SAVE RETURN CODE TAB ADDR
        .ENDC
;*******************************************************************************
;
;
;       GENERATE CODE HERE FOR ARGUMENT TYPE '2'
;
        .IF     IDN,CODE,2
        BRW     C2C                     ; BRANCH AROUND ARG TABLES
N=1
        .IRP    X,<<VALID>,<CP2>,<CP3>,<CP4>,<CP5>,<CP6>>
        .IF NB  <X>
        LABGEN  LABEL,N
        .LONG   X
        .ENDC
        .ENDR
C2T:
N=1
        .IRP    X,<<VALID>,<CP2>,<CP3>,<CP4>,<CP5>,<CP6>>
        .IF NB  <X>
        LABGEN  ADDRESS,N
        .ENDC
        .ENDR
        .ADDRESS L1
C2R:
        .IRP    X,<<CC2>,<CC3>,<CC4>,<CC5>,<CC6>>
        .IF     NB <X>
        .LONG   SS$_'X
        .ENDC
        .ENDR
C2C:    MOVAL   C2T,R10                 ; SAVE ARG TABLE ADDR
        MOVZBL  #NARGS-14/2,(R9)+       ; SAVE NO. ARGS TO BE TESTED
        MOVAL   C2R,(R9)+               ; SAVE RETURN COD TAB ADDR
        .ENDC
;***************************************************************************
;
;       GENERATE CODE HERE FOR ARGUMENT TYPE '3'
;
;
        .IF     IDN,CODE,3
        BRW     C3C                     ; BRANCH AROUND ARG TABLES
;
N=1
        .IRP    X,<<VALID>,<CP2>,<CP3>,<CP4>,<CP5>,<CP6>>
        .IF NB  <X>
        LABGEN  LABEL,N
        STRING I,<X>
        .ENDC
        .ENDR
C3T:
N=1
        .IRP    X,<<VALID>,<CP2>,<CP3>,<CP4>,<CP5>,<CP6>>
        .IF NB <X>
        LABGEN  ADDRESS,N
        .ENDC
        .ENDR
        .ADDRESS L1
C3R:
        .IRP    X,<<CC2>,<CC3>,<CC4>,<CC5>,<CC6>>
        .IF     NB <X>
        .LONG   SS$_'X
        .ENDC
        .ENDR
C3C:    MOVAL   C3T,R10         ; SAVE ARG TABLE ADDR
        MOVZBL  #NARGS-14/2,(R9)+       ; SAVE NO. ARGS TO BE TESTED
        MOVAL   C3R,(R9)+               ; SAVE RETURN CODE TAB ADDR
        .ENDC
;***************************************************************************
;
;       GENERATE CODE HERE FOR  ARGUMENT TYPE '4'
;
;
        .IF     IDN,CODE,4
        BRW     C4C
N=1
        .IRP    X,<<VALID>,<CP2>,<CC2>,<CP3>,<CP4>,<CC4>,<CC5>,<CP6>>
        .IF NB  <X>
        LABGEN  LABEL,N
        .LONG   X
        .ENDC
        .ENDR
C4T:
N=1
```

```
        .IRP    X,<<VALID>,<CC2>,<CP4>,<CC5>>
        .IF NB  <X>
        LABGEN  ADDRESS,N,2
        .ENDC
        .ENDR
        .ADDRESS L1
C4R:
        .IRP    X,<<CC3>,<CP5>,<CC6>>
        .IF     NB <X>
        .LONG   SS$_'X
        .ENDC
        .ENDR
C4C:    MOVAL   C4T,R10          ; SAVE ARG TABLE ADDR
        MOVZBL  #NARGS-15/3,(R9)+         ; SAVE NO. ARGS TO BE TESTED
        MOVAL   C4R,(R9)                  ; SAVE RETURN CODE TAB ADDR
        .ENDC
;
;
;
;
        .ENDM                   ; END OF 'DECODE' DEFINITION
;
```

```
; ********* THIS MACRO CAN BE DELETED WHEN LARRY FINISHED WITH IT
        .MACRO  DISPLAY CTLSTR,ARGLST
        MOVAL   CTLSTR,R10              ; PROVIDE CONTRL STRING AND
        .NARG   NARGS
        .IF EQ NARGS-1                  ; CHEK FOR ARGLIST OMISSION
        CLRL    R11                     ; PASS '0' TO FAO
        .IFF
        MOVAL   ARGLST,R11              ; PASS ARG LIST ADDR
        .ENDC
        JSB     DISPLAY                 ; TO DISPLAY GIVEN MESSAGE.
        .ENDM
```

```
;*************************************************************************
;
        .MACRO  DISPSERV        ?END,?RMSG,?ARGL,?MSFLAG,?SET,?PREP
;
;*************************************************************************
        .ENABL  LSB
;
        BRW     END                     ; PREVENT FALL THRU FROM OTHER CODE
;
RMSG:   STRING I,<!/!AC!1ZB!1ZB: RETURN = !XW , EXPECTED = !XW >,-
                <: ARGUMENT LIST WAS !#(9XL)>
;
;
;
OUTL:   .LONG   0
$$CALL$$:.LONG  0                       ; CONSECUTIVE CALL NO. (KEPT BY SATS MODULES)
$$INIT$$:.LONG  0                       ; INIT PARM NO. VALUE FOR EACH ARG (SATS)
;
ARGL:
$$SNAD$$:.ADDRESS 0                     ;ADDR OF SERVICE NAME
$$ASEQ$$:.LONG  0                       ; ARGUMENT SEQUENCE NUMBER
$$PSEQ$$:.LONG  0                       ; TEST PARAMETER SEQ NUMBER
$$ACT$$:.LONG   0                       ; CONTENTS OF R0 AFTER SERVICE CALL
$$EXP$$:.LONG   0                       ; EXPECTED RETURN FROM SERVICE
$$ARG$$:.LONG   0                       ; NO. ARGUMENTS ENTERED FOR 'TESTSERV'
        .LONG   0                       ; FIRST ARG FOR CURRENT SERVICE CALL
        .LONG   0                       ; SECOND ARG  "   "   "
        .LONG   0                       ; ETC.
        .LONG   0
        .LONG   0
;
MSFLAG:.BYTE    0                       ; MISCELLANEOUS FLAGS
;
OUTD:
        .LONG   OUTE-OUTB
        .ADDRESS OUTB
OUTB:   .BLKL   33
OUTE:
;
$$ERR$$:
        PUSHR   #^M<R9,R10,R11>                 ; SAVE WORK REGS
        MOVZBL  $$ARG$$,R1                      ; USE R1 FOR ARG COUNT
        MOVAL   $$ARG$$+4,R9                    ; SETUP ARG LIST  FILL
        MOVL    (R8),(R9)+                              ; STORE 1ST ARG IN MSG LIST
        DECL    R1                              ; DECREMENT ARG COUNT
        BEQL    PREP                            ; AND DROP IF LAST ARG
        MOVL    (R2),(R9)+                              ; ETC.,
        DECL    R1
        BEQL    PREP
        MOVL    (R3),(R9)+
        DECL    R1
        BEQL    PREP
        MOVL    (R4),(R9)+
        DECL    R1
        BEQL    PREP
        MOVL    (R5),(R9)+
        DECL    R1
        BEQL    PREP
;
PREP:   MOVAL   RMSG,R10                        ; PROVIDE MSG ADDRESS
        MOVAL   ARGL,R11                        ; AND ARG LIST
        JSB     $$DISP$$                        ; DISPLAY THE MSG
        POPR    #^M<R9,R10,R11>                 ; RESTORE THE MSG REGS
;
        RSB                                     ; RETURN TO CALLER
;
;
;       THE 'DISPLAY' SUBROUTINE PERFORMS A FORMATTED WRITE
;       TO THE TERMINAL.
;
;       INPUT:  R10 -> FAO CONTRL STRING (LENGTH IN FIRST BYTE)
;               R11 -> LIST OF FAO ARGUMENTS
;
;       OUTPUT: FORMATTED MSG IN BUFFER 'OUTB'
;               LENGTH OF SAME IN LONGWORD 'OUTL'
;
$$DISP$$:
        $FAOL_S (R10),OUTL,OUTD,(R11) ; SETUP TERM OUTPUT LINE
;
;       FOR THE FOLLOWING PUTMSG CALL 'OUTL' IS USED FOR THE LENGTH OF
;       THE TERMINAL WRITE. $FAO STORES THE LENGTH OF THE FORMATTED
;       OUTPUT LINE IN 'OUTL' FOR JUST SUCH USE.
;
        MOVW    OUTL,OUTD       ; ACTUAL OUTPUT LENGTH IN STRING DESCRIPTOR
        PUTMSG  <#UETP$_TEXT,#1,#OUTD> ; PRINT THE MSG
        MOVW    #OUTE-OUTB,OUTD ; GET MAX LENGTH BACK INTO DESCRIPTOR
;
        RSB                             ; RETURN TO CALLER
;
END:
        .DSABL  LSB
;
        .MACRO  DISPSERV        ; REDEFINE DISPSERV TO NULL
        .ENDM                   ; ... SO IT WILL BE DELETED AFTER USE
        .ENDM                   ; END OF 'DISPSERV' DEFINITION
        .MACRO  ERR_EXIT  CONT,MT1,MT2,MT3,P,?LAB1
        .IF DIF,P,PCV           ; IF PCV NOT SPECIFIED
        MOVAL   .-1,PCV         ; SAVE PC VALUE FOR MSG4
        .ENDC
        BRB     LAB1
$$$$=.
        STRING  I,<MT1>,<MT2>,<MT3>
LAB1:
        MOVAL   $$$$,MSG_A      ; GET ADDR OF STRING DESC'R FOR FAO (MSG3)
        MOVB    #'CONT,MSG_CTXT ; PREPARE CONTEXT FOR EXP/REC MSGS (MSG4)
        BSBW    PROCESS_ERR     ; PERFORM MORE ERROR PROCESSING
        RSB                     ; ... AND EXIT THIS SUBROUTINE
        .ENDM
        .MACRO  GRPLABEL  TCG_NO
TC'TCG_NO:
        .ENDM
```

```
.MACRO  IDENT   NAME,VERSION
.NCHR   $$$IDN  VERSION
IDENT2  NAME,\$$$IDN
.ENDM
.MACRO  IDENT2  NAME2,VERS2
        .IDENT  /NAME2'VERS2/
.ENDM
```

```
        .MACRO  MODE    DEST,BR,M,NR,?CONTIN
        .IF IDN,DEST,FROM       ; IF DESTINATION IS FROM
                RET             ; RETURN FROM CHMRTN
BR:
                .MEXIT          ; EXIT FROM MACRO EXPANSION
        .ENDC
        .IF DIF,DEST,TO         ; IF DESTINATION NOT TO
                .ERROR ; DESTINATION MUST BE FROM OR TO
                .MEXIT          ; QUIT MACRO EXPANSION
        .ENDC
        MOVAL   CONTIN,CHM_CONT ; GET RETURN ADDR STORED FOR CHMRTN
        .IF IDN,NR,NOREGS       ; IF NR ARGUMENT = NOREGS
        $CM'M'_S  CHMRTN        ; EXECUTE CHMRTN IN NEW MODE
        .IFF                    ; IF NR NOT = NOREGS
        BSBW    SAVE_REGS       ; SAVE REGS R2 THROUGH R6
        $CM'M'_S  CHMRTN        ; EXECUTE CHMRTN IN NEW MODE
        BSBW    REST_REGS       ; RESTORE REGS R2 THROUGH R6
        .ENDC
        BRW     BR              ; BRANCH TO INSTR FOLLOWING "MODE FROM,..."
CONTIN:
        .ENDM
        .MACRO  MOV_VAL CONTEXT,SRC,DST
        .IF EQ CONTEXT - BYTE           ; EXPAND IF BYTE CONTEXT
        MOV_VAL2        B,SRC,DST
        .ENDC
        .IF EQ CONTEXT - WORD           ; EXPAND IF WORD CONTEXT
        MOV_VAL2        W,SRC,DST
        .ENDC
        .IF EQ CONTEXT - LONG           ; EXPAND IF LONG CONTEXT
        MOV_VAL2        L,SRC,DST
        .ENDC
        .IF EQ CONTEXT - QUAD           ; EXPAND IF QUAD CONTEXT
        MOV_VAL2        Q,SRC,DST
        .ENDC
        .IF EQ CONTEXT - DESC           ; EXPAND IF DESC CONTEXT
        MOV_VAL2        Q,SRC,DST
        .ENDC
        .ENDM
        .MACRO  MOV_VAL2  SHORTCTX,SRC,DST
        MOVA'SHORTCTX   SRC,DST
        .ENDM
        .MACRO  MSG     MSGTEXT1,MSGTEXT2,MSGTEXT3,?$$$
        BRB     $$$
$$$$=.
        STRING  M,<MSGTEXT1>,<MSGTEXT2>,<MSGTEXT3>
$$$:
        $OUTPUT CHANNEL,$$$$,<$$$$+8>
        .ENDM
        .PAGE
```

```
.MACRO  NEXT_TEST_CASE  TCN,?$$$
        .IF EQ $$$FIRSTTC$$$
            MOVAL       $$$,CURRENT_TC   ; GET ADDR OF NEXT T.C.
            RSB         ; RETURN TO FINISH UP THIS T.C.
        .ENDC
$$$:
        .PAGE
        .SBTTL  TCN
        $$$FIRSTTC$$$ = 0
        .ENDM

;++
; FUNCTIONAL DESCRIPTION:
;
;       Macro to set or clear privileges
;
; CALLING SEQUENCE:
;
;       PRIV ADDREM,SYSPRV
;
; FORMAL PARAMETERS:
;
;       ADDREM
;           Parameter indicating whether the privilege is to be added or
;           deleted. Must be specified as ADD or REM.
;
;       SYSPRV
;           Name of the privilege to be changed
;
;--

.MACRO PRIV,REMADD,PR
        .NARG PRIV_ARGS
        .IF LESS_THAN,PRIV_ARGS-2
        .ERROR ; PRIV parameters missing
        .IF FALSE
          .IF IDENTICAL,REMADD,REM
            .IF IDENTICAL,PR,ALL
              MOVQ      #-1,-(SP)                   ; Remove all privilege mask
            .IF FALSE
              MOVQ      #1@PRV$V_'PR',-(SP)         ; Create the privilege mask
            .ENDC
            MOVL        SP,R0
            $SETPRV_S   ENBFLG = #0,-               ; Clear the privilege
                        PRVADR = (R0)
            ADDL2       #8,SP                       ; Fix the SP
          .IF FALSE
            .IF IDENTICAL REMADD,ADD
              .IF IDENTICAL,PR,ALL
                MOVQ    #-1,-(SP)                   ; Add all privilege mask
              .IF FALSE
                MOVQ    #1@PRV$V_'PR',-(SP)         ; Create the privilege mask
              .ENDC
                MOVL    SP,R0
                $SETPRV_S       ENBFLG = #1,-       ; Set the privilege
                                PRVADR = (R0)
                ADDL2   #8,SP                   ; Fix the SP
            .IF_FALSE
              .ERROR ; Must specify REM or ADD as first argument
            .ENDC
          .ENDC
        .ENDC
.ENDM PRIV
```

```
        .MACRO  PUTMSG  ARGS
;+
;
;       PUTMSG is a synonym for $SIGNAL_S -- i.e., PUTMSG
;       merely issues $SIGNAL_S, passing along its set
;       of arguments to $SIGNAL_S.  The arguments specified
;       on the PUTMSG invocation line must be bounded by a
;       single set of angle brackets, so that they appear
;       as a single argument to PUTMSG, but as multiple
;       arguments to $SIGNAL_S.
;
;-
                $SIGNAL_S       ARGS
        .ENDM
```

```
;*****************************************************************************
;
        .MACRO  SERVCALL  SERVNAM,MSG,OPTNS,TABR,?SLOOP,?SADD,?SCALC,?SSAVE,-
                        ?SERR
;
;*****************************************************************************
        INCL    $$ASEQ$$                ; INCR ARGUMENT NUMBER
        MOVL    (R9)+,R11               ; IF NO INVALID ARGS BEING TESTED
        BNEQ    SCALC                   ; IF NOT, CONTINUE...
        TSTL    (R9)+                   ; IF SO, BUMP PAST CODES PTR
        BRW     SADD                    ; THEN SKIP ALL SERVICE CALLS
SCALC:  ASHL    #2,R11,R11              ; NO.PARMS X ENTRY SIZE(4)...
        ADDL2   TABR,R11                        ; + TAB STRT ADDR = TAB LIMIT
        ADDL2   #4,TABR                 ; NOW POINT TO 1ST INVALID PARM
        MOVL    (R9)+,R10               ; R10 -> RETURN CODES TABLE
        MOVL    $$INIT$$,$$PSEQ$$       ; RESET INITIAL PARM NO.
SLOOP:
        ADDL    #1,$$PSEQ$$             ; INCR TEST PARAMETER COUNT
        .IF     IDN,OPTNS,SATS
        JSB     @(SP)+                  ; VISIT MASTER ROUTINE
        .ENDC
        .IF EQ,NSSARGS-1                ; THESE NESTED .  S GENERATE
            $'SERVNAM'_S @(R8)           ; ... A SYSTEM SERVICE CALL
        .IFF                              ; ... WI"H THE CORRECT NUMBER
            .IF EQ,NSSARGS-2               ; ... OF ARGUMENTS
                $'SERVNAM'_S @(R8),@(R2)
            .IFF
                .IF EQ,NSSARGS-3
                    $'SERVNAM'_S @(R8),@(R2),@(R3)
                .IFF
                    .IF EQ,NSSARGS-4
                        $'SERVNAM'_S @(R8),@(R2),@(R3),@(R4)
                    .IFF
                        .IF EQ,NSSARGS-5
                            $'SERVNAM'_S @(R8),@(R2),@(R3),@(R4),@(R5)
                        .ENDC
                    .ENDC
                .ENDC
            .ENDC
        .ENDC
        .IF     IDN,OPTNS,SATS
        JSB     @(SP)+                  ; VISIT MASTER ROUTINE
        .ENDC
        CMPL    R0,(R10)+               ; EXPECTED RETURN CODE?
        .IF     IDN,MSG,ERR
        BEQL    SADD                    ; OK IF EQUAL...
        .ENDC
        .IF     IDN,OPTNS,SATS
        BNEQ    SERR                    ; CONFLICT...
        MOVB    #^A' ',$$TSTN$$+2       ; PROVIDE 'CORRECT' MSG CODE
        BRB     SSAVE                   ; AND DROP THRU...
SERR:   MOVB    #^A'*',$$TSTN$$+2       ; SIGNAL 'ERROR' MSG CODE
SSAVE:
        .ENDC
        MOVL    R0,$$ACT$$              ; SAVE R0 FOR ''MSG'' INFO
        MOVL    -4(R10),$$EXP$$         ; SAVE EXPECTED RETURN ALSO
        JSB     $$ERR$$                 ; GOTO ERR MSG RTN IN 'DISPSERV'
;
$ADD:   ACBW    R11,#4,TABR,SLOOP       ; LOOP TESTING ALL PARMS
                                        ; FOR THIS ARG AND LEAVE POINTING
                                        ; TO THE VALID(LAST) ENTRY IN TABLE
        .ENDM                   ; END OF 'SERVCALL' DEFINITION
;
```

```
        .MACRO  SERVCHEK ?LAB1,?LAB2
LAB1:   BLBS    R0,LAB2         ; IF SUCCESSFUL, CONTINUE
        MOVAL   LAB1,DIEARG             ; IF ERROR, SAVE PC FOR MSG
        BRW     DIE                     ; AND GOTO 'DIE' ROUTINE.
LAB2:
        .ENDM
        .MACRO  SS_CHECK  SC,?LAB1,?LAB2
        TSTB    EFLAG               ; IS AN ERROR ALREADY BEING PROCESSED ?
        BNEQ    LAB1                ; YES -- GO EXIT THIS SUBROUTINE
        MOVAL   .-9,PCV             ; GET APPROX LOCATION OF SYSTEM SERVICE
        MOVL    #SS$_'SC',EXPV      ; LOAD UP EXPECTED AND ...
        MOVL    R0,RECV             ; ... RECEIVED VALUES
        BSBW    COMP_SC             ; PERFORM STATUS CODE COMPARE
        TSTB    EFLAG               ; DID COMP_SC FIND AN ERROR ?
        BEQL    LAB2                ; NO -- JUST CONTINUE
LAB1:
        RSB                         ; EXIT THIS SUBROUTINE
LAB2:
        .ENDM
```

```
;
;++
; ***************************************************************
; *
; *      MACRO STGRP GENERATES A GROUP OF STRING DESCRIPTORS
; *      AND THEIR ASSOCIATED STRINGS.  THE DESCRIPTORS
; *      ARE CONTIGUOUS AND PRECEDE ALL STRINGS, WHICH
; *      ARE ALSO CONTIGUOUS.  THE FIRST ARGUMENT FOR THE
; *      MACRO IS A TYPE CODE (I OR C) WHICH INDICATES
; *      WHETHER THE STRINGS ARE DESCRIBED BY QUADWORD
; *      STRING DESCRIPTORS (I), OR ARE COUNTED STRINGS (C).
; *      IN THE CASE OF COUNTED STRINGS, THE CONTIGUOUS
; *      DESCRIPTOR BLOCK IS MADE UP OF LONGWORD POINTERS
; *      TO THE COUNTED STRINGS, ONE POINTER PER STRING.
; *      THE BYTE COUNT IS RETAINED AS THE FIRST BYTE OF
; *      THE STRING.  THE COUNT DOES NOT INCLUDE THIS BYTE.
; *      THE TYPE CODE ARGUMENT IS REQUIRED;
; *      THE ISSUER MAY THEN SPECIFY UP TO 5 CHARACTER
; *      STRINGS, EACH ONE A SEPARATE ARGUMENT
; *      BOUNDED BY ANGLE BRACKETS.
; *
; ****************************************************************
;--
;
        .MACRO  STGRP   TYP,ST1,ST2,ST3,ST4,ST5,?N1,?N2,?N3,?N4,?N5
        .IF DIF,TYP,I            ; NOT I ?
        .IF DIF,TYP,C            ; NOT C ?
        .ERROR ; INVALID STGRP TYPE -- MUST BE I OR C
        .MEXIT
        .ENDC
        .ENDC
        .NARG   $$$STRINGS       ; GET NO. OF STRINGS + ANOTHER ARG
        $$$STRINGS=$$$STRINGS-1 ; GET NUMBER OF STRINGS (5 MAX)
        $$$STRINGS2=$$$STRINGS  ; REMEMBER IT FOR LATER
        .IF NE $$$STRINGS
        .NCHR   $$$CHARS1,<ST1> ; GET NUMBER OF CHARS FOR STRING 1
        .IF NE $$$CHARS1
        .IF IDN,TYP,I            ; IF STRING DESC'R, GENERATE LWORD CNT
        .LONG   $$$CHARS1        ; LENGTH OF STRING
        .ENDC
        .ADDRESS N1              ; POINTER TO STRING
        .ENDC
        $$$STRINGS=$$$STRINGS-1 ; DECREMENT STRING COUNT
        .IF NE $$$STRINGS
        .NCHR   $$$CHARS2,<ST2> ; GET NUMBER OF CHARS FOR STRING 2
        .IF NE $$$CHARS2
        .IF IDN,TYP,I            ; IF STRING DESC'R, GENERATE LWORD CNT
        .LONG   $$$CHARS2        ; LENGTH OF STRING
        .ENDC
        .ADDRESS N2              ; POINTER TO STRING
        .ENDC
        $$$STRINGS=$$$STRINGS-1 ; DECREMENT STRING COUNT
        .IF NE $$$STRINGS
        .NCHR   $$$CHARS3,<ST3> ; GET NUMBER OF CHARS FOR STRING 3
        .IF NE $$$CHARS3
        .IF IDN,TYP,I            ; IF STRING DESC'R, GENERATE LWORD CNT
        .LONG   $$$CHARS3       ; LENGTH OF STRING
        .ENDC
        .ADDRESS N3             ; POINTER TO STRING
        .ENDC
        $$$STRINGS=$$$STRINGS-1 ; DECREMENT STRING COUNT
        .IF NE $$$STRINGS
        .NCHR   $$$CHARS4,<ST4> ; GET NUMBER OF CHARS FOR STRING4
        .IF NE $$$CHARS4
        .IF IDN,TYP,I           ; IF STRING DESC'R, GENERATE LWORD CNT
        .LONG   $$$CHARS4       ; LENGTH OF STRING
        .ENDC
        .ADDRESS N4             ; POINTER TO STRING
        .ENDC
        $$$STRINGS=$$$STRINGS-1 ; DECREMENT STRING COUNT
        .IF NE $$$STRINGS
        .NCHR   $$$CHARS5,<ST5> ; GET NUMBER OF CHARS FOR STRING 5
        .IF NE $$$CHARS5
        .IF IDN,TYP,I           ; IF STRING DESC'R, GENERATE LWORD CNT
        .LONG   $$$CHARS5       ; LENGTH OF STRING
        .ENDC
        .ADDRESS N5             ; POINTER TO STRING
        .ENDC
        $$$STRINGS=$$$STRINGS-1 ; DECREMENT STRING COUNT
        .IF NE $$$STRINGS       ; IF STRINGS HAS NOT GONE TO 0, TOO MANY
        .ERROR  ; TOO MANY STRINGS SPECIFIED (5 MAX)
        .ENDC
        .ENDC
        .ENDC
        .ENDC
        .ENDC
        .ENDC
        $$$STRINGS=$$$STRINGS2  ; GET BACK ORIGINAL STRING COUNT
        .IF NE $$$STRINGS
        .IF NE $$$CHARS1
N1:
        .IF IDN,TYP,C           ; IF COUNTED STRING, INSERT BYTE CNT
        .BYTE   $$$CHARS1       ; STRING COUNT
        .ENDC
        .ASCII  \ST1\           ; CHARACTER STRING
        .ENDC
        $$$STRINGS=$$$STRINGS-1 ; DECREMENT STRING COUNT
        .IF NE $$$STRINGS
        .IF NE $$$CHARS2
N2:
        .IF IDN,TYP,C           ; IF COUNTED STRING, INSERT BYTE CNT
        .BYTE   $$$CHARS2       ; STRING COUNT
        .ENDC
        .ASCII  \ST2\           ; CHARACTER STRING
        .ENDC
        $$$STRINGS=$$$STRINGS-1 ; DECREMENT STRING COUNT
        .IF NE $$$STRINGS
        .IF NE $$$CHARS3
N3:
        .IF IDN,TYP,C           ; IF COUNTED STRING, INSERT BYTE CNT
        .BYTE   $$$CHARS3       ; STRING COUNT
        .ENDC
        .ASCII  \ST3\           ; CHARACTER STRING
        .ENDC
        $$$STRINGS=$$$STRINGS-1 ; DECREMENT STRING COUNT
        .IF NE $$$STRINGS
        .IF NE $$$CHARS4
N4:
        .IF IDN,TYP,C           ; IF COUNTED STRING, INSERT BYTE CNT
        .BYTE   $$$CHARS4       ; STRING COUNT
        .ENDC
        .ASCII  \ST4\           ; CHARACTER STRING
        .ENDC
        $$$STRINGS=$$$STRINGS-1 ; DECREMENT STRING COUNT
        .IF NE $$$STRINGS
        .IF NE $$$CHARS5
N5:
        .IF IDN,TYP,C           ; IF COUNTED STRING, INSERT BYTE CNT
        .BYTE   $$$CHARS5       ; STRING COUNT
        .ENDC
        .ASCII  \ST5\           ; CHARACTER STRING
        .ENDC
        $$$STRINGS=$$$STRINGS-1 ; DECREMENT STRING COUNT
        .ENDC
        .ENDC
        .ENDC
        .ENDC
        .ENDC
        .ENDM
        .MACRO  STRING STYPE,STRING1,STRING2,STRING3
;
;       THE STRING MACRO GENERATES ASCII STRING DATA
;       IN ONE OF FOUR FLAVORS, DEPENDING ON A SUPPLIED
;       CODE.  THE FORMAT IS:
;
;               STRING  CODE,<STRING1>,<STRING2>,<STRING3>
;
;          WHERE CODE IS A ONE-CHARACTER CODE DENOTING THE
;          TYPE OF STRING, AND STRING1, STRING2, STRING3
;          ARE SEGMENTS OF THE STRING TO BE CREATED.  THE
;          SEGMENTS ARE CONCATENATED TO FORM A SINGLE STRING.
;          THE STRING SEGMENTS MAY CONTAIN ANY ASCII CHARACTER
;          ALLOWABLE IN A .ASCII ASSEMBLER DIRECTIVE (EXCEPT
;          BACKSLASH, WHICH IS USED BY THE MACRO AS A DELIMITER).
;          THE CODE IS INTERPRETED AS FOLLOWS:
;
;               I -- GENERATES A QUADWORD STRING DESCRIPTOR
;                    FOLLOWED IMMEDIATELY BY THE SPECIFIED STRING.
;
;               M -- SAME AS I, WITH CR, LF CHARACTERS APPENDED
;                    TO SPECIFIED STRING.
;
;               O -- GENERATES A QUADWORD STRING DESCRIPTOR
;                    FOLLOWED IMMEDIATELY BY A STRING BUFFER.
;                    YOU DO NOT SPECIFY A STRING WITH THIS CODE
;                    BUT, INSTEAD, USE THE STRING1 ARGUMENT TO
;                    SPECIFY A LENGTH FOR THE STRING BUFFER.
;                    THE LENGTH WILL BE STORED IN THE FIRST
;                    LONGWORD OF THE DESCRIPTOR; THE STRING
;                    BUFFER WILL NOT BE INITIALIZED.
;
;               C -- GENERATES A COUNTED ASCII STRING (ONE BYTE
;                    COUNT FOLLOWED BY SPECIFIED STRING).
;
        .IF IDN,STYPE,O         ; IS IT O ?
        .LONG   STRING1
        .ADDRESS .+4
        .BLKB   STRING1
        .IFF
        .IF DIF,STYPE,I         ; NOT I ?
        .IF DIF,STYPE,M         ; NOT M ?
        .IF DIF,STYPE,C         ; NOT C ?
        .ERROR ; INVALID STRING TYPE -- MUST BE I, O, M, OR C
        .MEXIT
        .ENDC
        .ENDC
        .ENDC
        .NARG   $$$STRINGS
        $$$STRINGS=$$$STRINGS-1
        .NCHR   $$$CHARS,<STRING1'STRING2'STRING3>
        .IF IDN,STYPE,M         ; IS IT M ?
        $$$CHARS=$$$CHARS+2             ; INCR FOR CR, LF
        .ENDC
        .IF IDN,STYPE,C         ; IS IT C ?
        .BYTE   $$$CHARS
```

```
	.IFF
	.LONG	$$$CHARS
	.ADDRESS .+4
	.ENDC
	.IF NE $$$STRINGS
	.ASCII	\STRING1\
	$$$STRINGS=$$$STRINGS-1
	.IF NE $$$STRINGS
	.ASCII	\STRING2\
	$$$STRINGS=$$$STRINGS-1
	.IF NE $$$STRINGS
	.ASCII	\STRING3\
	.ENDC
	.ENDC
	.ENDC
	.IF IDN,STYPE,M		; IS IT M ?
	.ASCII	<13><10>		; CR, LF
	.ENDC
	.ENDC
	.ENDM
```

```
; ****** THE STRINGO MACRO HAS BEEN REPLACED WITH STRING (O FORM).
        .MACRO  STRINGO SIZE
        .LONG   SIZE
        .ADDRESS .+4
        .BLKB   SIZE
        .ENDM
```

```
.MACRO  TCEND
        RSB
        .PAGE
        .SBTTL
        .ENDM
        .MACRO TCJUMP  TCG_NO
        JSB     TC'TCG_NO       ; JUMP TO TEST CASE GROUP INIT
        .ENDM
```

```
.MACRO  TCSTART
        $$$FIRSTTC$$$ = 1
FIRST_TC:
        .ENDM
        .MACRO TC_GROUP  SS3,SEQ_SS,TSNAME,?$$$
        $$$FIRSTTC$$$ = 1         ; INDICATE FIRST T.C. FOR THIS GROUP
        GRP_TOTAL = GRP_TOTAL+1  ; INCR T.C. GROUP TOTAL
        GRPLABEL        \GRP_TOTAL ; LABEL BEGINNING OF GROUP
        MOVAL   $$$,CURRENT_TC  ; EST FIRST TEST CASE AS CURRENT
        MOVAL   TSNAME,TS_EP    ; EST TESTSERV ENTRY POINT
        MOVB    #'SEQ_SS,$$CALL$$ ; IND SEQ NO OF THIS GROUP WITHIN THIS SS
        MOVL    #^A/F'SS3/,$$TSTN$$+4 ; EST ASCII T.C. NAME
        RSB
$$$:
        .ENDM
```

```
;****************************************************************************
;****************************************************************************
        .MACRO  TESTSERV        SERVNAM,MSG=ERR,OPTN,PSET1,PSET2,PSET3,PSET4,-
                                PSET5,DUMMY1,?SERVNAML,?START,?ARGBLOK
;
; !!NOTE!! -- When TESTSERV is invoked, its argument list must be followed by a
;             trailing comma, this generating  an extraneous null argument.  In
;             addition, a trailing comma must appear following each sub-argument
;             list ( i.e., PSET1 through PSET5).
;
;
;
        .NARG   NARGS
        NARGS = NARGS - 1               ; Decrement fot trailing null argument
        NSSARGS = NARGS-3               ; CALCULATE NO. OF ARGS REQUIRED BY
                                        ; ... SYST. SERV. -- USED IN SERVCALL MACRO
        .IF GT NARGS-8
        .ERROR NARGS ; TOO MANY (>8) ARGUMENTS SUPPLIED.;
        .MEXIT
        .ENDC
;
        .IF LT NARGS-4
        .ERROR NARGS ; INSUFFICIENT (<4) ARGUMENTS SUPPLIED.;
        .MEXIT
        .ENDC
;
        .IF     DIF,MSG,ERR
        .IF     DIF,MSG,ALL
        .ERROR  ; MISSING OR INVALID ERROR DISPOSITION ARGUMENT.;
        .MEXIT
        .ENDC
        .ENDC
;
        .IF     NB OPTN
        .IF     DIF,OPTN,SATS
        .ERROR  ; INVALID OPTION ARGUMENT.;
        .MEXIT
        .ENDC
        .ENDC
;
        .IF     NDF $$ERR$$
        .ERROR  ; NO PRECEDING 'DISPSERV' MACRO CALL.;
        .MEXIT
        .ENDC
;
        .IF     B OPTN
        .NCHR   NCHRS,SERVNAM
        MOVB    #NCHRS,SERVNAML         ; SAVE NO. CHARS IN SERVICE NAME
        MOVAL   SERVNAML,$$SNAD$$       ; KEEP ADDR OF SERVICE NAME
        .ENDC
        .IF     IDN,OPTN,SATS
        MOVAL   $$TSTN$$,$$SNAD$$       ; GET TEST NAME FROM MASTER
        .ENDC
;
        MOVZBL  #NARGS-3,$$ARG$$        ; KEEP NO. SERVICE ARGS
$$MAXP$$= 5                     ; MAXIMUM NUMBER OF TEST PARMS PER ARG
        CLRL    $$ASEQ$$                ; RESET ARGUMENT SEQUENCE NUMBER
        CLRL    $$INIT$$                ; RESET TEST PARM SEQUENCE NUMBER
        .IF     IDN,OPTN,SATS
        SUBL3   #1,$$CALL$$,$$INIT$$    ; ADJUST CONSEC.CALL NO. (PASSED BY SATS)
        MULL2   #$$MAXP$$,$$INIT$$      ; CALL NO. X MAX PARM NO.= NEW START NO.
        DECL    $$INIT$$                ; COMPENSATE FOR 'INCL' LATER
        .ENDC
        BRW     START                   ; SKIP AROUND NAME
SERVNAML:.BYTE  0                       ; LENGTH OF SERVICE NAME
        .ASCII  \SERVNAM\               ;
ARGBLOK:
        .REPT   NARGS-3
        .BLKL   2                       ; ARGUMENT INFO.
        .ENDR
;
;
START:
;
        MOVAL   ARGBLOK,R9              ; POINT R9 TO TOP OF ARG BLOK
;
        .IF NB <PSET1>
        DECODE  ,,,,,,,,,,,PSET1        ; DECODE 1ST ARG SET...
        .IIF EQ MEXIT-1, .MEXIT
        MOVL    R10,R8                  ; SAVE TEST PARM TABLE ADDR
        .ENDC
;
        .IF NB <PSET2>
        DECODE  ,,,,,,,,,,,PSET2        ; DECODE 2ND ARG SET...
        .IIF EQ MEXIT-1, .MEXIT
        MOVL    R10,R2                  ; SAVE TEST PARM TABLE ADDR
        .ENDC
;
        .IF  NB <PSET3>
        DECODE  ,,,,,,,,,,,PSET3        ; DECODE 3RD ARG SET...
        .IIF EQ MEXIT-1, .MEXIT
        MOVL    R10,R3                  ; SAVE TEST PARM TABLE ADDR
        .ENDC
;
        .IF NB <PSET4>
        DECODE  ,,,,,,,,,,,PSET4        ; DECODE 4TH ARG SET...
        .IIF EQ MEXIT-1, .MEXIT
        MOVL    R10,R4                  ; SAVE TEST PARM TABLE ADDR
        .ENDC
;
        .IF NB <PSET5>
        DECODE  ,,,,,,,,,,,PSET5        ; DECODE 5TH ARG SET...
        .IIF EQ MEXIT-1, .MEXIT
        MOVL    R10,R5                  ; SAVE TEST PARM TABLE ADDR
        .ENDC
;
;
;       PARAMETER TABLES (1 PER 1ST LEVEL ARGUMENT) ARE NOW BUILT
;
;
        MOVAL   ARGBLOK,R9              ; RE-INIT ARG BLOK POINTER
;
        .IF NB <PSET1>
```

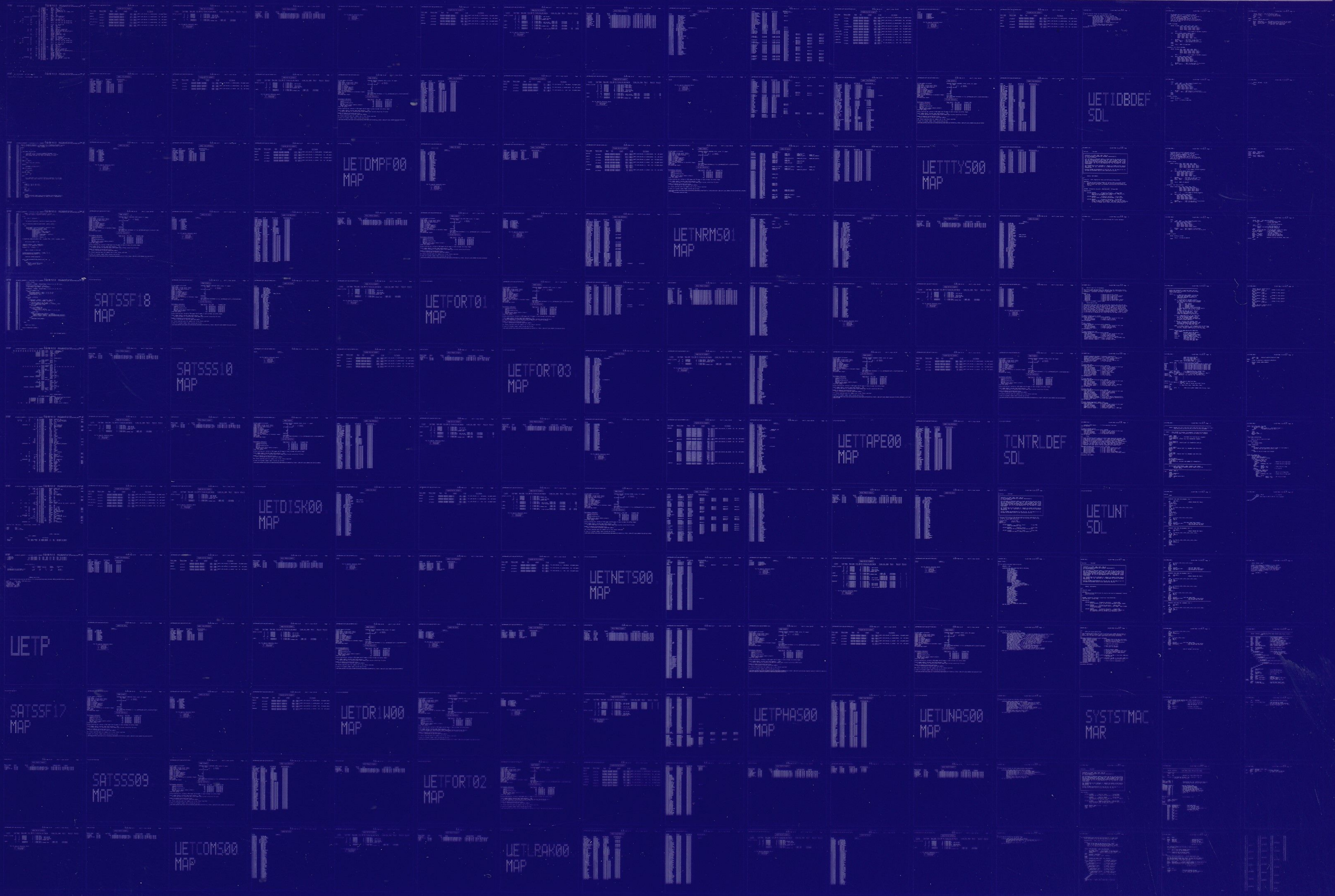
RMS
028

```
        SERVCALL SERVNAM,MSG,OPTN,R8    ; ISSUE SERV LOOP FOR 1ST ARG
        .ENDC
;
        .IF NB <PSET2>
        SERVCALL SERVNAM,MSG,OPTN,R2    ; ISSUE SERV CALLS TO TEST 2ND..
        .ENDC
;
        .IF NB <PSET3>
        SERVCALL SERVNAM,MSG,OPTN,R3    ; ISSUE SERV CALLS TO TEST 3RD...
        .ENDC
;
        .IF NB <PSET4>
        SERVCALL SERVNAM,MSG,OPTN,R4    ; ISSUE SERV CALLS TO TEST 4TH...
        .ENDC
;
        .IF NB <PSET5>
        SERVCALL SERVNAM,MSG,OPTN,R5    ; ISSUE SERV CALLS TO TEST 5TH...
        .ENDC
;
        .ENDM                           ; END OF 'TESTSERV' DEFINITION
;
;*******************************************************************************
;*******************************************************************************
```

```
        .MACRO TEST_SERV_EXEC
        .REPT   GRP_TOTAL
        TCG_NO = TCG_NO+1          ; INCR TEST CASE GROUP NUMBER
        TCJUMP  \TCG_NO ; PERFORM TEST CASE GROUP INITIALIZATION
        JSB     TC_CONTROL  ; RUN ALL T.C.'S FOR THIS GROUP
        .ENDR
        .ENDM
        .MACRO  TS_CLEANUP      ?$$$
        CMPB    #^A/*/,$$TSTN$$+2 ; DID FINAL SERVICE CALL FAIL ?
        BNEQU   $$$               ; NO -- CONTINUE
        MOVAL   TEST_MOD_FAIL,TMD_ADDR ; YES -- INDICATE FAILED IN END MSG
        INSV    #ERROR,#0,#3,MOD_MSG_CODE ; ADJUST STATUS CODE FOR ERROR
$$$:
        TSTL    (SP)+             ; POP TS ADDRESS WHICH WAS PUSHED ABOVE
        RSB                       ; RETURN TO TEST_SERV_EXEC MACRO
        .PAGE
        .ENDM
;
; Macro to test for the correct error code return
;
        .MACRO FAIL_CHECK FAIL_CODE
        .LIST MEB
                PUSHL   #FAIL_CODE      ; push desired failure code
                CALLS   #1,W^REG_CHECK  ; check all registers for correct content
        .NLIST MEB
        .ENDM   FAIL_CHECK
;
; Macro to test for the correct error code return without printing it
;
        .MACRO FAIL_CHECKNP FAIL_CODE
        .LIST MEB
                PUSHL   #FAIL_CODE      ; push desired failure code
                CALLS   #1,W^REG_CHECKNP ; check all registers for correct content but
                                         ; don't print out the failures
        .NLIST MEB
        .ENDM   FAIL_CHECKNP
;
; Macro to generate a label using the current step number.
;
        .MACRO LABEL N
        .LIST ME
STP'N:
                MOVL    #N,W^CURRENT_TC ; save the test case number
        .NLIST ME
        .ENDM   LABEL
;
; Macro to declare the start of a subtest
;
        .MACRO NEXT_TEST
        STEP=STEP+1
        LABEL   \STEP
        .LIST MEB
                PUSHL   #0                      ; push a dummy parameter
                CALLS   #1,W^REG_SAVE           ; save the registers
        .NLIST MEB
        .ENDM   NEXT_TEST
;
; Macro to end a test
;
        .MACRO TEST_END
        .LIST MEB
                PUTMSG  <W^MOD_MSG_CODE,#2,W^TMN_ADDR,W^TMD_ADDR> ; type ending message
                INSV    #1,#STS$V_INHIB_MSG,#1,W*MOD_MSG_CODE ; inhibit p-inting
                $EXIT S W^MOD_MSG_CODE                          ; exit to the O.S.
        .NLIST MEB
        .ENDM TEST_END
;
; macro to start a test
;
        .MACRO TEST_START TEST_NAME
        .PAGE
        .LIST ME
                .ENTRY TEST_NAME,0      ; entry mask
        .NLIST ME
        STEP=0
        .LIST MEB
                CLRL    W^CURRENT_TC    ; set initial test step
        $WAKE_S  W^TPID                 ; get pid of this process
        $HIBER_S                        ; undo wake
        $SETPRN_S W^TEST_MOD_NAME_D     ; set process name
                BSBW    W^MOD_MSG_PRINT ; print test module begin msg
                MOVAL   W^TEST_MOD_SUCC,W^TMD_ADDR ; assume end msg will show success
                INSV    #SUCCESS,#0,#3,W^MOD_MSG_CODE ; adjust status code for success
                PUSHL   #0              ; push a dummy parameter
                CALLS #1,W^REG_SAVE     ; save the registers
STP0:
        .NLIST MEB
        .ENDM   TEST_START
```

0408 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



UETIDBDEF
SDL

UETDMPF00
MAP

UETTTYS00
MAP

UETNRMS01
MAP

SATSSF18
MAP

UETFORT01
MAP

SATSSS10
MAP

UETFORT03
MAP

UETTAPE00
MAP

TCNTRLDEF
SDL

UETDISK00
MAP

UETUNT
SDL

UETNETS00
MAP

UETP

SATSSF17
MAP

UETDR1W00
MAP

UETPHAS00
MAP

UETUNAS00
MAP

SYSTSTMAC
MAR

SATSSS09
MAP

UETFORT02
MAP

UETCOMS00
MAP

UETLPAK00
MAP

0409 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RMSTEST4
LIS

RMSTEST7
LIS

RMSTEST8
LIS

RMSTESTR
LIS

RMSTEST0
LIS

RMSTEST2
LIS

RMSTEST3
LIS

RMSTEST5
LIS

RMSTEST6
LIS

RMSTESTT
LIS

RMSTEST1
LIS

SATSSF17
LIS