```
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
UUU          UUU  EEE                    TTT         PPP
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT         PPP
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT         PPP
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT         PPP
```

```
UU      UU  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD     IIIIII     SSSSSSSS  KK      KK    000000     000000
UU      UU  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD     IIIIII     SSSSSSSS  KK      KK    000000     000000
UU      UU  EE              TT      DD      DD     II     SS          KK      KK  00      00  00      00
UU      UU  EE              TT      DD      DD     II     SS          KK      KK  00      00  00      00
UU      UU  EE              TT      DD      DD     II     SS          KK    KK    00    0000  00    0000
UU      UU  EE              TT      DD      DD     II     SS          KK    KK    00    0000  00    0000
UU      UU  EEEEEEEE        TT      DD      DD     II       SSSSSS    KKKKKK      00  00  00  00  00  00
UU      UU  EEEEEEEE        TT      DD      DD     II       SSSSSS    KKKKKK      00  00  00  00  00  00
UU      UU  EE              TT      DD      DD     II            SS   KK    KK    0000    00  0000    00
UU      UU  EE              TT      DD      DD     II            SS   KK    KK    0000    00  0000    00
UU      UU  EE              TT      DD      DD     II            SS   KK      KK  00      00  00      00
UU      UU  EE              TT      DD      DD     II            SS   KK      KK  00      00  00      00      ....
UUUUUUUUUU  EEEEEEEEEE      TT      DDDDDDDD     IIIIII     SSSSSSSS  KK      KK    000000     000000      ....
UUUUUUUUUU  EEEEEEEEEE      TT      DDDDDDDD     IIIIII     SSSSSSSS  KK      KK    000000     000000      ....

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

```
0000     1          .TITLE  UETDISK00 - VAX/VMS UETP DISK EXERCISER
0000     2          .IDENT  'V04-000'
0000     3          .ENABLE SUPPRESSION
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:
0000    31  ;       This module will be distributed with VAX/VMS under the [SYSTEST]
0000    32  ;       account.
0000    33  ;
0000    34  ; ABSTRACT:
0000    35  ;       Exercise testable units of a disk controller by writing and verifying
0000    36  ;       miscellaneous blocks of test files on those units.
0000    37  ;
0000    38  ; ENVIRONMENT:
0000    39  ;       This program will run in user access mode, with ASTs enabled except
0000    40  ;       during error processing.  The program requires an AST limit of 55, a
0000    41  ;       buffered I/O limit of 55 and no privileges.
0000    42  ;
0000    43  ;--
0000    44  ;
0000    45  ; AUTHOR: Richard N. Holstein, CREATION DATE: August, 1981
0000    46  ;
0000    47  ; MODIFIED BY:
0000    48  ;
0000    49  ;       V03-011 DAS0001         David Solomon           29-Feb-1984
0000    50  ;               Remove inappropriate reference to FAB EDL bit.
0000    51  ;
0000    52  ;       V03-010 RNH0011         Richard N. Holstein,    15-Feb-1984
0000    53  ;               Take advantage of new UETP message codes.  Fix SSERROR
0000    54  ;               interaction with RMS_ERROR.  Fix ONE_SHOT_TEST register usage
0000    55  ;               for VERIFY_ERROR call.
0000    56  ;
0000    57  ;       V03-009 RNH0010         Richard N. Holstein,    19-Dec-1983
```

```
0000     58 ;                  Give correct  ntinels to Test Controller.
0000     59 ;
0000     60 ;       V03-008 RNH0009         Richard N. Holstein,    29-Nov-1983
0000     61 ;               Be more specific when cleaning up test data files - works better
0000     62 ;               if multiple tests run simultaneously.  Also when cleaning up,
0000     63 ;               terminate I/O cleanly to allow file erasure.
0000     64 ;
0000     65 ;       V03-007 RNH0008         Richard N. Holstein,    11-Nov-1983
0000     66 ;               Use decimal conversion routine for unit numbers.
0000     67 ;
0000     68 ;       V03-006 RNH0007         Richard N. Holstein,    05-Oct-1983
0000     69 ;               Get around an 11/750 Buffered Datapath bug which corrupts data,
0000     70 ;               and a driver patch which gives correct data at the expense of
0000     71 ;               data late errors being logged, by longword aligning the
0000     72 ;               read/write buffers.  This is V03-001 in the V3 sources.
0000     73 ;
0000     74 ;       V03-005 RNH0006         Richard N. Holstein,    26-May-1983
0000     75 ;               Exercise disks a bit more by setting the FOP EDL bit.
0000     76 ;
0000     77 ;       V03-004 RNH0005         Richard N. Holstein,    10-Mar-1983
0000     78 ;               Invoke SSERROR in AST routines to prevent some improperly
0000     79 ;               handled conditions.  Don't signal ending message in
0000     80 ;               EXIT_HANDLER.
0000     81 ;
0000     82 ;       V03-003 RNH0004         Richard N. Holstein,    06-Jan-1983
0000     83 ;               Allow for longer file names and longer device names.
0000     84 ;
0000     85 ;       V03-002 RNH0003         Richard N. Holstein,    15-Oct-1982
0000     86 ;               Miscellaneous fixes listed in the V3B UETP Workplan.
0000     87 ;
0000     88 ;       V03-001 RNH0002         Richard N. Holstein,    06-Aug-1982
0000     89 ;               Edit to bug source only.  See edit V03-006 above.
0000     90 ;
0000     91 ;       V02-001 RNH0001         Richard N. Holstein,    11-Dec-1981
0000     92 ;               Allow further flexibility in testing by recovering from most
0000     93 ;               RMS$_FUL situations when creating a file.  Include PATTERN to
0000     94 ;               force a definite pattern to disk.  Give a message when we've
0000     95 ;               completely stopped testing a unit.  Suppress leading zeroes
0000     96 ;               when reading and remembering disk unit numbers.  Clean up old
0000     97 ;               files before creating new ones.
0000     98 ;
0000     99 ;**
```

```
                        0000   101          .SBTTL  Declarations
                        0000   102 ;
                        0000   103 ; INCLUDE FILES:
                        0000   104 ;
                        0000   105 ;       SYS$LIBRARY:LIB.MLB      for general definitions
                        0000   106 ;       SHRLIB$:UETP.MLB         for UETP definitions
                        0000   107 ;
                        0000   108 ; MACROS:
                        0000   109 ;
                        0000   110          $CHFDEF                          ; Condition handler frame definitions
                        0000   111          $DEVDEF                          ; Device definitions
                        0000   112          $DIBDEF                          ; Device Information Block
                        0000   113          $DVIDEF                          ; $GETDVI ITMLST item codes
                        0000   114          $NAMDEF                          ; NAM block constants and offsets
                        0000   115          $SHRDEF                          ; Shared messages
                        0000   116          $SSDEF                           ; System Service status codes
                        0000   117          $STSDEF                          ; Status return
                        0000   118          $UETUNTDEF                       ; UETP unit block offset definitions
                        0000   119          $UETPDEF                         ; UETP
                        0000   120 ;
                        0000   121 ; EQUATED SYMBOLS:
                        0000   122 ;
                        0000   123 ;   Facility number definitions:
00000001                0000   124          RMS$_FACILITY = 1
                        0000   125
                        0000   126 ;   SHR message definitions:
00740000                0000   127          UETP = UETP$_FACILITY@STS$V_FAC_NO ; Define the UETP facility code
007410E0                0000   128          UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
00741038                0000   129          UETP$_BEGIND = UETP!SHR$_BEGIND
00741080                0000   130          UETP$_ENDEDD = UETP!SHR$_ENDEDD
00741098                0000   131          UETP$_OPENIN = UETP!SHR$_OPENIN
00741130                0000   132          UETP$_TEXT   = UETP!SHR$_TEXT
                        0000   133
                        0000   134 ;   Internal flag bits...:
00000001                0000   135          TEST_OVERV   = 1                 ; Set when test is over
00000002                0000   136          SAFE_TO_UPDV = 2                 ; Set if it's safe to update UETINIDEV
00000003                0000   137          BEGIN_MSGV   = 3                 ; Set if "BEGIN" msg has been printed
00000004                0000   138          ONE_SHOTV    = 4                 ; Set if running in one-shot mode
00000005                0000   139          PATTERNV     = 5                 ; Set if specific pattern data to write
                        0000   140 ;   ...and corresponding masks:
00000002                0000   141          TEST_OVERM   = 1@TEST_OVERV
00000004                0000   142          SAFE_TO_UPDM = 1@SAFE_TO_UPDV
00000008                0000   143          BEGIN_MSGM   = 1@BEGIN_MSGV
00000010                0000   144          ONE_SHOTM    = 1@ONE_SHOTV
00000020                0000   145          PATTERNM     = 1@PATTERNV
                        0000   146
                        0000   147 ;   Miscellany:
00000020                0000   148          LC_BITM         = ^X20           ; Mask to convert lower case to upper
00000028                0000   149          REC_SIZE        = 40             ; UETINIDEV.DAT record size
00000084                0000   150          TEXT_BUFFER     = 132            ; Internal text buffer size
00000004                0000   151          EFN2            = 4              ; EFN used for three minute timer
00000003                0000   152          SS_SYNCH_EFN    = 3              ; Synch miscellaneous system services
0000000F                0000   153          MAX_PROC_NAME   = 15             ; Longest possible process name
0000000A                0000   154          MAX_DEV_DESIG   = 10             ; Longest possible controller name
00000005                0000   155          MAX_UNIT_DESIG= 5                ; Longest possible unit number
```

```
                     0000   157 ;   Disk specific definitions:
00000800             0000   158         WRITE_SIZE      = 512*4          ; RSZ buffer size
00000800             0000   159         READ_SIZE       = WRITE_SIZE     ; USZ buffer size (both use same buffer)
CLCD3E4C             0000   160         PERCENTAGE      = ^F0.05         ; Percentage for incrementing files
00004040             0000   161         TOP_PCT         = ^F0.75         ; Maximum percentage of unit we use
00000027             0000   162         FN_LEN          = 39             ; Assumed maximum file name length
00000014             0000   163         BAKERS          = 20             ; Baker's dozen: determine when to expand
55555555             0000   164         DISK_K_PATTERN  = ^X55555555     ; Contents of test file FAB's CTX field
00000001             0000   165         DISK_K_CREATE   = 1              ; Code to say current func is $CREATE
00000002             0000   166         DISK_V_NOCREATE = UETUNT$V_TESTABLE+1 ; On if $CREATE failed
00000004             0000   167         DISK_M_NOCREATE = 1@DISK_V_NOCREATE
00000003             0000   168         DISK_V_NOEXTEND = DISK_V_NOCREATE+1 ; On if $EXTEND failed
00000008             0000   169         DISK_M_NOEXTEND = 1@DISK_V_NOEXTEND
                     0000   170
                     0000   171 ;
                     0000   172 ; For each file on each unit, there will be a data structure set up, called a
                     0000   173 ; node.  These nodes will be linked together in a self-relative queue whose
                     0000   174 ; header is UNIT_LIST.  The first part of each node will be the standard
                     0000   175 ; definition from $UETUNTDEF.  Following that will come the device test
                     0000   176 ; dependent stuff, defined below.  NOTE THAT THIS DEFINITION IS DONE WITH AN
                     0000   177 ; ABSOLUTE PSECT.  This means that what look like declarations are really
                     0000   178 ; definitions and the labels are really just offsets into a given node on the
                     0000   179 ; queue.  (A not necessarily obvious consequence of using an ABS PSECT is that
                     0000   180 ; space must be reserved with .BLKx operations, since .BYTE, etc., attempt to
                     0000   181 ; store data.)  This data structure differs from most other device tests in
                     0000   182 ; that there is more than one node per unit in normal and loop forever modes.
                     0000   183 ;
                     0000   184         .PSECT  DEVDEP_STR_DEF,ABS,NOEXE,NOWRT,PAGE ; Note ABS attribute!
                     0000   185
000001A4             0000   186         .BLKB   UETUNT$C_DEVDEP          ; Skip over standard UETUNT block
                     01A4   187
                     01A4   188 DISK_W_UNIT:                           ; This unit number
000001A6             01A4   189         .BLKW   1
                     01A6   190
                     01A6   191 DISK_B_FILE:                           ; Number of this file on this unit
000001A7             01A6   192         .BLKB   1
                     01A7   193
                     01A7   194 DISK_L_OTHER_PTR:                      ; Ptr to other file node on this unit
000001AB             01A7   195         .BLKL   1
                     01AB   196
                     01AB   197 DISK_L_TOP_PCT:                        ; Maximum space we'll use on this unit
000001AF             01AB   198         .BLKL   1
                     01AF   199
                     01AF   200 DISK_L_SIZE:                           ; Cur. size of this file on this unit
000001B3             01AF   201         .BLKL   1
                     01B3   202
                     01B3   203 DISK_Q_FILEDESC:                       ; ASCID desc for the filespec...
000001BB             01B3   204         .BLKQ   1                      ; ...actually used to create the file
                     01BB   205
                     01BB   206 DISK_C_NAM:                            ; NAM block for this file on this unit
0000021B             01BB   207         .BLKB   NAM$C_BLN
                     021B   208
                     021B   209 DISK_T_NAMFILSPC:                      ; Resultant file spec per file per unit
0000031A             021B   210         .BLKB   NAM$C_MAXRSS
                     031A   211
00000176             031A   212         DEVDEP_SIZE = .-UETUNT$C_DEVDEP ; Device dependent part size of node
                     031A   213                                         ; Note that this excludes buffers
```

```
                031A   214
                031A   215          .ALIGN  LONG                    ; Prevent DDP data lates on 11/750 UDA
                031C   216 DISK_K_BUFFER:                           ; I/O buffer for both reads and writes
00000B1C        031C   217          .BLKB   WRITE_SIZE
                0B1C   218
                0B1C   219          PAGES = <<UETUNT$C_INDSIZ+-     ; Add together all of the pieces...
                0B1C   220                    DEVDEP_SIZE+-         ; ...which make up a UETP unit block...
00000006        0B1C   221                    WRITE_SIZE+511>/512>  ; ...to give to the $EXPREG service
```

```
                                    0B1C   223          .SBTTL  Read-Only Data
                                00000000   224          .PSECT  RODATA,NOEXE,NOWRT,PAGE
                                    0000   225
                                    0000   226 ACNT_NAME:                                ; Process name on exit
53 45 54 53 59 53 00000008'010E0000'0000   227          .ASCID  /SYSTEST/
                                      54   000E
                                    000F   228
                                    000F   229 TEST_NAME:                                ; This test name
53 49 44 54 45 55 00000017'010E0000'000F   230          .ASCID  /UETDISK00/
                                   30 30 4B  001D
                                    0020   231
                                    0020   232 SUPDEV_GBLSEC:                            ; How we access UETSUPDEV.DAT
50 55 53 54 45 55 00000028'010E0000'0020   233          .ASCID  /UETSUPDEV/
                                   56 45 44  002E
                                    0031   234
                                    0031   235 CONTROLLER:                               ; Logical name of controller
41 4E 4C 52 54 43 00000039'010E0000'0031   236          .ASCID  /CTRLNAME/               ;
                                   45 4D  003F
                                    0041   237
                                    0041   238 MODE:                                    ; Run mode logical name
      45 44 4F 4D 00000049'010E0000'0041   239          .ASCID  /MODE/
                                    004D   240
                                    004D   241 NO_RMS_AST_TABLE:                        ; List of errors for which...
                         00000000'  004D   242          .LONG   RMS$_BLN                ; ...RMS cannot deliver an AST...
                         00000000'  0051   243          .LONG   RMS$_BUSY               ; ...even if one has an ERR= arg
                         00000000'  0055   244          .LONG   RMS$_CDA                ; Note that we can search table...
                         00000000'  0059   245          .LONG   RMS$_FAB                ; ...via MATCHC since <31:16>...
                         00000000'  005D   246          .LONG   RMS$_RAB                ; ...pattern can't be in <15:0>
                         00000014   0061   247 NRAT_LENGTH = .-NO_RMS_AST_TABLE
                                    0061   248
                                    0061   249 SYS$INPUT:                               ; Name of device from which...
4E 49 24 53 59 53 00000069'010E0000'0061   250          .ASCID  /SYS$INPUT/             ; ...the test can be aborted
                                   54 55 50  006F
                                    0072   251
                                    0072   252 INPUT_ITMLST:                           ; $GETDVI arg list for SYS$INPUT
                               0020 0040   0072   253          .WORD   64,DVI$_DEVNAM          ; We need the equivalence name
                     0000000C'00000014'    0076   254          .LONG   BUFFER,BUFFER_PTR
                         00000000          007E   255          .LONG   0                       ; Terminate the list
                                    0082   256
                                    0082   257 CS1:                                   ; Device class and type control string
21 20 42 58 32 21 0000008A'010E0000'0082   258          .ASCID  /!2XB !2XB /
                               20 42 58 32  0090
                                    0094   259
                                    0094   260 CS3:                                   ; Device class-only control string
2A 20 42 58 32 21 0000009C'010E0000'0094   261          .ASCID  /!2XB **/
                                      2A   00A2
                                    00A3   262
                                    00A3   263 CNTRLCMSG:
65 74 72 6F 62 41 000000AB'010E0000'00A3   264          .ASCID  \Aborted via a user CTRL/C\
72 65 73 75 20 61 20 61 69 76 20 64  00B1
                  43 2F 4C 52 54 43 20  00BD
                                    00C4   265
                                    00C4   266 NO_CTRLNAME:
6E 6F 63 20 6F 4E 000000CC'010E0000'00C4   267          .ASCID  /No controller specified./
63 65 70 73 20 72 65 6C 6C 6F 72 74  00D2
                  2E 64 65 69 66 69  00DE
                                    00E4   268
```

```
                                                00E4   269 DEAD_CTRLNAME:
20 74 27 6E 61 43 000000EC'010E0000'            00E4   270         .ASCID  /Can't test controller !AS, marked as unusable in UETINIDEV.DAT./
6C 6F 72 74 6E 6F 63 20 74 73 65 74             00F2
72 61 6D 20 2C 53 41 21 20 72 65 6C             00FE
61 73 75 6E 75 20 73 61 20 64 65 6B             010A
4E 49 54 45 55 20 6E 69 20 65 6C 62             0116
         2E 54 41 44 2E 56 45 44 49             0122
                                                012B   271
                                                012B   272 NOUNIT_SELECTED:
69 6E 75 20 6F 4E 00000133'010E0000'            012B   273         .ASCID  /No units selected for testing./
20 64 65 74 63 65 6C 65 73 20 73 74             0139
2E 67 6E 69 74 73 65 74 20 72 6F 66             0145
                                                0151   274
                                                0151   275 ILLEGAL_REC:
61 67 65 6C 6C 49 00000159'010E0000'            0151   276         .ASCID  /Illegal record format in file UETINIDEV.DAT!/
72 6F 66 20 64 72 6F 63 65 72 20 6C             015F
20 65 6C 69 66 20 6E 69 20 74 61 6D             016B
41 44 2E 56 45 44 49 4E 49 54 45 55             0177
                              21 54             0183
                                                0185   277
                                                0185   278 PASS_MSG:
66 6F 20 64 6E 45 0000018D'010E0000'            0185   279         .ASCID  /End of pass !UL with !UL iterations at !%D./
69 77 20 4C 55 21 20 73 73 61 70 20             0193
61 72 65 74 69 20 4C 55 21 20 68 74             019F
44 25 21 20 74 61 20 73 6E 6F 69 74             01AB
                              2E                01B7
                                                01B8   280
                                                01B8   281 INIDEV_UPDERR:                    ; Error during exit handler
20 72 6F 72 72 45 000001C0'010E0000'            01B8   282         .ASCID  /Error updating UETINIDEV.DAT./
54 45 55 20 67 6E 69 74 61 64 70 75             01C6
   2E 54 41 44 2E 56 45 44 49 4E 49             01D2
                                                01DD   283
                                                01DD   284 THREEMIN:                         ; 3 minute delta time
                  FFFFFFFF 94B62E00             01DD   285         .LONG   -10*1000*1000*180,-1
                                                01E5   286
                                                01E5   287 UNIT_DESC:                        ; Descriptor used to convert unit #
                           00000005             01E5   288         .LONG   5
                           0000001A'            01E9   289         .ADDRESS BUFFER+6
                                                01ED   290
                                                01ED   291 CONT_DESC:                        ; Descriptor used to convert controller...
                         0000 0028              01ED   292         .WORD   REC_SIZE,0         ; ...from lowercase to uppercase
                           00000014'            01F1   293         .ADDRESS BUFFER
                                                01F5   294
                                                01F5   295 FILE:                             ; Fills in RMS_ERR_STRING
      65 6C 69 66 000001FD'010E0000'            01F5   296         .ASCID  /file/
                                                0201   297
                                                0201   298 RECORD:                           ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 00000209'010E0000'            0201   299         .ASCID  /record/
                                                020F   300
                                                020F   301 RMS_ERR_STRING:                   ; Announces an RMS error
41 21 20 53 4D 52 00000217'010E0000'            020F   302         .ASCID  /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53             021D
                  44 41 21 20 65 6C 69          0229
                                                0230   303
                                                0230   304 PROMPT:
64 20 72 65 6C 6C 6F 72 74 6E 6F 43             0230   305         .ASCII  /Controller designation?: /
3A 3F 6E 6F 69 74 61 6E 67 69 73 65             023C
```

```
                                20  0248
                          00000019  0249   306          PMTSIZ = .-PROMPT
                                    0249   307
                                    0249   308 SYSTEST_DIR:                          ; Part of filespec for test files...
      5D 54 53 45 54 53 59 53 5B 00'0249   309          .ASCIC  /[SYSTEST]/          ; ...if they're on a scratch pack
                                09  0249
                                    0253   310
                                    0253   311 SYS0_SYSTEST_DIR:                     ; Part of filespec for test files...
45 54 53 59 53 2E 30 53 59 53 5B 00'0253   312          .ASCIC  /[SYS0.SYSTEST]/     ; ...if they're on a V3 system pack
                             5D 54 53  025F
                                0E  0253
                                    0262   313
                                    0262   314 DOTTST:                             ; Part of filespec for test files
                    54 53 54 2E 00' 0262   315          .ASCIC  /.TST/
                                04  0262
                                    0267   316
                                    0267   317 PATTERN:                            ; Logical name to specify pattern...
52 45 54 54 41 50 0000026F'010E0000'0267   318          .ASCID  /PATTERN/            ; ...to write to test files
                                4E  0275
```

```
                                0276   320          .SBTTL  Read/Write Data
                           00000000    321          .PSECT  RWDATA,WRT,NOEXE,PAGE
                                0000   322
                                0000   323 TTCHAN:                                ; Channel associated with ctrl. term.
                              0000  0000   324          .WORD   0
                                0002   325
                                0002   326 FLAG:                                  ; Miscellaneous flag bits
                              0000  0002   327          .WORD   0                 ; (See Equated Symbols for definitions)
                                0004   328
                                0004   329 FAO_BUF:                               ; FAO output string descriptor
                         0000 0084  0004   330          .WORD   TEXT_BUFFER,0
                          00000014' 0008   331          .ADDRESS BUFFER
                                000C   332
                                000C   333 BUFFER_PTR:                           ; Fake .ASCID buffer for misc. strings
                         0000 0084  000C   334          .WORD   TEXT_BUFFER,0    ; A word for length, a word for desc.
                          00000014' 0010   335          .ADDRESS BUFFER
                                0014   336
                                0014   337 BUFFER:                               ; FAO output and other misc. buffer
                           00000098  0014   338          .BLKB   TEXT_BUFFER
                                0098   339
                                0098   340 DEVDSC:                               ; Device name descriptor
                         0000 000A  0098   341          .WORD   MAX_DEV_DESIG,0
                          000000B7' 009C   342          .ADDRESS DEV_NAME
                                00A0   343
                                00A0   344 PROCESS_NAME:                         ; Process name
      4B 53 49 44 000000A8'010E0000' 00A0   345          .ASCID  /DISK/
                           0000000B  00AC   346          PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
                           000000B7  00AC   347          .BLKB   PROCESS_NAME_FREE
                                00B7   348
                                00B7   349 DEV_NAME:                             ; Device name buffer
                           000000C6  00B7   350          .BLKB   MAX_DEV_DESIG+MAX_UNIT_DESIG
                           0000000F  00C6   351          NAME_LEN = .-DEV_NAME
                                00C6   352
                                00C6   353 DIB:                                  ; Device Information Block
                         0000 0074  00C6   354          .WORD   DIB$K_LENGTH,0
                          000000CE' 00CA   355          .ADDRESS DIBBUF
                                00CE   356 DIBBUF:
                           00000142  00CE   357          .BLKB   DIB$K_LENGTH
                                0142   358
                                0142   359 ERROR_COUNT:                           ; Cumulative error count at runtime
                           00000000  0142   360          .LONG   0
                                0146   361
                                0146   362 STATUS:                                ; Status value on program exit
                           00000000  0146   363          .LONG   0
                                014A   364
                                014A   365 QUAD_STATUS:                          ; IO status block for misc sys. svcs.
                  00000000 00000000  014A   366          .QUAD   0
                                0152   367
                                0152   368 INADDRESS:                           ; $CRMPSC address storage
                  00000000 00000000  0152   369          .LONG   0,0
                                015A   370 OUTADDRESS:
                  00000000 00000000  015A   371          .LONG   0,0
                                0162   372
                                0162   373 UNIT_NUMBER:                          ; Current dev unit number
                               C000  0162   374          .WORD   0
                                0164   375
                                0164   376 DEVNAM_LEN:                          ; Current device name length
```

```
                0000  0164   377          .WORD   0
                      0166   378
                      0166   379 ; RANDOM1 and RANDOM2 may be combined to produce a set of pseudo-random numbers
                      0166   380 RANDOM1:                                       ; Random word #1
            AAAAAAAA  0166   381          .LONG   ^XAAAAAAAA
                      016A   382
                      016A   383 RANDOM2:                                       ; Random word #2
            A72EA72E  016A   384          .LONG   ^XA72EA72E
                      016E   385
                      016E   386 ITERATION:                                     ; # of times all tests were executed
            00000000  016E   387          .LONG   0
                      0172   388
                      0172   389 PASS:                                          ; Pass count
            00000000  0172   390          .LONG   0
                      0176   391
                      0176   392 MSG_BLOCK:                                     ; Auxiliary $GETMSG info
            0000017A  0176   393          .BLKB   4
                      017A   394
                      017A   395 EXIT_DESC:                                     ; Exit handler descriptor
            00000000  017A   396          .LONG   0
            00000C1C' 017E   397          .ADDRESS EXIT_HANDLER
            00000001  0182   398          .LONG   1
            00000146' 0186   399          .ADDRESS STATUS
                      018A   400
                      018A   401 ARG_COUNT:                                     ; Argument counter used by ERROR_EXIT
            00000000  018A   402          .LONG   0
                      018E   403
                      018E   404          .ALIGN QUAD                           ; For self-relative queue of unit blocks
                      0190   405
                      0190   406 UNIT_LIST:                                     ; Head of unit block circular list
00000000 00000000     0190   407          .QUAD   0
                      0198   408
                      0198   409 NEW_NODE:                                      ; Newly acquired node address
00000000 00000000     0198   410          .QUAD   0
                      01A0   411
                      01A0   412 RANDOM_DATA:                                   ; Buffer to fill with "random" data
            000009A0  01A0   413          .BLKL   <WRITE_SIZE+3>/4              ; Ensure it's a multiple of longwords
                      09A0   414
                      09A0   415 PATTERN_DATA:                                  ; Holds pattern to write to test...
            000009A4  09A0   416          .BLKL   1                             ; ...files if PATTERN is defined
```

```
                        09A4   418          .SBTTL  RMS-32 Data Structures
                        09A4   419          .ALIGN  LONG
                        09A4   420
                        09A4   421 SYSIN_FAB:                                   ; Allocate FAB for SYS$INPUT
                        09A4   422          $FAB-
                        09A4   423          FNM = <SYS$INPUT>
                        09F4   424
                        09F4   425 SYSIN_RAB:                                   ; Allocate RAB for SYS$INPUT
                        09F4   426          $RAB-
                        09F4   427          FAB = SYSIN_FAB,-
                        09F4   428          ROP = PMT,-
                        09F4   429          PBF = PROMPT,-
                        09F4   430          PSZ = PMTSIZ,-
                        09F4   431          UBF = DEV_NAME,-
                        09F4   432          USZ = NAME_LEN
                        0A38   433
                        0A38   434 INI_FAB:                                     ; Allocate FAB for UETINIDEV
                        0A38   435          $FAB-
                        0A38   436          FAC = <GET,PUT,UPD>,-
                        0A38   437          RAT = CR,-
                        0A38   438          SHR = <GET,PUT,UPI>,-
                        0A38   439          FNM = <UETINIDEV.DAT>
                        0A88   440
                        0A88   441 INI_RAB:                                     ; Allocate RAB for UETINIDEV
                        0A88   442          $RAB-
                        0A88   443          FAB = INI_FAB,-
                        0A88   444          RBF = BUFFER,-
                        0A88   445          UBF = BUFFER,-
                        0A88   446          USZ = REC_SIZE
                        0ACC   447
                        0ACC   448 DDB_RFA:                                     ; RFA storage for INI_RAB
00000AD2                0ACC   449          .BLKB   6
                        0AD2   450
                        0AD2   451          .ALIGN  LONG
                        0AD4   452 SUP_FAB:                                     ; Allocate FAB for UETSUPDEV
                        0AD4   453          $FAB-
                        0AD4   454          FAC = GET,-
                        0AD4   455          SHR = <UPI,GET>,-
                        0AD4   456          RAT = CR,-
                        0AD4   457          FOP = UFO,-
                        0AD4   458          FNM = <UETSUPDEV.DAT>
                        0B24   459 ;
                        0B24   460 ; Dummy FAB and RAB to copy to the UETP unit blocks
                        0B24   461 ; The following FAB and RAB must be contiguous and in this order!
                        0B24   462 ;
                        0B24   463
                        0B24   464 DUMMY_FAB:
                        0B24   465          $FAB-
                        0B24   466          CTX = DISK_K_PATTERN,-              ; This will identify a test file
                        0B24   467          FAC = <BIO,POT,GET>,-
                        0B24   468 ;        FOP = <EDL>,-
                        0B24   469          MRS = WRITE_SIZE,-
                        0B24   470          ORG = SEQ,-
                        0B24   471          RFM = UDF
                        0B74   472
                        0B74   473 DUMMY_RAB:
                        0B74   474          $RAB-
```

```
0B74    475          ROP = <ASY,BIO>,-                  ; RAB$M_ASY cleared in one-shot mode
0B74    476          RSZ = WRITE_SIZE,-
0B74    477          USZ = READ_SIZE
0BB8    478
0BB8    479 DUMMY_NAM:                                  ; More specifically identify test file
0BB8    480          $NAM-
0BB8    481          RSS = NAM$C_MAXRSS
```

```
                     0C18   483          .SBTTL  Test and Device Initialization
                 00000000   484          .PSECT  DISK,EXE,NOWRT,PAGE
                     0000   485
                     0000   486          .DEFAULT DISPLACEMENT,WORD
                     0000   487
                     0000   488 ;+
                     0000   489 ;        Start up the disk test.  This entails some overhead necessary to cope
                     0000   490 ;        with both expected and unforseen conditions, figuring out just what
                     0000   491 ;        devices are to be tested, making sure we can test the indicated devices
                     0000   492 ;        and setting up writeable space for each device to be tested.
                     0000   493 ;-
                     0000   494
                0000 0000   495 .ENTRY  UETDISK00,^M<>                  ; Entry mask
                     0002   496
      6D 09A5'CF     DE 0002   497          MOVAL   SSERROR,(FP)            ; Declare exception handler
                     0007   498          $SETSFM_S ENBFLG = #1           ; Enable system service failure mode
                     0010   499          $DCLEXH_S DESBLK = EXIT_DESC    ; Declare an exit handler
                     001B   500
                     001B   501          $OPEN   FAB = SYSIN_FAB,-       ; Open SYS$INPUT
                     001B   502                  ERR = RMS_ERROR
                     002A   503          $CONNECT RAB = SYSIN_RAB,-      ; Connect RAB to SYS$INPUT
                     002A   504                   ERR = RMS_ERROR
           02        E1 0039   505          BBC     S^#DEV$V_TRM,-          ; BR if SYS$INPUT is NOT a terminal
      1E 09E4'CF          003B   506                  SYSIN_FAB+FAB$L_DEV,10$
                     003F   507          $TRNLOG_S LOGNAM = CONTROLLER,- ; Allow terminal user to specify...
                     003F   508                    RSLLEN = DEVNAM_LEN,- ; ...a logical name...
                     003F   509                    RSLBUF = DEVDSC       ; ...for the controller to test
         01    50    D1 0058   510          CMPL    R0,#SS$_NORMAL          ; Was a controller specified?
               2E    13 005B   511          BEQL    PROC_CONT_NAME          ; BR if it was - go process it
                     005D   512 10$:
                     005D   513          $GET    RAB = SYSIN_RAB,-       ; Read SYS$INPUT...
                     005D   514                  ERR = RMS_ERROR         ; ...for controller name
            CA16'CF  B0 006C   515          MOVW    SYSIN_RAB+RAB$W_RSZ,-   ; Save the name length
            0164'CF          0070   516                  DEVNAM_LEN
                  16    12 0073   517          BNEQ    PROC_CONT_NAME          ; BR if we got something
         0146'CF  14    D0 0075   518          MOVL    #SS$_BADPARAM,STATUS    ; Save an exit status if not
            00C4'CF  DF 007A   519          PUSHAL  NO_CTRLNAME             ; Prepare for message...
                  01    DD 007E   520          PUSHL   #1                      ; ...
          00741132  8F    DD 0080   521          PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...
                  03    DD 0086   522          PUSHL   #3                      ; ...
                0B11    31 0088   523          BRW     ERROR_EXIT              ; ...to tell of bad setup
                     008B   524
                     008B   525 PROC_CONT_NAME:
0098'CF      0164'CF    3C 008B   526          MOVZWL  DEVNAM_LEN,DEVDSC       ; Set the device name length
            0098'CF  DF 0092   527          PUSHAL  DEVDSC                  ; Make sure...
            0098'CF  DF 0096   528          PUSHAL  DEVDSC                  ; ...that the specified controller...
 00000000'GF    02    FB 009A   529          CALLS   #2,G^STR$UPCASE         ; ...is all uppercase for later comparison
52   0098'CF    01    C1 00A1   530          ADDL3   #1,DEVDSC,R2            ; Estimate the eventual...
     00A0'CF    52    A0 00A7   531          ADDW2   R2,PROCESS_NAME         ; ...process name length (incl. "_")
                      DE 00AC   532          MOVAL   PROCESS_NAME+8-         ; Locate first available byte...
                     00AD   533                  +MAX_PROC_NAME-         ; ...in process name handle...
      50    00AC'CF          00AD   534                  -PROCESS_NAME_FREE,R0   ; ...for device name
                  0B    C3 00B1   535          SUBL3   #PROCESS_NAME_FREE,-    ; Will the device name fit...
               51    52          00B3   536                  R2,R1           ; ...in the remaining space?
                  08    15 00B5   537          BLEQ    10$                     ; BR if it will
               50    51    C2 00B7   538          SUBL2   R1,R0                   ; Overwrite handle otherwise...
     00A0'CF    0F    B0 00BA   539          MOVW    #MAX_PROC_NAME,PROCESS_NAME ; ...and define the maximum length
```

```
                                00BF    540 10$:
                80   5F 8F     90  00BF    541         MOVB    #^A/ /,(R0)+            ; Separate handle from device name
        60   00B7'CF   0098'CF   28  00C3    542         MOVC3   DEVDSC,DEV_NAME,(R0)    ; Concatenate handle with device name
                             7E D4  00CB    543         CLRL    -(SP)                   ; Set the time stamp flag
                      000F'CF   DF  00CD    544         PUSHAL  TEST_NAME               ; Set the test name
                             02 DD  00D1    545         PUSHL   #2                      ; Push the argument count
                    00741039 8F DD  00D3    546         PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
                00000000'GF  04 FB  00D9    547         CALLS   #4,G^LIB$SIGNAL         ; Print the startup message
                    0002'CF  08 A8  00E0    548         BISW2   #BEGIN_MSGM,FLAG        ; Set flag so we don't print it again
                                    00E5    549         $SETPRN_S PRCNAM = PROCESS_NAME ; Set the process name to UETDISK00_x
                                    00F0    550
                             02 E1  00F0    551         BBC     S^#DEV$V_TRM,-          ; BR if SYS$INPUT is NOT a terminal
                        66 09E4'CF  00F2    552                 SYSIN_FAB+FAB$L_DEV,20$
                                    00F6    553         $GETDVI_S DEVNAM = SYS$INPUT,-      ; Get the name of...
                                    00F6    554                   EFN    = #SS_SYNCH_EFN,- ; ...device which may abort test
                                    00F6    555                   ITMLST = INPUT_ITMLST,-
                                    00F6    556                   IOSB   = QUAD_STATUS
                        45 014A'CF  E9  0112    557         BLBC    QUAD_STATUS,20$         ; Avoid CTRL/C handler if any error
                                    0117    558         $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handling
                                    0117    559                   CHAN   = TTCHAN
                                    0128    560         $QIOW_S CHAN     = TTCHAN,-     ; Enable CTRL/C AST's...
                                    0128    561                 FUNC     = #IO$_SETMODE!IO$M_CTRLCAST,-
                                    0128    562                 P1       = CCASTHAND
                      00A0'CF   DF  0149    563         PUSHAL  PROCESS_NAME            ; ...and tell the user...
                             01 DD  014D    564         PUSHL   #1                      ; ...
                    0074832B 8F DD  014F    565         PUSHL   #UETP$_ABORTC!STS$K_SUCCESS ; ...how to abort gracefully...
                00000000'GF  03 FB  0155    566         CALLS   #3,G^LIB$SIGNAL         ; ...
                                    015C    567 20$:
                                    015C    568         $TRNLOG_S LOGNAM = MODE,-       ; Get the run mode
                                    015C    569                   RSLLEN = BUFFER_PTR,-
                                    015C    570                   RSLBUF = FAO_BUF
                    0014'CF   20 8A 0175    571         BICB2   #LC_BITM,BUFFER         ; Convert to upper case
                0014'CF   4F 8F 91  017A    572         CMPB    #^A/O/,BUFFER           ; Is this a one shot?
                             0A 12  0180    573         BNEQ    25$                     ; BR if not
                    0002'CF  10 A8  0182    574         BISW2   #ONE_SHOTM,FLAG         ; Set flag for one-shot mode...
                    0B78'CF  01 CA  0187    575         BICL2   #RAB$M_ASY,DUMMY_RAB+RAB$L_ROP ; ...and clear asynchronous flag
                                    018C    576 25$:
                                    018C    577         $TRNLOG_S LOGNAM = PATTERN,-    ; Do we have a specific pattern...
                                    018C    578                   RSLLEN = BUFFER_PTR,- ; ...to write to all buffers?
                                    018C    579                   RSLBUF = FAO_BUF
                          17 50 E9  01A5    580         BLBC    R0,27$                  ; BR if we don't
                        09A0'CF  DF 01A8    581         PUSHAL  PATTERN_DATA            ; We may...
                        000C'CF  DF 01AC    582         PUSHAL  BUFFER_PTR              ; ...
                00000000'GF  02 FB  01B0    583         CALLS   #2,G^OTS$CVT_TZ_L       ; ...convert ASCII pattern to binary
                          05 50 E9  01B7    584         BLBC    R0,27$                  ; BR if there's an error
                    0002'CF  20 A8  01BA    585         BISW2   #PATTERNM,FLAG          ; Indicate the presence of a pattern
                                    01BF    586 27$:
```

```
                                01BF   588 ;
                                01BF   589 ; From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller
                                01BF   590 ; and unit configuration and lets us know if the setup to run this test was
                                01BF   591 ; done correctly.  Allocate space for each file on a unit.
                                01BF   592 ;
                                01BF   593          $OPEN   FAB = INI_FAB,-          ; Open file 'UETINIDEV.DAT"
                                01BF   594                  ERR = RMS_ERROR
                                01CE   595          $CONNECT RAB = INI_RAB,-         ; Connect the RAB and FAB
                                01CE   596                   ERR = RMS_ERROR
                                01DD   597          $MGBLSC_S  INADR = INADDRESS,-   ; Connect to UETSUPDEV global section
                                01DD   598                     RETADR = OUTADDRESS,-
                                01DD   599                     GSDNAM = SUPDEV_GBLSEC,-
                                01DD   600                      FLAGS = #SEC$M_EXPREG
        00000978 8F    50   D1  01FC   601          CMPL    R0,#SS$_NOSUCHSEC        ; Was the section already there?
                       37   12  0203   602          BNEQ    30$                      ; BR if it was...
                                0205   603          $OPEN   FAB = SUP_FAB,-          ; ...else open "UETSUPDEV.DAT"
                                0205   604                  ERR = RMS_ERROR
                                0214   605          $CRMPSC_S CHAN = SUP_FAB+FAB$L_STV,- ; Create the global section
                                0214   606                    INADR = INADDRESS,-
                                0214   607                   RETADR = OUTADDRESS,-
                                0214   608                   GSDNAM = SUPDEV_GBLSEC,-
                                0214   609                    FLAGS = #SEC$M_EXPREG!SEC$M_GBL
                                023C   610 30$:
    56  015E'CF    015A'CF     C3  023C   611          SUBL3   OUTADDRESS,OUTADDRESS+4,R6 ; Compute global section length
                                0244   612
                                0244   613 FIND_IT:
                                0244   614          $GET    RAB = INI_RAB,-          ; Get the first record
                                0244   615                  ERR = RMS_ERROR
                   01ED'CF      DF  0253   616          PUSHAL  CONT_DESC                ; Make sure...
                   01ED'CF      DF  0257   617          PUSHAL  CONT_DESC                ; ...that the controller name...
           00000000'GF    02   FB  025B   618          CALLS   #2,G^STR$UPCASE          ; ...is all uppercase letters
            0014'CF    44 8F   91  0262   619          CMPB    #^A/D/,BUFFER            ; Is this a DDB?
                       27   13  0268   620          BEQL    10$                      ; Go on if not
            0014'CF    45 8F   91  026A   621          CMPB    #^A/E/,BUFFER            ; Is this the end of the file?
                       D2   12  0270   622          BNEQ    FIND_IT                  ; Continue on if not
                   0098'CF      DF  0272   623          PUSHAL  DEVDSC                   ; Push device not supported message
                   00A0'CF      DF  0276   624          PUSHAL  PROCESS_NAME             ; Parameters on the stack
                        02      DD  027A   625          PUSHL   #2
                 00748333 8F    DD  027C   626          PUSHL   #UETP$_DENOSU
                        02      F0  0282   627          INSV    #STS$K_ERROR,-           ; Set the severity code...
                        00          0284   628                  #STS$V_SEVERITY,-
                      6E    03    0285   629                  #STS$S_SEVERITY,(SP)
                0146'CF     6E  D0  0287   630          MOVL    (SP),STATUS              ; ...and save it as the exit status
                        04      DD  028C   631          PUSHL   #4
                      090B      31  028E   632          BRW     ERROR_EXIT               ; Exit in error
                                0291   633 10$:
00B7'CF   001A'CF   0164'CF     29  0291   634          CMPC    DEVNAM_LEN,BUFFER+6,DEV_NAME ; Is this the right controller?
                        A7      12  029B   635          BNEQ    FIND_IT                  ; BR if not
     0ACC'CF   0A98'CF      06  28  029D   636          MOVC3   #6,INI_RAB+RAB$W_RFA,DDB_RFA ; Save the record file address
            0018'CF    54 8F   91  02A5   637          CMPB    #^A/T/,BUFFER+4          ; Can we test this controller?
                       2F   13  02AB   638          BEQL    FOUND_IT                 ; BR if we can...
                                02AD   639          $FAO_S  CTRSTR = DEAD_CTRLNAME,- ; ...and yell at user if we can't
                                02AD   640                  OUTLEN = BUFFER_PTR,-
                                02AD   641                  OUTBUF = FAO_BUF,-
                                02AD   642                  P1     = #DEVDSC
                0146'CF     14  D0  02C6   643          MOVL    #SS$_BADPARAM,STATUS     ; Set return status
                     000C'CF    DF  02CB   644          PUSHAL  BUFFER_PTR               ; ...
```

```
                    01   DD  02CF   645          PUSHL   #1                              ; ...
          00741132 8F    DD  02D1   646          PUSHL   #UETP$_TEXT!STS$K_cRROR         ; ...
                    03   DD  02D7   647          PUSHL   #3                              ; ...
                  08C0   31  02D9   648          BRW     ERROR_EXIT                      ; We can't test what we can't test
                         02DC   649
                         02DC   650 FOUND_IT:
                         02DC   651          $GET    RAB = INI_RAB,-                 ; Get a record
                         02DC   652                  ERR = RMS_ERROR
                01ED'CF  DF  02EB   653          PUSHAL  CONT_DESC                       ; Make sure...
                01ED'CF  DF  02EF   654          PUSHAL  CONT_DESC                       ; ...that this line...
00000000'GF       02     FB  02F3   655          CALLS   #2,G^STR$UPCASE                 ; ...is all uppercase letters
     0014'CF   55 8F     91  02FA   656          CMPB    #^A/U/,BUFFER                   ; Is this a UCB?
                 24     13  0300   657          BEQL    30$                             ; BR if it is
     0014'CF   44 8F     91  0302   658          CMPB    #^A/D/,BUFFER                   ; Is this a DDB?
                 19     13  0308   659          BEQL    20$                             ; BR if yes
     0014'CF   45 8F     91  030A   660          CMPB    #^A/E/,BUFFER                   ; Is this the end?
                 11     13  0310   661          BEQL    20$                             ; BR if yes
                         0312   662 10$:
                0151'CF  DF  0312   663          PUSHAL  ILLEGAL_REC                     ; Then this is an error in the record
                    01   DD  0316   664          PUSHL   #1                              ; Push the error message
          00741132 8F    DD  0318   665          PUSHL   #UETP$_TEXT!STS$K_ERROR         ; Push the signal name
                    03   DD  031E   666          PUSHL   #3                              ; Push the temp arg count
                  0879   31  0320   667          BRW     ERROR_EXIT                      ; Finish for good
                         0323   668 20$:
                  00EE   31  0323   669          BRW     ALL_SET                         ; Found DDB or END
                         0326   670 30$:
     0018'CF   54 8F     91  0326   671          CMPB    #^A/T/,BUFFER+4                 ; Is the unit testable?
                 AE     12  032C   672          BNEQ    FOUND_IT                        ; BR if not
                 01     DD  032E   673          PUSHL   #1                              ; Flag to ignore blanks when converting
                 02     DD  0330   674          PUSHL   #2                              ; Set byte size of results
               0162'CF  DF  0332   675          PUSHAL  UNIT_NUMBER                     ; Set address to receive word
               01E5'CF  DF  0336   676          PUSHAL  UNIT_DESC                       ; Push string address
     00000000'GF    04    FB  033A   677          CALLS   #4,G^OTS$CVT_TI_L               ; Convert ASCII unit # to decimal
                   CE 50   E9  0341   678          BLBC    R0,10$                          ; Don't allow a bogus unit to pass
               05    20    3B  0344   679          SKPC    #^A/ /,#MAX_UNIT_DESIG,-        ; Find out where unit number really is
               001A'CF            0347   680                  BUFFER+6
                     50    D7  034A   681          DECL    R0                              ; Units must all be at least one digit
            61 50    30    3B  034C   682          SKPC    #^A/0/,R0,(R1)                  ; Skip leading zeroes on the unit
                     50    D6  0350   683          INCL    R0                              ; Compensate for DECL above
     0098'CF 0164'CF  50    A1  0352   684          ADDW3   R0,DEVNAM_LEN,DEVDSC            ; Calculate device'unit string length
            52 0164'CF  3C    3C  035A   685          MOVZWL  DEVNAM_LEN,R2                   ; Offset to unit number in DEVDSC
        00B7'C2 61    50    28  035F   686          MOVC3   R0,(R1),DEV_NAME(R2)            ; Append unit number to device
                         0365   687          $GETDEV_S DEVNAM = DEVDSC,-             ; Get the device characteristics
                         0365   688                    PRIBUF = DIB
            57 00D2'CF  9A    37A   689          MOVZBL  DIBBUF+DIB$B_DEVCLASS,R7 ; Save the device class
            58 00D3'CF  9A    37F   690          MOVZBL  DIBBUF+DIB$B_DEVTYPE,R8 ; Save the device type
                         0384   691          $FAO_S  CTRSTR = CS1,-
                         0384   692                  OUTBUF = FAO_BUF,-
                         0384   693                  P1     = R7,-
                         0384   694                  P2     = R8                     ; Make it into a string
015A'DF  56  0014 CF  06  39  0399   695          MATCHC  #6,BUFFER,R6,@OUTADDRESS ; Find the device class and type
                    1E 13  03A2   696          BEQL    40$                             ; BR if it was found
                         03A4   697          $FAO_S  CTRSTR = CS3,-                  ; Try for full class support
                         03A4   698                  OUTBUF = FAO_BUF,-
                         03A4   699                      P1 = R7
015A'DF  56  0014'CF  06  39  03B7   700          MATCHC  #6,BUFFER,R6,@OUTADDRESS ; Find the device class only
                    0D 12  03C0   701          BNEQ    50$                             ; BR if not found
```

```
                         03C2   702 40$:
      55  000F'CF    9A  03C2   703          MOVZBL  TEST_NAME,R5             ; Get the test name length
0017'CF   63    55   29  03C7   704          CMPC3   R5,(R3),TEST_NAME+8     ; Are we the right test?
                1F   13  03CD   705          BEQL    60$                     ; BR if yes
                         03CF   706 50$:
          0098'CF    DF  03CF   707          PUSHAL  DEVDSC                  ; Push device not supported message
          00A0'CF    DF  03D3   708          PUSHAL  PROCESS_NAME            ; Parameters on the stack
                02   DD  03D7   709          PUSHL   #2                      ; Push the argument count
       00748333 8F   DD  03D9   710          PUSHL   #UETP$_DENOSU
                02   F0  03DF   711          INSV    #STS$K_ERROR,-
                00       03E1   712                  #STS$V_SEVERITY,-
          6E    03       03E2   713                  #STS$S_SEVERITY,(SP)    ; Set the severity code...
    0146'CF     6E   D0  03E4   714          MOVL    (SP),STATUS             ; ...and save it as the exit status
                04   DD  03E9   715          PUSHL   #4                      ; Push the partial arg count...
              07AE   31  03EB   716          BRW     ERROR_EXIT              ; ...and split this scene
                         03EE   717 60$:
                00   DD  03EE   718          PUSHL   #0                      ; Allocate and initialize...
    060B'CF     01   FB  03F0   719          CALLS   #1,UNIT_FILE_SETUP      ; ...space for file 0 on this unit
16  0002'CF     04   E0  03F5   720          BBS     #ONE_SHOT_V,FLAG,70$    ; Omit file 1 if one-shot mode
                50   DD  03FB   721          PUSHL   R0                      ; Save pointer so file 1 can point here
                01   DD  03FD   722          PUSHL   #1                      ; Allocate and initialize...
    060B'CF     01   FB  03FF   723          CALLS   #1,UNIT_FILE_SETUP      ; ...space for file 1 on this unit
                51 8ED0  0404   724          POPL    R1                      ; Have file 0...
    01A7 C1     50   D0  0407   725          MOVL    R0,DISK_L_OTHER_PTR(R1) ; ...point to file 1...
    01A7 C0     51   D0  040C   726          MOVL    R1,DISK_L_OTHER_PTR(R0) ; ...and file 1 point to file 0
                         0411   727 70$:
              FEC8   31  0411   728          BRW     FOUND_IT                ; Do the next UCB
```

```
                     0414   730 ;
                     0414   731 ; Arrive here when we have the device configuration.  In normal or loop forever
                     0414   732 ; mode, set a timer far enough in the future such that we can do a reasonable
                     0414   733 ; set of tests before the timer expires, but if our device gets hung, the
                     0414   734 ; program won't waste too much time before noticing.  Let one-shot mode be a
                     0414   735 ; special case.
                     0414   736 ;
                     0414   737 ALL_SET:
          0190'CF    D5   0414   738         TSTL    UNIT_LIST                       ; Anything to test?
                16   12   0418   739         BNEQ    10$                             ; BR if yes
          0128'CF    DF   041A   740         PUSHAL  NOUNIT_SELECTED                 ; Else set up the error message...
                01   DD   041E   741         PUSHL   #1                              ; ...argument count...
        00741132 8F    DD   0420   742         PUSHL   #UETP$_TEXT!STS$K_ERROR         ; ...signal name...
                03   DD   0426   743         PUSHL   #3                              ; ...and parameter count
          0146'CF    14   D0   0428   744         MOVL    #SS$_BADPARAM,STATUS            ; Set return status
                076C   31   042D   745         BRW     ERROR_EXIT                      ; ...and give up, complaining
                     0430   746 10$:
          0002'CF    04   A8   0430   747         BISW2   #SAFE_TO_UPDM,FLAG              ; OK, safe to update UETINIDEV.DAT now
0098'CF   0164'CF    D0   0435   748         MOVL    DEVNAM_LEN,DEVDSC               ; DEVDSC will describe controller name
  13 0002'CF    04   E0   043C   749         BBS     #ONE_SHOTV,FLAG,RESTART         ; In one-shot mode start testing now...
                     0442   750                                                         ; ...else fall into TIME_IT
                     0442   751
                     0442   752 TIME_IT:
                     0442   753         $SETIMR_S DAYTIM = THR_MIN,-             ; Set timer AST to 3 minutes
                     0442   754                   ASTADR = TIME_OUT,-
                     0442   755                   EFN    = #EFN2
```

```
                                0455   757          .SBTTL  Test the Disk
                                0455   758
                                0455   759 RESTART:
                                0455   760 ;
                                0455   761 ; At this point the controller designation is in location DEV_NAME pointed to
                                0455   762 ; by descriptor DEVDSC.  The controller is known to be supported by this test.
                                0455   763 ;
          07 0002'CF    05      E1  0455   764          BBC     #PATTERNV,FLAG,5$       ; BR if we're allowing various patterns
       0166'CF    09A0'CF      D0  045B   765          MOVL    PATTERN_DATA,RANDOM1    ; We're forcing a specific pattern
                                0462   766 5$:
             56    01A0'CF     DE  0462   767          MOVAL   RANDOM_DATA,R6          ; Set up...
           57     00000200 8F   D0  0467   768          MOVL    #<WRITE_SIZE+3>/4,R7    ; ...a bufferful of longwords...
                                046E   769 10$:
          07 0002'CF    05      E0  046E   770          BBS     #PATTERNV,FLAG,15$      ; ...which can be either pattern...
       0166'CF    016A'CF      CC  0474   771          ADDL2   RANDOM2,RANDOM1         ; ...or pseudo_random...
                                047B   772 15$:
             86    0166'CF     D0  047B   773          MOVL    RANDOM1,(R6)+           ; ...data...
                      EB 57     F5  0480   774          SOBGTR  R7,10$                  ; ...
                                0483   775 ;
                                0483   776 ; For each unit-file node in the queue, clean up any old version of the file we
                                0483   777 ; want to create and create a new file.  The FABs are already set up such that
                                0483   778 ; each file uses PERCENTAGE blocks of the total space on the disk, the files
                                0483   779 ; names are controller'unit'x, where "x" is either "0" or "1", and the files
                                0483   780 ; will be accessed in Block I/O mode by RMS.  In normal and loop forever modes,
                                0483   781 ; the test will proceed asynchronously with the file I/O.  In one-shot mode,
                                0483   782 ; I/O will be a single, synchronous write/read per file, handled separately.
                                0483   783 ;
    56  00000190'8F    0190'CF  C1  0483   784          ADDL3   UNIT_LIST,#UNIT_LIST,R6 ; R6 will point to the current node
                                048D   785 20$:
                  0397  30      BA  048D   786          BSBW    PRE_CLEAN               ; Remove leftover files from previous run
                 0E A6   01     B0  0490   787          MOVW    #DISK_K_CREATE,UETUNT$W_FUNC(R6) ; Indicate func in case error
                         02     88  0494   788          BISB2   #UETUNT$M_TESTABLE,-    ; Assume file will be testable and...
                      0B A6         0496   789                  UETUNT$B_FLAGS(R6)
                         0C     8A  0498   790          BICB2   #DISK_M_NOCREATE!DISK_M_NOEXTEND,-
                      0B A6         049A   791                  UETUNT$B_FLAGS(R6)      ; ...$CREATE and $EXTEND succeed
                                049C   792 $CREATE FAB = UETUNT$K_FAB(R6)  ; Try to create a file on a disk unit
                  048B  30      BA  04A7   793          BSBW    CREATE_CHECK            ; Check for errors, recover from some
                         01     E1  04AA   794          BBC     #UETUNT$V_TESTABLE,-    ; Skip $CONNECT if $CREATE failed
                   3D 0B A6         04AC   795                  UETUNT$B_FLAGS(R6),40$
                  01BE C6    9A   04AF   796          MOVZBL  DISK_C_NAM+NAM$B_RSL(R6),- ; Form an ASCID desc to the...
                  01B3 C6         04B3   797                  DISK_Q_FILEDESC(R6)
                  01BF C6    D0   04B6   798          MOVL    DISK_C_NAM+NAM$L_RSA(R6),- ; ...file spec we actually used
                  01B7 C6         04BA   799                  DISK_Q_FILEDESC+4(R6)
                    0E A6    B4   04BD   800          CLRW    UETUNT$W_FUNC(R6)       ; Reset func for error handling
                  0120 C6    D0   04C0   801          MOVL    UETUNT$K_FAB+FAB$L_ALQ(R6),- ; Set current file size
                  01AF C6         04C4   802                  DISK_L_SIZE(R6)
                  0198 C6    D4   04C7   803          CLRL    UETUNT$K_RAB+RAB$L_BKT(R6) ; Initialize current block
                    10 A6    D4   04CB   804          CLRL    UETUNT$L_ITER(R6)       ; Reset iteration count if loop forever
             57   06F0'CF    DE   04CE   805          MOVAL   WRITE_NEXT,R7           ; Assume normal or loop mode asynch I/O
          02 0002'CF    04      E1  04D3   806          BBC     #ONE_SHOT,FLAG,30$      ; BR if that is indeed the case
                     57    D4   04D9   807          CLRL    R7                      ; One-shot mode, no SUC= return
                                04DB   808 30$:
                                04DB   809          $CONNECT RAB = UETUNT$K_RAB(R6),- ; Set up the file's record access
                                04DB   810                   ERR = RMS_ERROR,-
                                04DB   811                   SUC = (R7)
                                04EC   812 40$:
             56    66     C0   04EC   813          ADDL2   (R6),R6                 ; Loop through all units in the queue
```

```
00000190'8F   56   D1   04EF   814          CMPL    R6,#UNIT_LIST          ; Back at the head of the queue?
              95   12   04F6   815          BNEQ    20$                    ; BR if not
```

```
                                04F8    817 ;
                                04F8    818 ; All the $CREATEs and $CONNECTs are done.  Before we try to hibernate in
                                04F8    819 ; normal or loop forever modes, or start I/O in one-shot mode, see if any
                                04F8    820 ; files were started successfully.  R6 is still points to UNIT_LIST.
                                04F8    821 ;
                                04F8    822 50$:
                          56    66    C0  04F8    823         ADDL2   (R6),R6                 ; Point to the next possible node
                00000190'8F     56    D1  04FB    824         CMPL    R6,#UNIT_LIST           ; Back at the head of the queue?
                          03    12  0502    825         BNEQ    60$                     ; Check this node if not
                        00F0    31  0504    826         BRW     TEST_OVER               ; Nothing left to test so go home
                                    0507    827 60$:
                          01    E1  0507    828         BBC     #UETUNT$V_TESTABLE,-    ; Pass to next node if this one...
                      EC 0B A6          0509    829                 UETUNT$B_FLAGS(R6),50$  ; ...isn't testable
                                    050C    830                                         ; Otherwise start testing
                                    050C    831
                 0C 0002'CF     04    E0  050C    832         BBS     #ONE_SHOTV,FLAG,ONE_SHOT_TEST ; Synchronous I/O if one-shot mode
                                    0512    833 ;
                                    0512    834 ; We'll arrive here in normal or loop forever testing modes.
                                    0512    835 ; All testable units have their files started up.  Life proceeds asynchronously
                                    0512    836 ; as write requests are issued and the results are verified.  The AST routines
                                    0512    837 ; are responsible for that as well as file expansion.  The process terminates
                                    0512    838 ; when the $SETIMR alarm goes off and wakes us from the following...
                                    0512    839 ;
                                    0512    840         $HIBER_S                        ; File testing continues via ASTs
                                    0519    841 ;
                                    0519    842 ; We return when all testing est finis.  Clean up any messy files.
                                    0519    843 ;
                        034B    30  0519    844         BSBW    CLEAN_UP                ; This closes and erases files
                          6D    11  051C    845         BRB     SUC_EXIT                ; Go finish off this pass of the test
```

```
                                051E   847 ;
                                051E   848 ; Get here only in one-shot testing mode.
                                051E   849 ; All testable units have their files started up.  We'll do one write/verify
                                051E   850 ; pair per file.  Since everything is synchronous except error handling, all
                                051E   851 ; the code is inline.
                                051E   852 ;
                                051E   853 ONE_SHOT_TEST:
    58   00000190'8F    0190'CF   C1   051E   854          ADDL3   UNIT_LIST,#UNIT_LIST,R8 ; This will point to the current node
                                    0528   855 10$:
                 56    0160 C8   DE   0528   856          MOVAL   UETUNT$K_RAB(R8),R6     ; Point to the RAB in current node
                       01   E1   052D   857          BBC     #UETUNT$V_TESTABLE,-    ; Skip node if not testable
                    45 0B A8         052F   858                  UETUNT$B_FLAGS(R8),20$
                  38 A6    01   D0   0532   859          MOVL    #1,RAB$L_BKT(R6)        ; Pick a record to write
              28 A6    01A0'CF   DE   0536   860          MOVAL   RANDOM_DATA,RAB$L_RBF(R6) ; Indicate write buffer
                                    053C   861          $WRITE  RAB = (R6),-            ; Write a record
                                    053C   862                  ERR = RMS_ERROR
                       01   E1   0549   863          BBC     #UETUNT$V_TESTABLE,-    ; Skip node if not testable
                    29 0B A8         054B   864                  UETUNT$B_FLAGS(R8),20$
        0800 8F    00    68   00   2C   054E   865          MOVC5   #0,(R8),#0,#WRITE_SIZE,DISK_K_BUFFER(R8) ; Clear the read buffer
                       031C C8         0555
                                    0558   866          $READ   RAB = (R6),-            ; Read a record
                                    0558   867                  ERR = RMS_ERROR
           01A0'CF    0800 8F   29   0565   868          CMPC3   #READ_SIZE,RANDOM_DATA,- ; Verify a record
                       031C C8         056C   869                  DISK_K_BUFFER(R8)
                          06   13   056F   870          BEQL    20$
                       38 A6    D4   0571   871          CLRL    RAB$L_BKT(R6)           ; Indicate random data for last...
                                    0574   872                                          ; ...block read (see GET_PATTERN)
                        0331    30   0574   873          BSBW    VERIFY_ERROR            ; Yell out the error
                                    0577   874 20$:
                 58    68   C0   0577   875          ADDL2   (R8),R8                 ; Loop through all units in the queue
      00000190'8F    58   D1   057A   876          CMPL    R8,#UNIT_LIST           ; Back at the head of the queue?
                     A5   12   0581   877          BNEQ    10$                     ; BR if more nodes to test
                                    0583   878
          0002'CF    02   A8   0583   879          BISW2   #TEST_OVERM,FLAG        ; Indicate a normal test termination
                    02DC   30   0588   880          BSBW    CLEAN_UP                ; Close and erase files
                                    058B   881          ;BRW    SUC_EXIT                ; Fall into finishing off the test
```

```
                     058B   883 SUC_EXIT:
          0146'CF    D5 058B   884         TSTL    STATUS                          ; Are we trying to exit?
               6F    12 058E   885         BNEQ    TEST_OVER_STATUS                ; BR if we are
                        0590   886         $TRNLOG_S LOGNAM = MODE,-
                        0590   887                   RSLLEN = BUFFER_PTR,-
                        0590   888                   RSLBUF = FAO_BUF              ; Get the run mode
     0014'CF    20    8A 05A9   889         BICB2   #LC_BITM,BUFFER                 ; Convert to upper case
0014'CF   4C 8F    91 05AE   890         CMPB    #^A7L/,BUFFER                   ; Is this a loop for ever?
               40    12 05B4   891         BNEQ    TEST_OVER                       ; BR if not
     0002'CF    02    AA 05B6   892         BICW2   #TEST_OVERM,FLAG                ; Reset the termination flag
          0172'CF    D6 05BB   893         INCL    PASS                            ; Bump the pass count
                     05BF   894         $FAO_S  CTRSTR = PASS_MSG,-
                     05BF   895                 OUTLEN = BUFFER_PTR,-
                     05BF   896                 OUTBUF = FAO_BUF,-
                     05BF   897                 P1     = PASS,-
                     05BF   898                 P2     = ITERATION,-
                     05BF   899                 P3     = #0                     ; Make the end of pass message
          000C'CF    DF 05DC   900         PUSHAL  BUFFER_PTR                      ; Push the string desc.
               01    DD 05E0   901         PUSHL   #1                              ; Push arg count
       00741133 8F    DD 05E2   902         PUSHL   #UETP$_TEXT!STS$K_INFO          ; Push the signal name
00000000'GF    03    FB 05E8   903         CALLS   #3,G^LIB$SIGNAL                 ; Print the end of pass message
          016E'CF    D4 05EF   904         CLRL    ITERATION                       ; Reset the iteration count
             FE4B    31 05F3   905         BRW     TIME_IT                         ; Do the next pass
                     05F6   906
                     05F6   907 TEST_OVER:
0146'CF    10000001 8F    D0 05F6   908         MOVL    #SS$_NORMAL!STS$M_INHIB_MSG,STATUS ; Set successful exit status
                     0600   909 TEST_OVER_STATUS:
                     0600   910         $EXIT_S STATUS                          ; Exit with the status
```

```
                              060B          .SBTTL  Setup Routine for One File on a Unit
                              060B     ;++
                              060B     ; FUNCTIONAL DESCRIPTION:
                              060B     ;        This routine is called to allocate space and do the general
                              060B     ;        initialization for the data structure associated with a file on a unit.
                              060B     ;
                              060B     ; CALLING SEQUENCE:
                              060B     ;        PUSHL   number-of-this-file-on-this-unit
                              060B     ;        CALLS   #1,UNIT_FILE_SETUP
                              060B     ;
                              060B     ; INPUT PARAMETERS:
                              060B     ;        A byte giving the cardinal number of the current file being set up on
                              060B     ;        the current unit (4(AP)).
                              060B     ;
                              060B     ; IMPLICIT INPUTS:
                              060B     ;        DEVDSC is a descriptor to the current device and unit name in DEV_NAME.
                              060B     ;        DIBBUF is the result of a $GETDEV on that device.
                              060B     ;        UNIT_NUMBER has the current unit number.
                              060B     ;        UNIT_LIST is the queue header to which a new node will be added.
                              060B     ;        DUMMY_FAB and DUMMY_RAB are adjacent and contain templates of a FAB and
                              060B     ;        a RAB, respectively.
                              060B     ;
                              060B     ; OUTPUT PARAMETERS:
                              060B     ;        R0 will have the address of the node just created.
                              060B     ;
                              060B     ; IMPLICIT OUTPUTS:
                              060B     ;        A new node will be added to the queue headed by UNIT_LIST.
                              060B     ;
                              060B     ; COMPLETION CODES:
                              060B     ;        NONE
                              060B     ;
                              060B     ; SIDE EFFECTS:
                              060B     ;        The space just allocated will be initialized.
                              060B     ;
                              060B     ;--
                              060B     UNIT_FILE_SETUP:
                        0FFC  060B          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                              060D
                              060D          $EXPREG_S PAGCNT = #PAGES,-     ; Get a new node of demand zero memory
                              060D                    RETADR = NEW_NODE
0190'CF   0198'DF         5D  061E          INSQTI  @NEW_NODE,UNIT_LIST     ; Put the new node in the unit list
     56   0198'CF         D0  0625          MOVL    NEW_NODE,R6             ; Save a copy of its address
       08 A6   01         90  062A          MOVB    #1,UETUNT$B_TYPE(R6)    ; Set the structure type
          031A 8F         B0  062E          MOVW    #UETUNT$C_INDSIZ+DEVDEP_SIZE,-
            09 A6             0632                  UETUNT$W_SIZE(R6)       ; Set the structure size
          0094 8F         28  0634          MOVC3   #FAB$C_BLN+RAB$C_BLN,-
0110 C6   0B24'CF             0638                  DUMMY_FAB,UETUNT$C_FAB(R6) ; Save a FAB and a RAB away
     57   0110 C6         DE  063E          MOVAL   UETUNT$K_FAB(R6),R7     ; Save the FAB address
     58   0160 C6         DE  0643          MOVAL   UETUNT$K_RAB(R6),R8     ; Save the RAB address
       3C A8   57         D0  0648          MOVL    R7,RAB$L_FAB(R8)        ; Set the FAB address in the RAB
  24 A8   031C C6         DE  064C          MOVAL   DISK_K_BUFFER(R6),RAB$L_UBF(R8) ; Set read buffer address...
  28 A8   031C C6         DE  0652          MOVAL   DISK_K_BUFFER(R6),RAB$L_RBF(R8) ; ...and write buffer address
```

```
                                                        I 14
                0BB8'CF    0060 8F    28  0658   965          MOVC3   #NAM$C_BLN,DUMMY_NAM,-  ; Set up a NAM block...
                           01BB C6        065F   966                  DISK_C_NAM(R6)
                           01BB C6    DE  0662   967          MOVAL   DISK_C_NAM(R6),-        ; ...so RMS knows exactly which file...
                           0138 C6        0666   968                  UETUNT$C_FAB+FAB$L_NAM(R6)
                           021B C6    DE  0669   969          MOVAL   DISK_T_NAMFILSPC(R6),-  ; ...to delete when cleaning up...
                           01BF C6        066D   970                  DISK_C_NAM+NAM$L_RSA(R6); ...in case multiple tests are running
                     59    C13E'CF    4E  0670   971          CVTLF   DIBBOF+DIB$L_MAXBLOCK,R9 ; Use the total space on the disk...
                     5A    59         04  0675   972          MULF3   #TOP_PCT,R9,R10         ; ...to get the max we'll use...
                     59    CCCD3E4C   8F  0679   973          MULF2   #PERCENTAGE,R9          ; ...and how much each file is extended
                           01AB C6    5A  0680   974          CVTFL   R10,DISK_L_TOP_PCT(R6)  ; Store the max space for the unit...
                             10 A7    59  0685   975          CVTFL   R9,FAB$L_ALQ(R7)        ; ...and the original file size
                                        0689   976
                        2C A7    14 A6  DE  0689   977          MOVAL   UETUNT$T_FILSPC(R6),FAB$L_FNA(R7); Set addr of filspec
                     59    0098'CF    01  A1  068E   978          ADDW3   #1,DEVDSC,R9            ; Device names are device'unit':
                                        0694   979                                          ; NOTE:  VMS guarantees length is OK
                           5A    59     B0  0694   980          MOVW    R9,R10                  ; Make file names as device'unit'x,...
                           27    5A     B1  0697   981          CMPW    R10,#FN_LEN             ; ...up to a maximum length
                                 03     15  069A   982          BLEQ    10$                     ; BR if length is OK
                           5A    27     B0  069C   983          MOVW    #FN_LEN,R10             ; Otherwise force length to be OK
                                        069F   984                                          ; In this case, file number is lost...
                                        069F   985                                          ; ...but files may be distinguished...
                                        069F   986                                          ; ...by their version number
                                        069F   987 10$:
                       5B    04 AC    30  81  069F   988          ADDB3   #^A/0',4(AP),R11        ; Form final filename char (file number)
                59  3A 00B7'CF    0098'CF  2C  06A4   989          MOVC5   DEVDSC,DEV_NAME,#^A/:/,R9,UETUNT$T_FILSPC(R6) ; Store device...
                                       14 A6   06AD
                5A  5B 00B7'CF    0098'CF  2C  06AF   990          MOVC5   DEVDSC,DEV_NAME,R11,R10,(R3) ; ...filename...
                                       63      06B8
                           50    0262'CF    9B  06B9   991          MOVZBW  DOTTST,R0               ; ...
                     63    0263'CF    50  28  06BE   992          MOVC3   R0,DOTTST+1,(R3)        ; ...and extension
                       30 A7    024A'CF    DE  06C4   993          MOVAL   SYSTEST_DIR+1,FAB$L_DNA(R7) ; Set up typical default directory
                                           06CA   994
                                           06CA   995          ; Note that MOVCx thinks length is a word but RMS thinks it's a byte.
                       34 A7    5A    59   81  06CA   996          ADDB3   R9,R10,FAB$B_FNS(R7)    ; Store the file name string length...
                       34 A7    0262'CF    80  06CF   997          ADDB2   DOTTST,FAB$B_FNS(R7)    ; ...(dev:file.ext)
                       35 A7    0249'CF    90  06D5   998          MOVB    SYSTEST_DIR,FAB$B_DNS(R7) ; Store typical default directory length
                01A4 C6         0162'CF    B0  06DB   999          MOVW    UNIT_NUMBER,DISK_W_UNIT(R6) ; Save the unit number...
                  01A6 C6         04 AC    90  06E2  1000          MOVB    4(AP),DISK_B_FILE(R6)   ; ...and the file number on this unit
                                    02     88  06E8  1001          BISB2   #UETUNT$M_TESTABLE,-    ; Assume file will be testable
                                 0B A6          06EA  1002                  UETUNT$B_FLAGS(R6)
                                 50    56   D0  06EC  1003          MOVL    R6,R0                   ; Return node address as function value
                                           04  06EF  1004          RET
```

```
                                    J 14
                                    06F0  1006          .SBTTL  Successful $READ AST Routine
                                    06F0  1007 ;++
                                    06F0  1008 ; FUNCTIONAL DESCRIPTION:
                                    06F0  1009 ;       This routine will receive the AST when a $READ operation finishes for
                                    06F0  1010 ;       a file on a unit.  It verifies that the data read were correct and
                                    06F0  1011 ;       starts up the next $WRITE.  In situations detailed below, it will try
                                    06F0  1012 ;       to extend the file.
                                    06F0  1013 ;
                                    06F0  1014 ; CALLING SEQUENCE:
                                    06F0  1015 ;       Called via AST at I/O completion for all files on all units.
                                    06F0  1016 ;
                                    06F0  1017 ; INPUT PARAMETERS:
                                    06F0  1018 ;       4'AP) is RAB address
                                    06F0  1019 ;
                                    06F0  1020 ; IMPLICIT INPUTS:
                                    06F0  1021 ;       All FABs and RABs.
                                    06F0  1022 ;       An AST argument list with the ASTPRM value equal to the address of the
                                    06F0  1023 ;       associated RAB.
                                    06F0  1024 ;
                                    06F0  1025 ; OUTPUT PARAMETERS:
                                    06F0  1026 ;       NONE
                                    06F0  1027 ;
                                    06F0  1028 ; IMPLICIT OUTPUTS:
                                    06F0  1029 ;       Fields may be modified in FABs and RABs.
                                    06F0  1030 ;       If a file is being processed correctly and the test has not terminated,
                                    06F0  1031 ;       a new block of the file is written.
                                    06F0  1032 ;       A file may be extended.
                                    06F0  1033 ;
                                    06F0  1034 ; COMPLETION CODES:
                                    06F0  1035 ;       NONE
                                    06F0  1036 ;
                                    06F0  1037 ; SIDE EFFECTS:
                                    06F0  1038 ;       Other fields in the node associated with this file may be modified.
                                    06F0  1039 ;--
                                    06F0  1040 WRITE_NEXT:
                            0FFC    06F0  1041          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                    06F2  1042
6D    09A5'CF    DE                 06F2  1043          MOVAL   SSERROR,(FP)            ; Declare exception handler
  56    04 AC    D0                 06F7  1044          MOVL    4(AP),R6                ; Get the address of our RAB...
  57    3C A6    D0                 06FB  1045          MOVL    RAB$L_FAB(R6),R7        ; ...and its associated FAB
58    FEA0 C6    DE                 06FF  1046          MOVAL   -UETUNT$K_RAB(R6),R8    ; Set up a node pointer
                                    0704  1047                                          ; These registers remain constant...
                                    0704  1048                                          ; ...throughout this routine
```

```
                                0704  1050 ;
                                0704  1051 ; Verify that the data read back were those that were written out.  GET_PATTERN
                                0704  1052 ; below explains how we figure that out.  Block 0 is an exception, however.  To
                                0704  1053 ; RMS, block 0 would imply "write (or read) the next block of the file."  We
                                0704  1054 ; therefore avoid its use in choosing a next block and additionally reserve
                                0704  1055 ; it to mean that the test has just started and there is no data to verify.
                                0704  1056 ;
                      38 A6  D5 0704  1057         TSTL    RAB$L_BKT(R6)           ; If the last block accessed was 0...
                         03  12 0707  1058         BNEQ    10$                     ; ...
                       0079  31 0709  1059         BRW     DO_THE_WRITE            ; ...then we have nothing to verify; BR
                                070C  1060 10$:
              06 0002'CF   05  E0 070C  1061         BBS     #PATTERNV,FLAG,15$      ; BR if we're forcing a pattern
                    38 A6  03  D3 0712  1062         BITL    #3,RAB$L_BKT(R6)        ; Determine which pattern was used last
                       0B  12 0716  1063         BNEQ    20$                     ; BR if any bits are set
                                0718  1064 15$:
     28 B6   01A0'CF 0800 8F  29 0718  1065         CMPC3   #READ_SIZE,RANDOM_DATA,@RAB$L_RBF(R6) ; Either ^B00 or pattern
                       0C  11 0721  1066         BRB     30$                     ; Rejoin common code
                                0723  1067 20$:
                     00BF  30 0723  1068         BSBW    GET_PATTERN             ; Fill R9 with the pattern to verify
28 B6   0800 8F   59  66   00  2D 0726  1069         CMPC5   #0,(R6),R9,#READ_SIZE,@RAB$L_RBF(R6) ; Verify the buffer
                                072F  1070 30$:
                       04  13 072F  1071         BEQL    CHECK_EXTEND            ; BR if the strings were the same
                     0174  30 0731  1072         BSBW    VERIFY_ERROR            ; Yell if strings differed (bad news)
                       04  04 0734  1073         RET                             ; No more I/O to this file
                                0735  1074 ;
                                0735  1075 ; The data are verified.  See if it's time to try to extend the file.  File
                                0735  1076 ; extension is meant to test software as much as anything else.  We will
                                0735  1077 ; attempt it if there is no flag contraindicating it for this file, we've done
                                0735  1078 ; a positive multiple of BAKERS $WRITEs to the file and the total space across
                                0735  1079 ; all files on this unit doesn't exceed the value in DISK_L_TOP_PCT.
                                0735  1080 ; Most of the errors we will get from $EXTEND will be caused by exceeding disk
                                0735  1081 ; quota or filling up a pack.  RMS_ERROR has code to ignore these two and we
                                0735  1082 ; must compensate here when FAB$L_ALQ is cleared by an $EXTEND failure.
                                0735  1083 ;
                                0735  1084 CHECK_EXTEND:
                         03  E0 0735  1085         BBS     #DISK_V_NOEXTEND,-      ; BR if flag says don't try to extend
                    4B 0B A8        0737  1086                 UETUNT$B_FLAGS(R8),DO_THE_WRITE
                   59   10 A8  D0 073A  1087         MOVL    UETUNT$L_ITER(R8),R9    ; See if we've done...
                       45  13 073E  1088         BEQL    DO_THE_WRITE            ; ...a positive...
                       5A  D4 0740  1089         CLRL    R10                     ; ... (EDIV wants quadword divisor)
              59   59   59  14  7B 0742  1090         EDIV    #BAKERS,R9,R9,R9        ; ...multiple of BAKERS...
                       59  D5 0747  1091         TSTL    R9                      ; ...writes to this file
                       3A  12 0749  1092         BNEQ    DO_THE_WRITE            ; BR if there's a remainder
                   59   01A7 C8  D0 074B  1093         MOVL    DISK_L_OTHER_PTR(R8),R9 ; Calculate...
                        01AF C9  C1 0750  1094         ADDL3   DISK_L_SIZE(R9),-       ; ...the total space we're using...
                   59   01AF C8        0754  1095                 DISK_L_SIZE(R8),R9      ; ...on this unit
                   01AB C8   59  D1 0758  1096         CMPL    R9,DISK_L_TOP_PCT(R8)   ; Have we used as much as we should?
                       26  18 075D  1097         BGEQ    DO_THE_WRITE            ; BR if we've reached maximum space
                                075F  1098         $EXTEND FAB = (R7),-            ; We're OK, try to expand the file
                                075F  1099                 ERR = RMS_ERROR
                01AF C8   10 A7  C0 076C  1100         ADDL2   FAB$L_ALQ(R7),DISK_L_SIZE(R8) ; Compute new current size
                       10 50  E8 0772  1101         BLBS    R0,DO_THE_WRITE         ; BR if we're OK
                       08  88 0775  1102         BISB2   #DISK_M_NOEXTEND,-      ; Prevent further attempts to extend
                    0B A8        0777  1103                 UETUNT$B_FLAGS(R8)
                59   10 A8  C7 0779  1104         DIVL3   #BAKERS,UETUNT$L_ITER(R8),R9 ; Number writes/writes-per-extend...
          10 A7   01AF C8   59  C7 077E  1105         DIVL3   R9,DISK_L_SIZE(R8),FAB$L_ALQ(R7) ; ...divides file size giving exten
```

```
                                0785  1107 ;
                                0785  1108 ; Get the next block number to write.
                                0785  1109 ;
                                0785  1110 DO_THE_WRITE:
          01 0002'CF    01      E1  0785  1111          BBC     #TEST_OVERV,FLAG,10$    ; BR if the test is still going
                                04  078B  1112          RET                             ; Finished.  Dismiss AST
                                078C  1113 10$:
      0166'CF    016A'CF        C0  078C  1114          ADDL2   RANDOM2,RANDOM1         ; Generate a "random" number
59    0166'CF    80000000      8F   CB  0793  1115          BICL3   #^X80000000,RANDOM1,R9 ; From it get a new block number,...
                                5A  D4  079D  1116          CLRL    R10                     ; ... (EDIV wants quadword dividend)
59    59    59    01AF  C8      7B  079F  1117          EDIV    DISK_L_SIZE(R8),R9,R9,R9 ; ...modulus the file size,...
                  38 A6   59    D0  07A6  1118          MOVL    R9,RAB$L_BKT(R6)        ; ...for next $WRITE
                          E0    13  07AA  1119          BEQL    10$                     ; But don't allow 0 as a block number!
                                07AC  1120 ;
                                07AC  1121 ; We have the block number to write next;  now we must determine what data will
                                07AC  1122 ; be written.
                                07AC  1123 ;
          06 0002'CF    05      E0  07AC  1124          BBS     #PATTERNV,FLAG,15$      ; BR if we've forced pattern data
                  38 A6   03    D3  07B2  1125          BITL    #3,RAB$L_BKT(R6)        ; Determine which pattern to write next
                          0B    12  07B6  1126          BNEQ    20$                     ; BR if any bits are set
                                07B8  1127 15$:
                0800 8F   28    28  07B8  1128          MOVC3   #WRITE_SIZE,-           ; Either random or forced pattern
          28 B6   01A0'CF               07BC  1129                  RANDOM_DATA,@RAB$L_RBF(R6)
                          0C    11  07C1  1130          BRB     30$                     ; Go write the block
                                07C3  1131 20$:
                   001F   30    30  07C3  1132          BSBW    GET_PATTERN             ; Fill R9 with the pattern to write
28 B6   0800 8F   59    66   00  2C  07C6  1133          MOVC5   #0,(R6),R9,#WRITE_SIZE,@RAB$L_RBF(R6) ; Buffer gets pattern data
                                07CF  1134 30$:
                     10 A8    D6  07CF  1135          INCL    UETUNT$L_ITER(R8)       ; Count this file's writes
                                07D2  1136          $WRITE  RAB = @4(AP),-          ; Write another block to current file
                                07D2  1137                  ERR = RMS_ERROR,-
                                07D2  1138                  SUC = READ_NEXT
                                04  07E4  1139          RET                             ; Dismiss the AST
                                07E5  1140
                                07E5  1141
                                07E5  1142
                                07E5  1143
                                07E5  1144 ;
                                07E5  1145 ; Subroutine to fill the low order byte of R9 with the bit pattern of bits 0
                                07E5  1146 ; and 1 of the block number (presumably random) in RAB$L_BKT.  The pattern will
                                07E5  1147 ; be carried throughout the buffer on a $WRITE and later used to verify that
                                07E5  1148 ; the data were written out and read in correctly.  This subroutine will not
                                07E5  1149 ; be used if the bit pattern is ^B00 or if the flag PATTERNM is set; in those
                                07E5  1150 ; cases the contents of the RANDOM_DATA buffer will be used.
                                07E5  1151 ;
                                07E5  1152 GET_PATTERN:
    59    38 A6    02    00    EE  07E5  1153          EXTV    #0,#2,RAB$L_BKT(R6),R9  ; Get sign-extended bit pattern
                          05    19  07EB  1154          BLSS    10$                     ; BR if ^B11 or ^B10 pattern
                59    55 8F   90  07ED  1155          MOVB    #^B01010101,R9          ; Prepare to propagate the bit pattern
                          05    05  07F1  1156          RSB
                                07F2  1157 10$:
                      04 59   E8  07F2  1158          BLBS    R9,20$                  ; BR if ^B11 pattern (R9 will have -1)
                59    AA 8F   90  07F5  1159          MOVB    #^B10101010,R9          ; Prepare to propagate the bit pattern
                                07F9  1160 20$:
                          05    05  07F9  1161          RSB
```

```
                                07FA  1163          .SBTTL  Successful $WRITE AST Routine
                                07FA  1164 ;++
                                07FA  1165 ; FUNCTIONAL DESCRIPTION:
                                07FA  1166 ;       This routine will receive the AST when a $WRITE operation finishes for
                                07FA  1167 ;       a file on a unit.  It starts up a $READ operation and dismisses the
                                07FA  1168 ;       AST.
                                07FA  1169 ;
                                07FA  1170 ; CALLING SEQUENCE:
                                07FA  1171 ;       Called via AST at I/O completion for any file on any unit.
                                07FA  1172 ;
                                07FA  1173 ; INPUT PARAMETERS:
                                07FA  1174 ;       4(AP) is RAB address
                                07FA  1175 ;
                                07FA  1176 ; IMPLICIT INPUTS:
                                07FA  1177 ;       All RABs and FABs, implicitly).
                                07FA  1178 ;       An AST argument list with the ASTPRM value equal to the address of the
                                07FA  1179 ;       associated RAB.
                                07FA  1180 ;
                                07FA  1181 ; OUTPUT PARAMETERS:
                                07FA  1182 ;       NONE
                                07FA  1183 ;
                                07FA  1184 ; IMPLICIT OUTPUTS:
                                07FA  1185 ;       Fields may be modified in RABs.
                                07FA  1186 ;       If a file is being processed correctly and the test has not terminated,
                                07FA  1187 ;       the block of the file just written is read back in as verification.
                                07FA  1188 ;
                                07FA  1189 ; COMPLETION CODES:
                                07FA  1190 ;       NONE
                                07FA  1191 ;
                                07FA  1192 ; SIDE EFFECTS:
                                07FA  1193 ;       NONE
                                07FA  1194 ;
                                07FA  1195 ;--
                                07FA  1196
                                07FA  1197 READ_NEXT:
                          0FFC  07FA  1198          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                07FC  1199
                6D   09A5'CF    DE  07FC  1200          MOVAL   SSERROR,(FP)                ; Declare exception handler
             1F 0002'CF    01    E0  0801  1201          BBS     #TEST_OVERV,FLAG,10$        ; Stop ASAP if test is over
                  56    04 AC    D0  0807  1202          MOVL    4(AP),R6                    ; Get the address of our RAB
28 B6    0800 8F    00    66    00  2C  080B  1203          MOVC5   #0,(R6),#0,#WRITE_SIZE,@RAB$L_RBF(R6) ; Clear the read buffer
                                0814  1204          $READ   RAB = @4(AP),-              ; Read back the block just written
                                0814  1205                  ERR = RMS_ERROR,-
                                0814  1206                  SUC = WRITE_NEXT
                                0826  1207 10$:
                          04    0826  1208          RET                                 ; Dismiss the AST
```

```
                     0827  1210          .SBTTL  Left Over File Cleanup Routine
                     0827  1211 ;++
                     0827  1212 ; FUNCTIONAL DESCRIPTION:
                     0827  1213 ;       If there are files left over from a previous run of the disk test (as
                     0827  1214 ;       would happen, for example, if the system crashed during a run), this
                     0827  1215 ;       routine will clean up the highest version of the file we're about to
                     0827  1216 ;       create.  This is preferred over a supersede operation because supersede
                     0827  1217 ;       will not delete the old file's space until after the new file is
                     0827  1218 ;       created.
                     0827  1219 ;
                     0827  1220 ; CALLING SEQUENCE:
                     0827  1221 ;       BSBW    PRE_CLEAN
                     0827  1222 ;
                     0827  1223 ; INPUT PARAMETERS:
                     0827  1224 ;       NONE
                     0827  1225 ;
                     0827  1226 ; IMPLICIT INPUTS:
                     0827  1227 ;       Current FAB, pointed to as UETUNT$K_FAB(R6).
                     0827  1228 ;
                     0827  1229 ; OUTPUT PARAMETERS:
                     0827  1230 ;       NONE
                     0827  1231 ;
                     0827  1232 ; IMPLICIT OUTPUTS:
                     0827  1233 ;       NONE
                     0827  1234 ;
                     0827  1235 ; COMPLETION CODES:
                     0827  1236 ;       NONE
                     0827  1237 ;
                     0827  1238 ; SIDE EFFECTS:
                     0827  1239 ;       File may be deleted.  Note that we do not check for any errors.  An
                     0827  1240 ;       error from the $ERASE may be expected in situations where this test
                     0827  1241 ;       is run in parallel with itself on a given disk (e.g., from separate
                     0827  1242 ;       nodes in a cluster to an MSCP disk).
                     0827  1243 ;       R0 is trashed.
                     0827  1244 ;
                     0827  1245 ;--
                     0827  1246
                     0827  1247 PRE_CLEAN:
     07FE 8F    BB   0827  1248          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ; We clobber nothing here...
     5A   5E    D0   082B  1249          MOVL    SP,R10                      ; ...including our stack!
57   0110 C6    DE   082E  1250          MOVAL   UETUNT$K_FAB(R6),R7         ; Use R7 for more visible access
  58   01 A7    9A   0833  1251          MOVZBL  FAB$B_BLN(R7),R8            ; Again, we clobber nothing...
     5E   58    C2   0837  1252          SUBL2   R8,SP                       ; ...so we'll do our RMS access...
6E   67   58    28   083A  1253          MOVC3   R8,(R7),(SP)                ; ...using a FAB on the stack
     59   5E    D0   083E  1254          MOVL    SP,R9                       ; Save a pointer to our new FAB
                     0841  1255          $OPEN   FAB = R9                    ; See if old version of file is there
       12 50    E9   084A  1256          BLBC    R0,10$                      ; We can do nothing if any error
                     084D  1257          $CLOSE  FAB = R9                    ; We found one...
                     0856  1258          $ERASE  FAB = R9                    ; ...get rid of it
                     085F  1259 10$:
     5E   5A    D0   085F  1260          MOVL    R10,SP                      ; Restore our stack...
     07FE 8F    BA   0862  1261          POPR    #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ; ...and all our registers
                05   0866  1262          RSB
```

```
                     0867  1264          .SBTTL  File Cleanup Routine
                     0867  1265 ;++
                     0867  1266 ; FUNCTIONAL DESCRIPTION:
                     0867  1267 ;       This routine will clean up the files used for the test.
                     0867  1268 ;
                     0867  1269 ; CALLING SEQUENCE:
                     0867  1270 ;       BSBW    CLEAN_UP
                     0867  1271 ;
                     0867  1272 ; INPUT PARAMETERS:
                     0867  1273 ;       NONE
                     0867  1274 ;
                     0867  1275 ; IMPLICIT INPUTS:
                     0867  1276 ;       All FABs.  UNIT_LIST points to a queue with unit-file nodes.
                     0867  1277 ;
                     0867  1278 ; OUTPUT PARAMETERS:
                     0867  1279 ;       NONE
                     0867  1280 ;
                     0867  1281 ; IMPLICIT OUTPUTS:
                     0867  1282 ;       NONE
                     0867  1283 ;
                     0867  1284 ; COMPLETION CODES:
                     0867  1285 ;       NONE
                     0867  1286 ;
                     0867  1287 ; SIDE EFFECTS:
                     0867  1288 ;       Files are closed, deleted and erased.  All registers may be trash.
                     0867  1289 ;
                     0867  1290 ;--
                     0867  1291
                     0867  1292 CLEAN_UP:
    56   0190'CF    DE  0867  1293          MOVAL   UNIT_LIST,R6            ; This will point to the current unit
                     086C  1294 10$:
         56    66    C0  086C  1295          ADDL2   (R6),R6                 ; Loop through all nodes
00000190'8F    56    D1  086F  1296          CMPL    R6,#UNIT_LIST           ; Back at the head of the queue yet?
               2F    13  0876  1297          BEQL    40$                     ; BR if we've finished
                     0878  1298
               02    E0  0878  1299          BBS     #DISK_V_NOCREATE,-      ; Don't try $CLOSE if $CREATE failed
      EF 0B A6           087A  1300                  UETUNT$B_FLAGS(R6),10$
    58   0110 C6    DE  087D  1301          MOVAL   UETUNT$K_FAB(R6),R8     ; Get FAB address
                     0882  1302 20$:
                     0882  1303          $WAIT   RAB = UETUNT$K_RAB(R6)  ; Twiddle thumbs until I/O completes
                     088D  1304          $CLOSE  FAB = (R8)              ; Try to close the file
 016E'CF    10 A6    C0  0896  1305          ADDL2   UETUNT$L_ITER(R6),ITERATION ; Count total I/O for test
                     089C  1306          $ERASE  FAB = (R8)              ; File closed, remove directory entry
               C5    11  08A5  1307          BRB     10$
                     08A7  1308 40$:
                     05  08A7  1309          RSB                             ; We're all cleaned up
```

```
                        08A8  1311          .SBTTL  Bum Data Routine
                        08A8  1312 ;++
                        08A8  1313 ; FUNCTIONAL DESCRIPTION:
                        08A8  1314 ;       Print an error message if we get a mismatch during data verification.
                        08A8  1315 ;
                        08A8  1316 ; CALLING SEQUENCE:
                        08A8  1317 ;       CMPCx   ...@RAB$L_RBF(R6)
                        08A8  1318 ;       BEQL    data-was-good
                        08A8  1319 ;       BSBW    VERIFY_ERROR
                        08A8  1320 ;
                        08A8  1321 ; INPUT PARAMETERS:
                        08A8  1322 ;       NONE
                        08A8  1323 ;
                        08A8  1324 ; IMPLICIT INPUTS:
                        08A8  1325 ;       R1 points to the good data if we just read random data or forced
                        08A8  1326 ;       pattern data, never-never land if we read generated pattern data.
                        08A8  1327 ;       R3 points to the offending byte of differing data.
                        08A8  1328 ;       R6 points to the RAB for the bad file.
                        08A8  1329 ;       R8 points to the unit data block.
                        08A8  1330 ;       R9 will have the pattern if we matched generated pattern data.
                        08A8  1331 ;
                        08A8  1332 ; OUTPUT PARAMETERS:
                        08A8  1333 ;       NONE
                        08A8  1334 ;
                        08A8  1335 ; IMPLICIT OUTPUTS:
                        08A8  1336 ;       Message to SYS$OUTPUT and SYS$ERROR.
                        08A8  1337 ;       Registers R2-R8 and R10-R11 returned unscathed.
                        08A8  1338 ;
                        08A8  1339 ; COMPLETION CODES:
                        08A8  1340 ;       NONE
                        08A8  1341 ;
                        08A8  1342 ; SIDE EFFECTS:
                        08A8  1343 ;       NONE
                        08A8  1344 ;
                        08A8  1345 ;--
                        08A8  1346
                        08A8  1347 VERIFY_ERROR:
06 0002'CF    05    E0  08A8  1348          BBS     #PATTERNV,FLAG,5$       ; BR if we used forced pattern data
     38 A6    03    D3  08AE  1349          BITL    #3,RAB$L_BKT(R6)        ; Did we have random data?
              03    12  08B2  1350          BNEQ    10$                     ; BR if we used generated pattern data
                        08B4  1351 5$:
        59    61    90  08B4  1352          MOVB    (R1),R9                 ; Copy the byte where failure was found
                        08B7  1353 10$:
        7E    63    9A  08B7  1354          MOVZBL  (R3),-(SP)              ; Save the bad...
        7E    59    9A  08BA  1355          MOVZBL  R9,-(SP)                ; ...the good...
7E    53    28 A6    C3  08BD  1356          SUBL3   RAB$L_RBF(R6),R3,-(SP)  ; ...the place...
        01B3 C8    7F  08C2  1357          PUSHAQ  DISK_0_FILEDESC(R8)     ; ...the file spec...
    000F0004 8F    DD  08C6  1358          PUSHL   #^XF0004                ; ...
    0074801A 8F    DD  08CC  1359          PUSHL   #UETP$_DATADEVERR!STS$K_ERROR ; ...and the ugly info
        0142'CF    D6  08D2  1360          INCL    ERROR_COUNT             ; Keep a running error count
        0142'CF    DD  08D6  1361          PUSHL   ERROR_COUNT             ; ...
        00A0'CF    DF  08DA  1362          PUSHAL  PROCESS_NAME            ; ...
    00010002 8F    DD  08DE  1363          PUSHL   #^X10002                ; ...
    00748022 8F    DD  08E4  1364          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; Have error stand out in log file
00000000'GF    0A  FB  08EA  1365          CALLS   #10,G^LIB$SIGNAL        ; Let the user know what happened
              02    8A  08F1  1366          BICB2   #UETUNT$M_TESTABLE,-    ; Indicate that this file is no good
        FEAB C6         08F3  1367                  UETUNT$B_FLAGS-UETUNT$K_RAB(R6)
```

```
                0000   31  08F6  1368          BRW     DESTP_CHECK             ; Exit via possible additional message
```

```
                        08F9  1370          .SBTTL  Check if We've Stopped Testing a Unit
                        08F9  1371 ;++
                        08F9  1372 ; FUNCTIONAL DESCRIPTION:
                        08F9  1373 ;       In normal and loop forever modes, we warn the user that a device is
                        08F9  1374 ;       seen to be totally untestable (at least during this pass).
                        08F9  1375 ;
                        08F9  1376 ; CALLING SEQUENCE:
                        08F9  1377 ;       BSBW    DESTP_CHECK
                        08F9  1378 ;
                        08F9  1379 ; INPUT PARAMETERS:
                        08F9  1380 ;       NONE
                        08F9  1381 ;
                        08F9  1382 ; IMPLICIT INPUTS:
                        08F9  1383 ;       R8 points to the unit data block
                        08F9  1384 ;
                        08F9  1385 ; OUTPUT PARAMETERS:
                        08F9  1386 ;       NONE
                        08F9  1387 ;
                        08F9  1388 ; IMPLICIT OUTPUTS:
                        08F9  1389 ;       NONE
                        08F9  1390 ;
                        08F9  1391 ; COMPLETION CODES:
                        08F9  1392 ;       NONE
                        08F9  1393 ;
                        08F9  1394 ; SIDE EFFECTS:
                        08F9  1395 ;       Possible error message if all files for this unit are marked as
                        08F9  1396 ;       untestable.
                        08F9  1397 ;
                        08F9  1398 ;--
                        08F9  1399
                        08F9  1400 DESTP_CHECK:
                  56    DD  08F9  1401          PUSHL   R6                              ; We'll affect no registers herein
 30 0002'CF       04    E0  08FB  1402          BBS     #ONE_SHOTV,FLAG,20$             ; This is meaningless in one-shot mode
                  56    58  0901  1403          MOVL    R8,R6                           ; Set up a working pointer to node
                            0904  1404 10$:
                  01    E0  0904  1405          BBS     #UETUNT$V_TESTABLE,-            ; If this file is still good...
                28 0B   A6        0906  1406                  UETUNT$B_FLAGS(R6),20$  ; ...then we need no message
              56  01A7   C6  D0  0909  1407          MOVL    DISK_L_OTHER_PTR(R6),R6 ; Point to next node for this unit
                  58    56  D1  090E  1408          CMPL    R6,R8                   ; Back where we started?
                        F1  12  0911  1409          BNEQ    10$                     ; BR if not - check this one, too
                        00  DD  0913  1410          PUSHL   #0                      ; None testable - we need message
              7E  01A4  C8  3C  0915  1411          MOVZWL  DISK_W_UNIT(R8),-(SP)   ; Unit number
                  0098'CF   DF  091A  1412          PUSHAL  DEVDSC                  ; Controller
                  00A0'CF   DF  091E  1413          PUSHAL  PROCESS_NAME            ; Image name
                        04  DD  0922  1414          PUSHL   #4
               007481A2   8F  DD  0924  1415          PUSHL   #UETP$_DESTP!STS$K_ERROR
00000000'GF            06  FB  092A  1416          CALLS   #6,G^LIB$SIGNAL
                            0931  1417 20$:
                  56 8ED0   0931  1418          POPL    R6
                        05  0934  1419          RSB
```

```
                        0935  1421          .SBTTL  $CREATE-Specific Error Checker
                        0935  1422 ;++
                        0935  1423 ; FUNCTIONAL DESCRIPTION:
                        0935  1424 ;       This routine handles error returns from $CREATEs of test files.  It
                        0935  1425 ;       allows us to first try creating a file in device:[SYSTEST], and then
                        0935  1426 ;       for system packs with multiple system directory hierarchies, to try
                        0935  1427 ;       creating the file in device:[SYS0.SYSTEST].  This means we can test
                        0935  1428 ;       all packs, whether mounted as system packs or not.  The routine also
                        0935  1429 ;       allows us to recover from most RMS$_FUL errors by trying to allocate
                        0935  1430 ;       a file that is only 1% of the total space on a pack.
                        0935  1431 ;
                        0935  1432 ; CALLING SEQUENCE:
                        0935  1433 ;       BSBW    CREATE_CHECK
                        0935  1434 ;
                        0935  1435 ; INPUT PARAMETERS:
                        0935  1436 ;       NONE
                        0935  1437 ;
                        0935  1438 ; IMPLICIT INPUTS:
                        0935  1439 ;       R6 points to our unit data block for this file on this device.
                        0935  1440 ;       The FAB associated with the RMS $CREATE as UETUNT$K_FAB(R6).
                        0935  1441 ;
                        0935  1442 ; OUTPUT PARAMETERS:
                        0935  1443 ;       NONE
                        0935  1444 ;
                        0935  1445 ; IMPLICIT OUTPUTS:
                        0935  1446 ;       Error message
                        0935  1447 ;
                        0935  1448 ; COMPLETION CODES:
                        0935  1449 ;       NONE
                        0935  1450 ;
                        0935  1451 ; SIDE EFFECTS:
                        0935  1452 ;       If the error can be corrected by retrying with a different directory,
                        0935  1453 ;       the test file will be created.  If the file can be created with a
                        0935  1454 ;       smaller allocation, it will be.  Otherwise, the program may exit,
                        0935  1455 ;       depending on severity of the error.  If the program does not exit
                        0935  1456 ;       and the file is not created, DISK_M_NOCREATE is set in UETUNT$B_FLAGS
                        0935  1457 ;       by the RMS error handler.  R0 and R2 are trashed.
                        0935  1458 ;
                        0935  1459 ;--
                        0935  1460
                        0935  1461 CREATE_CHECK:
                59 50   E8  0935  1462          BLBS    R0,30$                          ; BR if there was no error
            52  0110 C6   DE  0938  1463          MOVAL   UETUNT$K_FAB(R6),R2             ; Point to our FAB for convenience
    08 A2    00000000'8F   D1  093D  1464          CMPL    #RMS$_DNF,FAB$L_STS(R2)         ; Did we get directory not found error?
                    1B   12  0945  1465          BNEQ    10$                             ; BR if not
                0253'CF   90  0947  1466          MOVB    SYS0_SYSTEST_DIR,-              ; Yes, try it again...
                  35 A2       094B  1467                  FAB$B_DNS(R2)
                0254'CF   DE  094D  1468          MOVAL   SYS0_SYSTEST_DIR+1,-            ; ...using...
                  30 A2       0951  1469                  FAB$L_DNA(R2)                   ; ...V3 system directory hierarchies
                   FED1   30  0953  1470          BSBW    PRE_CLEAN                       ; Try again to clean up old file
                              0956  1471          $CREATE FAB = (R2)                      ; Try again to create new file
                  2F 50   E8  095F  1472          BLBS    R0,30$                          ; BR if we got it this time
                              0962  1473
                              0962  1474 10$:
            50   00000000'8F   D1  0962  1475          CMPL    #RMS$_FUL,R0                    ; Did we fail because file was too big?
                    1F   12  0969  1476          BNEQ    20$                             ; BR if not
50  CCCD3E4C 8F   000043C8 8F   45  096B  1477          MULF3   #^F100 0,#PERCENTAGE,R0 ; Try to get...
```

```
                50    50    4A  0977   1478          CVTFL   R0,R0                           ; ...
            10  A2    50    C6  097A   1479          DIVL2   R0,FAB$L_ALQ(R2)             ; ...1% of this unit's space...
                                0097E  1480          $CREATE FAB = (R2)                  ; ...and try again for the file
                      07  50    E8  0987   1481          BLBS    R0,30$                  ; BR if we got the file
                                0098A  1482
                                0098A  1483 20$:
                          52    DD  0098A  1484          PUSHL   R2                      ; We don't know which error we got...
                0A88'CF   01    FB  098C   1485          CALLS   #1,RMS_ERROR            ; ...so let real routine handle it
                                0991   1486 30$:
                                05  0991   1487          RSB
```

```
                     0992  1489          .SBTTL  Timer Expiration Routine
                     0992  1490 ;++
                     0992  1491 ; FUNCTIONAL DESCRIPTION:
                     0992  1492 ;       This routine will signal the end of the test when we time out.
                     0992  1493 ;
                     0992  1494 ; CALLING SEQUENCE:
                     0992  1495 ;       Called via AST at $SETIMR expiration.
                     0992  1496 ;
                     0992  1497 ; INPUT PARAMETERS:
                     0992  1498 ;       NONE
                     0992  1499 ;
                     0992  1500 ; IMPLICIT INPUTS:
                     0992  1501 ;       NONE
                     0992  1502 ;
                     0992  1503 ; OUTPUT PARAMETERS:
                     0992  1504 ;       NONE
                     0992  1505 ;
                     0992  1506 ; IMPLICIT OUTPUTS:
                     0992  1507 ;       NONE
                     0992  1508 ;
                     0992  1509 ; COMPLETION CODES:
                     0992  1510 ;       Sets a flag to indicate timer expiration.
                     0992  1511 ;
                     0992  1512 ; SIDE EFFECTS:
                     0992  1513 ;       The TEST_OVER flag is set in FLAG.  This will tell the AST level
                     0992  1514 ;       routines to not schedule any further I/O.
                     0992  1515 ;       The program is aWAKEned from its HIBERnating state.
                     0992  1516 ;       Note that no cleanup is done here.
                     0992  1517 ;
                     0992  1518 ;--
                     0992  1519
                     0992  1520 TIME_OUT:
                OFFC 0992  1521          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0994  1522
0002'CF    02   A8  0994  1523          BISW2   #TEST_OVERM,FLAG        ; Tell AST routines to stop
                     0999  1524          $WAKE_S                         ; Resume normal execution
                  04 09A4  1525          RET
```

```
                                                I 15
09A5  1527          .SBTTL  System Service Exception Handler
09A5  1528 ;++
09A5  1529 ; FUNCTIONAL DESCRIPTION:
09A5  1530 ;       This routine is executed if a software or hardware exception occurs or
09A5  1531 ;       if a LIB$SIGNAL system service is used to output a message.
09A5  1532 ;
09A5  1533 ; CALLING SEQUENCE:
09A5  1534 ;       Entered via an exception from the system
09A5  1535 ;
09A5  1536 ; INPUT PARAMETERS:
09A5  1537 ;       ERROR_COUNT    = previous cumulative error count
09A5  1538 ;
09A5  1539 ;          AP ---->      +----------------+
09A5  1540 ;                        |       2        |
09A5  1541 ;                        +----------------+
09A5  1542 ;                        |  SIGNL ARY PNT |
09A5  1543 ;                        +----------------+
09A5  1544 ;                        |  MECH  ARY PNT | ---------
09A5  1545 ;                        +----------------+         ^
09A5  1546 ;                        |       4        |         :
09A5  1547 ;                        +----------------+         :
09A5  1548 ;                        |  ESTABLISH FP  |         :
09A5  1549 ;                        +----------------+         :
09A5  1550 ;                        |     DEPTH      |  Mechanism Array
09A5  1551 ;                        +----------------+         :
09A5  1552 ;                        |       R0       |         :
09A5  1553 ;                        +----------------+         :
09A5  1554 ;                        |       R1       |         v
09A5  1555 ;                        +----------------+ ---------
09A5  1556 ;                        |       N        |         ^
09A5  1557 ;                        +----------------+         :
09A5  1558 ;                        |  CONDITION NAME|         :
09A5  1559 ;                        +----------------+         :
09A5  1560 ;                        | N-3 ADDITIONAL |    Signal Array
09A5  1561 ;                        | LONG WORD ARGS |         :
09A5  1562 ;                        +----------------+         :
09A5  1563 ;                        |       PC       |         :
09A5  1564 ;                        +----------------+         :
09A5  1565 ;                        |       PSL      |         v
09A5  1566 ;                        +----------------+ ---------
09A5  1567 ; IMPLICIT INPUTS:
09A5  1568 ;       NONE
09A5  1569 ;
09A5  1570 ; OUTPUT PARAMETERS:
09A5  1571 ;       NONE
09A5  1572 ;
09A5  1573 ; IMPLICIT OUTPUTS:
09A5  1574 ;       NONE
09A5  1575 ;
09A5  1576 ; COMPLETION CODES:
09A5  1577 ;       SS$_NORMAL if it's a UETP condition or RMS error.
09A5  1578 ;       Error status from exception, otherwise.
09A5  1579 ;
09A5  1580 ; SIDE EFFECTS:
09A5  1581 ;       May branch to ERROR_EXIT.
09A5  1582 ;       May print a message.
09A5  1583 ;--
```

```
                        09A5  1584 SSERROR:
                   0FFC 09A5  1585         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        09A7  1586
                        09A7  1587         $SETAST_S ENBFLG = #0            ; Disable AST delivery
                01      DD    09B0  1588         PUSHL   #1                       ; Assume ASTs were enabled
           50   09      D1    09B2  1589         CMPL    S^#SS$_WASSET,R0         ; Were ASTs enabled?
                02      13    09B5  1590         BEQL    10$                      ; BR if they were
                6E      D4    09B7  1591         CLRL    (SP)                     ; Set ASTs to remain disabled
                        09B9  1592 10$:
                        09B9  1593         $SETSFM_S ENBFLG = #0            ; Disable SS failure mode
                01      DD    09C2  1594         PUSHL   #1                       ; Assume SS failure mode was enabled
           50   09      D1    09C4  1595         CMPL    S^#SS$_WASSET,R0         ; Was SS failure mode enabled?
                02      13    09C7  1596         BEQL    20$                      ; BR if it was
                6E      D4    09C9  1597         CLRL    (SP)                     ; Set SS failure mode to remain off
                        09CB  1598 20$:
         56   04 AC     D0    09CB  1599         MOVL    CHF$L_SIGARGLST(AP),R6   ; Get the signal array pointer
         59   04 A6     7D    09CF  1600         MOVQ    CHF$L_SIG_NAME(R6),R9    ; Get NAME in R9 and ARG1 in R10
                10      ED    09D3  1601         CMPZV   #STS$V_FAC_NO,-          ; Is this a message from LIB$SIGNAL?
                0C            09D5  1602                 #STS$S_FAC_NO,-
00000074   8F   59            09D6  1603                 R9,#UETP$_FACILITY
                14      12    09DC  1604         BNEQ    30$                      ; BR if this is not a UETP exception
           66   02      C2    09DE  1605         SUBL2   #2,CHF$L_SIG_ARGS(R6)    ; Drop the PC and PSL
                        09E1  1606         $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6) ; Print the message
                21      11    09F0  1607         BRB     40$                      ; Restore ASTs and SS fail mode
                        09F2  1608 30$:
59 0000045C 8F   D1    09F2  1609         CMPL    #SS$_SSFAIL,R9           ; RMS failures are SysSvc failures
                32      12    09F9  1610         BNEQ    50$                      ; BR if this can't be an RMS failure
                10      ED    09FB  1611         CMPZV   #STS$V_FAC_NO,-          ; Is it an RMS failure?
                0C            09FD  1612                 #STS$S_FAC_NO,-
           01   5A            09FE  1613                 R10,#RMS$_FACILITY
                2B      12    0A00  1614         BNEQ    50$                      ; BR if not
5A   F0000000   8F      CA    0A02  1615         BICL2   #^XF0000000,R10           ; Strip control bits from status code
          08 A6 04      39    0A09  1616         MATCHC  #4,CHF$L_SIG_ARG1(R6),-  ; Is it an RMS failure for which...
                14            0A0D  1617                 #NRAT_LENGTH,-
           004D'CF            0A0E  1618                 NO_RMS_AST_TABLE         ; ...no AST can be delivered?
                1A      13    0A11  1619         BEQL    50$                      ; BR if so - must give error here
                        0A13  1620 40$:
                01      BA    0A13  1621         POPR    #^M<R0>                  ; Restore SS failure mode...
                        0A15  1622         $SETSFM_S ENBFLG = R0            ; ...
                01      BA    0A1E  1623         POPR    #^M<R0>                  ; Restore AST enable...
                        0A20  1624         $SETAST_S ENBFLG = R0            ; ...
           50   01      D0    0A29  1625         MOVL    S^#SS$_NORMAL,R0         ; Supply a standard status for exit
                04      0A2C  1626         RET                              ; Resume processing (or goto RMS_ERROR)
                        0A2D  1627 50$:
     0146'CF    59      D0    0A2D  1628         MOVL    R9,STATUS                ; Save the status
                58      D4    0A32  1629         CLRL    R8                       ; Assume for now it's not SS failure
59   0000045C   8F      D1    0A34  1630         CMPL    #SS$_SSFAIL,R9           ; But is it a System Service failure?
                38      12    0A3B  1631         BNEQ    70$                      ; BR if not - no special case message
                        0A3D  1632         $GETMSG_S MSGID = R10,-          ; Get SS failure code associated text
                        0A3D  1633                   MSGLEN = BUFFER_PTR,-
                        0A3D  1634                   BUFADR = FAO_BUF,-
                        0A3D  1635                   FLAGS  = #14,-
                        0A3D  1636                   OUTADR = MSG_BLOCK
           0177'CF  95    0A54  1637         TSTB    MSG_BLOCK+1              ; Get FAO arg count for SS failure code
                16      13    0A58  1638         BEQL    60$                      ; Don't use $GETMSG if no $FAO args...
           000C'CF  DF    0A5A  1639         PUSHAL  BUFFER_PTR               ; ...else build up...
                01      DD    0A5E  1640         PUSHL   #1                       ; ...a message describing...
```

```
                00741130 8F    DD   0A60  1641          PUSHL   #UETP$_TEXT                     ; ...why the System Service failed
                     00  5A    F0   0A66  1642          INSV    R10,#STS$V_SEVERITY,-           ; Give the message...
                     6E  03         0A69  1643                  #STS$S_SEVERITY,(SP)            ; ...the correct severity code
                     58  03    D0   0A6B  1644          MOVL    #3,R8                           ; Count the number of args we pushed
                         05    11   0A6E  1645          BRB     70$
                                    0A70  1646 60$:
                         5A    DD   0A70  1647          PUSHL   R10                             ; Save SS failure code
                     58  01    D0   0A72  1648          MOVL    #1,R8                           ; Count the number of args we pushed
                                    0A75  1649 70$:
                57   66  04    C5   0A75  1650          MULL3   #4,CHF$L_SIG_ARGS(R6),R7 ; Convert longwords to bytes
                     5E  57    C2   0A79  1651          SUBL2   R7,SP                           ; Save the current signal array...
            6E   04  A6  57    28   0A7C  1652          MOVC3   R7,CHF$L_SIG_NAME(R6),(SP) ; ...on the stack
                7E   66  58    C1   0A81  1653          ADDL3   R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
                        0114   31   0A85  1654          BRW     ERROR_EXIT
                                    0A88  1655
```

```
                     0A88  1657          .SBTTL  General RMS Error Handler
                     0A38  1658 ;++
                     0A88  1659 ; FUNCTIONAL DESCRIPTION:
                     0A88  1660 ;       This routine handles error returns from RMS calls.
                     0A88  1661 ;
                     0A88  1662 ; CALLING SEQUENCE:
                     0A88  1663 ;       Called by RMS when a file processing error is found.  Will also be
                     0A88  1664 ;       called by TRY_SYS0 if an error other than directory-not-found is
                     0A88  1665 ;       encountered when creating a test file.
                     0A88  1666 ;
                     0A88  1667 ; INPUT PARAMETERS:
                     0A88  1668 ;       The FAB or RAB associated with the RMS call.
                     0A88  1669 ;
                     0A88  1670 ; IMPLICIT INPUTS:
                     0A88  1671 ;       NONE
                     0A88  1672 ;
                     0A88  1673 ; OUTPUT PARAMETERS:
                     0A88  1674 ;       NONE
                     0A88  1675 ;
                     0A88  1676 ; IMPLICIT OUTPUTS:
                     0A88  1677 ;       Error message
                     0A88  1678 ;
                     0A88  1679 ; COMPLETION CODES:
                     0A88  1680 ;       NONE
                     0A88  1681 ;
                     0A88  1682 ; SIDE EFFECTS:
                     0A88  1683 ;       Program may exit, depending on severity of the error.
                     0A88  1684 ;
                     0A88  1685 ;--
                     0A88  1686
                     0A88  1687 RMS_ERROR:
                0FFC 0A88  1688          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0A8A  1689
      56   04 AC    D0  0A8A  1690          MOVL    4(AP),R6                ; See whether we're dealing with...
         66   03    91  0A8E  1691          CMPB    #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
              16    12  0A91  1692          BNEQ    10$                     ; BR if it's a RAB
   57   01F5'CF    DE  0A93  1693          MOVAL   FILE,R7                 ; FAB-specific code:  text string...
        58   56    D0  0A98  1694          MOVL    R6,R8                   ; ...address of FAB...
          0C A6    DD  0A9B  1695          PUSHL   FAB$L_STV(R6)           ; ...STV field for error...
          08 A6    DD  0A9E  1696          PUSHL   FAB$L_STS(R6)           ; ...STS field for error...
0146'CF   08 A6    D0  0AA1  1697          MOVL    FAB$L_STS(R6),STATUS    ; ...and save the error code
              15    11  0AA7  1698          BRB     COMMON                  ; FAB and RAB share other code
                     0AA9  1699 10$:
   57   0201'CF    DE  0AA9  1700          MOVAL   RECORD,R7               ; RAB-specific code:  text string...
     58   3C A6    D0  0AAE  1701          MOVL    RAB$L_FAB(R6),R8        ; ...address of associated FAB...
          0C A6    DD  0AB2  1702          PUSHL   RAB$L_STV(R6)           ; ...STV field for error...
          08 A6    DD  0AB5  1703          PUSHL   RAB$L_STS(R6)           ; ...STS field for error...
0146'CF   08 A6    D0  0AB8  1704          MOVL    RAB$L_STS(R6),STATUS    ; ...and save the error code
                     0ABE  1705 COMMON:
      5A   34 A8    9A  0ABE  1706          MOVZBL  FAB$B_FNS(R8),R10
                     0AC2  1707          $FAO_S  CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
                     0AC2  1708                  OUTLEN = BUFFER_PTR,-
                     0AC2  1709                  OUTBUF = FAO_BUF,-
                     0AC2  1710                  P1     = R7,-
                     0AC2  1711                  P2     = R10,-
                     0AC2  1712                  P3     = FAB$L_FNA(R8)
          000C'CF    DF  0ADC  1713          PUSHAL  BUFFER_PTR              ; ...and arguments for ERROR_EXIT...
```

```
                000F0001 8F    DD   0AE0  1714          PUSHL   #^XF0001                        ; ...
                00741130 8F    DD   0AE6  1715          PUSHL   #UETP$_TEXT                     ; ...
                      00       EF   0AEC  1716          EXTZV   #STS$V_SEVERITY,-
                      03            0AEE  1717                  #STS$S_SEVERITY,-
              59   0146'CF           0AEF  1718                  STATUS,R9                       ; ...get the severity code...
                   6E   59    88   0AF3  1719          BISB2   R9,(SP)                         ; ...and add it into the signal name
          55555555 8F   18 A8   D1   0AF6  1720          CMPL    FAB$L_CTX(R8),#DISK_K_PATTERN ; A test or an overhead file?
                      05       13   0AFE  1721          BEQL    TEST_FILE_ERROR                 ; BR if it's a test file
                      05       DD   0B00  1722          PUSHL   #5                              ; Current arg count
                    0097       31   0B02  1723          BRW     ERROR_EXIT                      ; Can't complete test normally
                               0B05  1724
                               0B05  1725 TEST_FILE_ERROR:
                   0146'CF     D4   0B05  1726          CLRL    STATUS                          ; Recoverable error, reset exit status
              5B   FEF0 C8     DE   0B09  1727          MOVAL   -UETUNT$K_FAB(R8),R11   ; Point to node for this FAB
                OE AB   01     B1   0B0E  1728          CMPW    #DISK_K_CREATE,UETUNT$W_FUNC(R11) ; Were we doing a $CREATE?
                      14       13   0B12  1729          BEQL    10$                             ; BR if so (all errors count)
                               0B14  1730
                               0B14  1731 ; We sometimes expect SS$_EXDISKQUOTA or RMS$_FUL errors as a natural
                               0B14  1732 ; consequence of trying to extend a test file.  Ignore them here and let
                               0B14  1733 ; inline code where the $EXTEND is done handle the error.  That same code will
                               0B14  1734 ; prevent further attempts at expansion because R0 is immediately returned
                               0B14  1735 ; with an error value and DISK_M_NOEXTEND is set for the file.  Note that the
                               0B14  1736 ; RET instruction below will clean up the stack.
          0C A8   000003EC 8F  D1   0B14  1737          CMPL    #SS$_EXDISKQUOTA,FAB$L_STV(R8) ; Quota exhausted error?
                      3A       13   0B1C  1738          BEQL    20$                             ; BR if so
          08 A8   00000000'8F  D1   0B1E  1739          CMPL    #RMS$_FUL,FAB$L_STS(R8) ; Error because unit was full?
                      30       13   0B26  1740          BEQL    20$                             ; BR if so
                               0B28  1741
                               0B28  1742 10$:
                0142'CF     D6   0B28  1743          INCL    ERROR_COUNT                     ; Keep a running error count
                0142'CF     DD   0B2C  1744          PUSHL   ERROR_COUNT                     ; Have the error stand out in log file
                00A0'CF     DF   0B30  1745          PUSHAL  PROCESS_NAME                    ; ...
              00010002 8F   DD   0B34  1746          PUSHL   #^X10002                        ; ...
              00748020 8F   DD   0B3A  1747          PUSHL   #UETP$_ERBOXPROC                ; Set the message code...
                   6E   59    88   0B40  1748          BISB2   R9,(SP)                         ; ...and the severity
        00000000'GF     09    FB   0B43  1749          CALLS   #9,G^LIB$SIGNAL                 ; Let the user know what happened
                      02       8A   0B4A  1750          BICB2   #UETUNT$M_TESTABLE,-     ; Indicate that this file is no good
                   0B AB            0B4C  1751                  UETUNT$B_FLAGS(R11)
                01 0E AB    B1     0B4E  1752          CMPW    UETUNT$W_FUNC(R11),#DISK_K_CREATE ; Were we doing a $CREATE?
                      04       12   0B52  1753          BNEQ    20$                             ; BR if not
                      04       88   0B54  1754          BISB2   #DISK_M_NOCREATE,-      ; Prevent trying to $CLOSE
                   0B AB            0B56  1755                  UETUNT$B_FLAGS(R11)
                               0B58  1756 20$:
              58   5B     D0     0B58  1757          MOVL    R11,R8                          ; Set up args to see...
                 FD9B     30     0B5B  1758          BSBW    DESTP_CHECK                     ; ...if additional message is needed
                               04   0B5E  1759          RET                                     ; No more I/O to this file
                               0B5F  1760
```

```
                        0B5F  1762          .SBTTL  CTRL/C Handler
                        0B5F  1763 ;++
                        0B5F  1764 ; FUNCTIONAL DESCRIPTION:
                        0B5F  1765 ;       This routine handles CTRL/C AST's by printing a message and exiting.
                        0B5F  1766 ;
                        0B5F  1767 ; CALLING SEQUENCE:
                        0B5F  1768 ;       Called via AST
                        0B5F  1769 ;
                        0B5F  1770 ; INPUT PARAMETERS:
                        0B5F  1771 ;       NONE
                        0B5F  1772 ;
                        0B5F  1773 ; IMPLICIT INPUTS:
                        0B5F  1774 ;       NONE
                        0B5F  1775 ;
                        0B5F  1776 ; OUTPUT PARAMETERS:
                        0B5F  1777 ;       NONE
                        0B5F  1778 ;
                        0B5F  1779 ; IMPLICIT OUTPUTS:
                        0B5F  1780 ;       $EXIT status value
                        0B5F  1781 ;
                        0B5F  1782 ; COMPLETION CODES:
                        0B5F  1783 ;       NONE
                        0B5F  1784 ;
                        0B5F  1785 ; SIDE EFFECTS:
                        0B5F  1786 ;       NONE
                        0B5F  1787 ;
                        0B5F  1788 ;--
                        0B5F  1789
                        0B5F  1790 CCASTHAND:
                OFFC    0B5F  1791          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0B61  1792
        00A3'CF   DF    0B61  1793          PUSHAL  CNTRLCMSG                   ; Set message pointer
              01  DD    0B65  1794          PUSHL   #1                          ; Set arg count
        00741130  8F    0B67  1795          PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
              00  DD    0B6D  1796          PUSHL   #0                          ; Indicate an abnormal termination
        00A0'CF   DF    0B6F  1797          PUSHAL  PROCESS_NAME                ; ...
              02  DD    0B73  1798          PUSHL   #2                          ; ...
        007410E0  8F    0B75  1799          PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
00000000'GF   07  FB    0B7B  1800          CALLS   #7,G^LIB$SIGNAL             ; Output the message
                  D0    0B82  1801          MOVL    #<STS$M_INHIB_MSG!-         ; Set the exit status
                        0B83  1802                  SS$_CONTROLC-=
                        0B83  1803                  STS$K_SUCCESS+STS$K_WARNING>,-
0146'CF 10000650  8F    0B83  1804                  STATUS
      0002'CF   02  A8  0B8B  1805          BISW2   #TEST_OVERM,FLAG            ; Indicate that the test is over...
                        0B90  1806          $WAKE_S                             ; ...and allow it...
                  04    0B9B  1807          RET                                 ; ...to terminate cleanly
```

```
                        0B9C  1809          .SBTTL  Error Exit
                        0B9C  1810 ;++
                        0B9C  1811 ; FUNCTIONAL DESCRIPTION:
                        0B9C  1812 ;       This routine prints an error message and exits.
                        0B9C  1813 ;
                        0B9C  1814 ; CALLING SEQUENCE:
                        0B9C  1815 ;       MOVx  error status value,STATUS
                        0B9C  1816 ;       PUSHx error specific information on the stack
                        0B9C  1817 ;       PUSHL current argument count
                        0B9C  1818 ;       BRW   ERROR_EXIT
                        0B9C  1819 ;
                        0B9C  1820 ; INPUT PARAMETERS:
                        0B9C  1821 ;       Arguments to LIB$SIGNAL, as above
                        0B9C  1822 ;
                        0B9C  1823 ; IMPLICIT INPUTS:
                        0B9C  1824 ;       NONE
                        0B9C  1825 ;
                        0B9C  1826 ; OUTPUT PARAMETERS:
                        0B9C  1827 ;       Message to SYS$OUTPUT and SYS$ERROR
                        0B9C  1828 ;
                        0B9C  1829 ; IMPLICIT OUTPUTS:
                        0B9C  1830 ;       Program exit
                        0B9C  1831 ;
                        0B9C  1832 ; COMPLETION CODES:
                        0B9C  1833 ;       NONE
                        0B9C  1834 ;
                        0B9C  1835 ; SIDE EFFECTS:
                        0B9C  1836 ;       NONE
                        0B9C  1837 ;
                        0B9C  1838 ;--
                        0B9C  1839
                        0B9C  1840 ERROR_EXIT:
                        0B9C  1841
                        0B9C  1842          $SETAST_S ENBFLG = #0           ; ASTs can play havoc with messages
      15 0002'CF    03    E0  0BA5  1843          BBS     #BEGIN_MSGV,FLAG,10$    ; BR if "begin" msg already printed
                     7E     D4  0BAB  1844          CLRL    -(SP)                   ; Set the time stamp flag
            000F'CF  DF     0BAD  1845          PUSHAL  TEST_NAME               ; Set the test name
                     02     DD  0BB1  1846          PUSHL   #2                      ; Push the argument count
        00741039  8F     DD  0BB3  1847          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
    00000000'GF  04     FB  0BB9  1848          CALLS   #4,G^LIB$SIGNAL         ; Print the startup message
                        0BC0  1849 10$:
    018A'CF  08     8E  C1  0BC0  1850          ADDL3   (SP)+,#8,ARG_COUNT      ; Get total # args, pop partial count
            0142'CF  D6     0BC6  1851          INCL    ERROR_COUNT             ; Keep running error count
                     00     DD  0BCA  1852          PUSHL   #0                      ; Push the time parameter
            00A0'CF  DF     0BCC  1853          PUSHAL  PROCESS_NAME            ; Push test name...
        000F0002  8F     DD  0BD0  1854          PUSHL   #^XF0002                ; ...arg count...
        007410E2  8F     DD  0BD6  1855          PUSHL   #UETP$_ABENDD!STS$K_ERROR ; ...and signal name
            0142'CF  DD     0BDC  1856          PUSHL   ERROR_COUNT             ; Finish off arg list...
            00A0'CF  DF     0BE0  1857          PUSHAL  PROCESS_NAME            ; ...
        00010002  8F     DD  0BE4  1858          PUSHL   #^X10002                ; ...
        00748022  8F     DD  0BEA  1859          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
00000000'GF  018A'CF  FB     0BF0  1860          CALLS   ARG_COUNT,G^LIB$SIGNAL  ; Truly bitch
                        0BF9  1861
            0146'CF  D5     0BF9  1862          TSTL    STATUS                  ; Did we exit with an error code?
                     09     12  0BFD  1863          BNEQ    20$                     ; BR if we did
        007410E2  8F     D0  0BFF  1864          MOVL    #UETP$_ABENDD!STS$K_ERROR,- ; Supply a generic one otherwise
            0146'CF         0C05  1865                  STATUS
```

```
                     0C08  1866 20$:
0146'CF   10000000 8F CB 0C08  1867          BISL    #STS$M_INHIB_MSG,STATUS ; Don't print messages twice!
                     0C11  1868          $EXIT_S STATUS                   ; Exit in error
```

```
                        0C1C  1870          .SBTTL  Exit Handler
                        0C1C  1871 ;++
                        0C1C  1872 ; FUNCTIONAL DESCRIPTION:
                        0C1C  1873 ;       This routine handles cleanup at exit.  If the MODE logical name is
                        0C1C  1874 ;       equated to "ONE", the routine will update the test flag in the
                        0C1C  1875 ;       UETINIDEV.DAT file depending on the UETUNT$M_TESTABLE flag state in the
                        0C1C  1876 ;       UETUNT$B_FLAGS field of the unit blocks corresponding to a line in the
                        0C1C  1877 ;       file.
                        0C1C  1878 ;
                        0C1C  1879 ; CALLING SEQUENCE:
                        0C1C  1880 ;       Invoked automatically by $EXIT System Service.
                        0C1C  1881 ;
                        0C1C  1882 ; INPUT PARAMETERS:
                        0C1C  1883 ;       STATUS  contains the exit status.
                        0C1C  1884 ;       FLAG    has synchronizing bits.
                        0C1C  1885 ;       DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV
                        0C1C  1886 ;
                        0C1C  1887 ; IMPLICIT INPUTS:
                        0C1C  1888 ;       UNIT_LIST points to the head of a doubly linked circular list of unit
                        0C1C  1889 ;                 blocks for the device under test.
                        0C1C  1890 ;
                        0C1C  1891 ; OUTPUT PARAMETERS:
                        0C1C  1892 ;       NONE
                        0C1C  1893 ;
                        0C1C  1894 ; IMPLICIT OUTPUTS:
                        0C1C  1895 ;       Various files are de-accessed, the process name is reset, and any
                        0C1C  1896 ;       necessary synchronization with UETPDEV01 is carried out.
                        0C1C  1897 ;       If the MODE logical name is equated to "ONE", the routine will update
                        0C1C  1898 ;       the test flag in the UETINIDEV.DAT file depending on the
                        0C1C  1899 ;       UETUNT$M_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
                        0C1C  1900 ;       blocks corresponding to the units on the disk controller.
                        0C1C  1901 ;
                        0C1C  1902 ; COMPLETION CODES:
                        0C1C  1903 ;       NONE
                        0C1C  1904 ;
                        0C1C  1905 ; SIDE EFFECTS:
                        0C1C  1906 ;       NONE
                        0C1C  1907 ;
                        0C1C  1908 ;--
                        0C1C  1909
                        0C1C  1910 EXIT_HANDLER:
                   0FFC 0C1C  1911          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0C1E  1912
                        0C1E  1913          $SETSFM_S ENBFLG = #0               ; Turn off System Service failure mode
                        0C27  1914          $SETAST_S ENBFLG = #0               ; We're finished - no more ASTs
          03 0002'CF    04 E0 0C30  1915          BBS     #ONE_SHOTV,FLAG,10$     ; If one-shot, update testability...
                00CD    31 0C36  1916          BRW     END_UPDATE              ; ...else don't update UETINIDEV.DAT
                        0C39  1917 10$:
          03 0002'CF    02 E0 0C39  1918          BBS     #SAFE_TO_UPDV,FLAG,20$  ; Only update if it's safe
                00C4    31 0C3F  1919          BRW     END_UPDATE              ; Else forget it
                        0C42  1920 20$:
             5A 0A88'CF    DE 0C42  1921          MOVAL   INI_RAB,R10             ; Set the RAB address
               1E AA   02 90 0C47  1922          MOVB    #RAB$C_RFA,RAB$B_RAC(R10) ; Set RFA mode
    10 AA    0ACC'CF    06 28 0C4B  1923          MOVC3   #6,DDB_RFA,RAB$W_RFA(R10) ; Set RFA to DDB line
                        0C52  1924          $GET    RAB = (R10)             ; Go back to the DDB record
                 03 50  E8 0C5B  1925          BLBS    R0,30$
                  0087  31 0C5E  1926          BRW     UPDATE_FAILED           ; If failure then forget it
```

```
                                0C61  1927 30$:
                1E AA   00   90 0C61  1928          MOVB    #RAB$C_SEQ,RAB$B_RAC(R10) ; Set back to sequential mode
    5B  0190'CF  00000190'8F C1 0C65  1929          ADDL3   #UNIT_LIST,UNIT_LIST,R11 ; Set the unit block list header
                        59   D4 0C6F  1930          CLRL    R9                        ; Init a counter
                                0C71  1931 UNIT_LOOP:
                        01   E1 0C71  1932          BBC     #UETUNT$V_TESTABLE,-      ; BR if this unit is not testable
                     02 0B AB        0C73  1933                  UETUNT$B_FLAGS(R11),10$
                        59   D6 0C76  1934          INCL    R9                        ; Count testable units
                                0C78  1935 10$:
                     5B   6B C0 0C78  1936          ADDL2   (R11),R11                 ; Next unit block
                00000190'8F  5B D1 0C7B  1937          CMPL    R11,#UNIT_LIST            ; Are we full circle in the list?
                        ED   12 0C82  1938          BNEQ    UNIT_LOOP                 ; BR if not
                        59   D5 0C84  1939          TSTL    R9                        ; Any testable units?
                        12   12 0C86  1940          BNEQ    20$                       ; BR if yes...
                 0018'CF   4E 8F 90 0C88  1941          MOVB    #^A/N/,BUFFER+4           ; ...else disable the DDB record...
                                0C8E  1942          $UPDATE RAB = (R10)               ; ...here
                        4E   50 E9 0C97  1943          BLBC    R0,UPDATE_FAILED          ; If error then forget it
                                0C9A  1944 20$:
                     5B   6B C0 0C9A  1945          ADDL2   (R11),R11                 ; Next unit block
                00000190'8F  5B D1 0C9D  1946          CMPL    R11,#UNIT_LIST            ; Are we full circle in the list?
                        60   13 0CA4  1947          BEQL    END_UPDATE                ; BR if yes
                    01A6 CB  95 0CA6  1948          TSTB    DISK_B_FILE(R11)          ; Is this node for file 0 on a unit?
                        EE   12 0CAA  1949          BNEQ    20$                       ; BR if not; we'll only $GET for first node
                                0CAC  1950          $GET    RAB = (R10)               ; Get a record
                     30   50 E9 0CB5  1951          BLBC    R0,UPDATE_FAILED          ; If error then forget it
                 0014'CF   20 8A 0CB8  1952          BICB2   #LC_BITM,BUFFER           ; Convert to uppercase
                0014'CF   55 8F 91 0CBD  1953          CMPB    #^A/U/,BUFFER             ; Is it a UCB record?
                        41   12 0CC3  1954          BNEQ    END_UPDATE                ; BR if not
                        01   E0 0CC5  1955          BBS     #UETUNT$V_TESTABLE,-      ; BR if this unit is testable
                  D0 0B AB       0CC7  1956                  UETUNT$B_FLAGS(R11),20$
                58   01A7 CB  D0 0CCA  1957          MOVL    DISK_L_OTHER_PTR(R11),R8 ; See if other file on this unit...
                        05   13 0CCF  1958          BEQL    30$                       ; ...if there is one...
                        02   E0 0CD1  1959          BBS     #UETUNT$M_TESTABLE,-      ; ...might be testable
                     C4 0B A8       0CD3  1960                  UETUNT$B_FLAGS(R8),20$
                                0CD6  1961 30$:
                0018'CF   4E 8F 90 0CD6  1962          MOVB    #^A/N/,BUFFER+4           ; None testable, disable UCB record...
                                0CDC  1963          $UPDATE RAB = (R10)               ; ...here
                        B2   50 E8 0CE5  1964          BLBS    R0,20$                    ; Look at the next record if no error
                                0CE8  1965 UPDATE_FAILED:
                     0C AA   DD 0CE8  1966          PUSHL   RAB$L_STV(R10)            ; Do a simple message...
                        50   DD 0CEB  1967          PUSHL   R0                        ; ...to tell of the failure
                 01B8'CF     DF 0CED  1968          PUSHAL  INIDEV_UPDERR
                        01   DD 0CF1  1969          PUSHL   #1
                        00   EF 0CF3  1970          EXTZV   #STS$V_SEVERITY,-         ; Copy the severity from RMS status...
                   7E   50      03   0CF5  1971                  #STS$S_SEVERITY,R0,-(SP)
               6E   00741130 8F C8 0CF8  1972          BISL2   #UETP$_TEXT,(SP)          ; ...to our message
            00000000'GF      05 FB 0CFF  1973          CALLS   #5,G^LIB$SIGNAL
                                0D06  1974 END_UPDATE:
             03 0002'CF      01 E0 0D06  1975          BBS     #TEST_OVERV,FLAG,10$      ; Did the test end normally (timeout)?
                        FB58 30   0D0C  1976          BSBW    CLEAN_UP                  ; It didn't so must erase files
                                0D0F  1977 10$:
                        00   DD 0D0F  1978          PUSHL   #0                        ; Set the time flag
                 000F'CF     DF 0D11  1979          PUSHAL  TEST_NAME                 ; Push the test name
                        02   DD 0D15  1980          PUSHL   #2                        ; Push arg count
                        00   EF 0D17  1981          EXTZV   #STS$V_SEVERITY,-         ; Push the proper exit severity...
                        03   0D19  1982                  #STS$S_SEVERITY,-
                   7E   0146'CF      0D1A  1983                  STATUS,-(SP)
```

```
                6E    00741080 8F    C8  0D1E  1984          BISL2   #UETP$_ENDEDD,(SP)      ; ...and use it in our message code
                                04    DD  0D25  1985          PUSHL   #4
                                51    5E  D0  0D27  1986      MOVL    SP,R1
                                          0D2A  1987          $PUTMSG_S MSGVEC = (R1)         ; Output the message
                                          0D39  1988          $SETPRN_S PRCNAM = ACNT_NAME    ; Reset the process name
                                      04  0D44  1989          RET                             ; That's all folks!
                                          0D45  1990
                                          0D45  1991          .END    UETDISK00
```

```
$$.TAB                  = 00000BB8 R      04
$$.TABEND               = 00000C18 R      04
$$.TMP                  = 00000000
$$.TMP1                 = 00000001
$$.TMP2                 = 0000006A
$$.TMPX                 = 00000016 R      05
$$.TMPX1                = 0000000D
$$T1                    = 00000000
$$T2                    = 00000006
ACNT_NAME                 00000000 R      03
ALL_SET                   00000414 R      06
ARG_COUNT                 0000018A R      04
BAKERS                  = 00000014
BEGIN_MSGM              = 00000008
BEGIN_MSGV              = 00000003
BUFFER                    00000014 R      04
BUFFER_PTR                0000000C R      04
CCASTHAND                 00000B5F R      06
CHECK_EXTEND              00000735 R      06
CHF$L_SIGARGLST         = 00000004
CHF$L_SIG_ARG1          = 00000008
CHF$L_SIG_ARGS          = 00000000
CHF$L_SIG_NAME          = 00000004
CLEAN_UP                  00000867 R      06
CNTRLCMSG                 000000A3 R      03
COMMON                    00000ABE R      06
CONTROLLER                00000031 R      03
CONT_DESC                 000001ED R      03
CREATE_CHECK              00000935 R      06
CS1                       00000082 R      03
CS3                       00000094 R      03
DDB_RFA                   00000ACC R      04
DEAD_CTRLNAME             000000E4 R      03
DESTP_CHECK               000008F9 R      06
DEV$V_TRM               = 00000002
DEVDEP_SIZE             = 00000176
DEVDSC                    00000098 R      04
DEVNAM_LEN                00000164 R      04
DEV_NAME                  000000B7 R      04
DIB                       000000C6 R      04
DIB$B_DEVCLASS          = 00000004
DIB$B_DEVTYPE           = 00000005
DIB$K_LENGTH            = 00000074
DIB$L_MAXBLOCK          = 00000070
DIBBUF                    000000CE R      04
DISK_B_FILE               000001A6
DISK_C_NAM                000001BB
DISK_K_BUFFER             0000031C
DISK_K_CREATE           = 00000001
DISK_K_PATTERN          = 55555555
DISK_L_OTHER_PTR          000001A7
DISK_L_SIZE               000001AF
DISK_L_TOP_PCT            000001AB
DISK_M_NOCREATE         = 00000004
DISK_M_NOEXTEND         = 00000008
DISK_Q_FILEDESC           000001B3
DISK_T_NAMFILSPC          0000021B
DISK_V_NOCREATE         = 00000002
DISK_V_NOEXTEND         = 00000003
DISK_W_UNIT               000001A4
DOTTST                    00000262 R      03
DO_THE_WRITE              00000785 R      06
DUMMY_FAB                 00000B24 R      04
DUMMY_NAM                 00000BB8 R      04
DUMMY_RAB                 00000B74 R      04
DVI$_DEVNAM             = 00000020
EFN2                    = 00000004
END_UPDATE                00000D06 R      06
ERROR_COUNT               00000142 R      04
ERROR_EXIT                00000B9C R      06
EXIT_DESC                 0000017A R      04
EXIT_HANDLER              00000C1C R      06
FAB$B_BID               = 00000000
FAB$B_BLN               = 00000001
FAB$B_DNS               = 00000035
FAB$B_FNS               = 00000034
FAB$C_BID               = 00000003
FAB$C_BLN               = 00000050
FAB$C_SEQ               = 00000000
FAB$C_UDF               = 00000000
FAB$C_VAR               = 00000002
FAB$L_ALQ               = 00000010
FAB$L_CTX               = 00000018
FAB$L_DEV               = 00000040
FAB$L_DNA               = 00000030
FAB$L_FNA               = 0000002C
FAB$L_FOP               = 00000004
FAB$L_NAM               = 00000028
FAB$L_STS               = 00000008
FAB$L_STV               = 0000000C
FAB$V_BIO               = 00000005
FAB$V_CHAN_MODE         = 00000002
FAB$V_CR                = 00000001
FAB$V_FILE_MODE         = 00000004
FAB$V_GET               = 00000001
FAB$V_LNM_MODE          = 00000000
FAB$V_PUT               = 00000000
FAB$V_UFO               = 00000011
FAB$V_UPD               = 00000003
FAB$V_UPI               = 00000006
FAB$W_GBC               = 00000048
FAO_BOF                   00000004 R      04
FILE                      000001F5 R      03
FIND_IT                   00000244 R      06
FLAG                      00000002 R      04
FN_LEN                  = 00000027
FOUND_IT                  000002DC R      06
GET_PATTERN               000007E5 R      06
ILLEGAL_REC               00000151 R      03
INADDRESS                 00000152 R      04
INIDEV_UPDERR             000001B8 R      03
INI_FAB                   00000A38 R      04
INI_RAB                   00000A88 R      04
INPUT_ITMLST              00000072 R      03
```

```
IO$M_CTRLCAST           ********  X  06
IO$_SETMODE             ********  X  06
ITERATION               0000016E  R  04
LC_BITM               = 00000020
LIB$SIGNAL              ********  X  06
MAX_DEV_DESIG         = 0000000A
MAX_PROC_NAME         = 0000000F
MAX_UNIT_DESIG        = 00000005
MODE                    00000041  R  03
MSG_BLOCK               00000176  R  04
NAM$B_ESS             = 0000000A
NAM$B_NOP             = 00000008
NAM$B_RSL             = 00000003
NAM$B_RSS             = 00000002
NAM$C_BID             = 00000002
NAM$C_BLN             = 00000060
NAM$C_MAXRSS          = 000000FF
NAM$L_ESA             = 0000000C
NAM$L_RSA             = 00000004
NAME_LEN              = 0000000F
NEW_NODE                00000198  R  04
NOUNIT_SELECTED         0000012B  R  03
NO_CTRLNAME             000000C4  R  03
NO_RMS_AST_TABLE        0000004D  R  03
NRAT_LENGTH           = 00000014
ONE_SHOTM             = 00000010
ONE_SHOTV             = 00000004
ONE_SHOT_TEST           0000051E  R  06
OTS$CVT_TI_L            ********  X  06
OTS$CVT_TZ_L            ********  X  06
OUTADDRESS              0000015A  R  04
PAGES                 = 00000006
PASS                    00000172  R  04
PASS_MSG                00000185  R  03
PATTERN                 00000267  R  03
PATTERNM              = 00000020
PATTERNV              = 00000005
PATTERN_DATA            000009A0  R  04
PERCENTAGE            = CCCD3E4C
PMTSIZ                = 00000019
PRE_CLEAN               00000827  R  06
PROCESS_NAME            000000A0  R  04
PROCESS_NAME_FREE     = 0000000B
PROC_CONT_NAME          00000088  R  06
PROMPT                  00000230  R  03
QUAD_STATUS             0000014A  R  04
RAB$B_PSZ             = 00000034
RAB$B_RAC             = 0000001E
RAB$C_BID             = 00000001
RAB$C_BLN             = 00000044
RAB$C_RFA             = 00000002
RAB$C_SEQ             = 00000000
RAB$L_BKT             = 00000038
RAB$L_CTX             = 00000018
RAB$L_FAB             = 0000003C
RAB$L_PBF             = 00000030
RAB$L_RBF             = 00000028
RAB$L_ROP             = 00000004
RAB$L_STS             = 00000008
RAB$L_STV             = 0000000C
RAB$L_UBF             = 00000024
RAB$M_ASY             = 00000001
RAB$V_ASY             = 00000000
RAB$V_BIO             = 0000000B
RAB$V_PMT             = 0000001E
RAB$W_RFA             = 00000010
RAB$W_RSZ             = 00000022
RANDOM1                 00000166  R  04
RANDOM2                 0000016A  R  04
RANDOM_DATA             0000U1A0  R  04
READ_NEXT               000007FA  R  06
READ_SIZE             = 00000800
RECORD                  00000201  R  03
REC_SIZE              = 00000028
RESTART                 00000455  R  06
RMS$_BLN                ********  X  03
RMS$_BUSY               ********  X  03
RMS$_CDA                ********  X  03
RMS$_DNF                ********  X  06
RMS$_FAB                ********  X  03
RMS$_FACILITY         = 00000001
RMS$_FUL                ********  X  06
RMS$_RAB                ********  X  03
RMS_ERROR               00000A88  R  06
RMS_ERR_STRING          0000020F  R  03
SAFE_TO_UPDM          = 00000004
SAFE_TO_UPDV          = 00000002
SEC$M_EXPREG            ********  X  06
SEC$M_GBL               ********  X  06
SHR$_ABENDD           = 000010E0
SHR$_BEGIND           = 00001038
SHR$_ENDEDD           = 00001080
SHR$_OPENIN           = 00001098
SHR$_TEXT             = 00001130
SS$_BADPARAM          = 00000014
SS$_CONTROLC          = 00000651
SS$_EXDISKQUOTA       = 000003EC
SS$_NORMAL            = 00000001
SS$_NOSUCHSEC         = 00000978
SS$_SSFAIL            = 0000045C
SS$_WASSET            = 00000009
SSERROR                 000009A5  R  06
SS_SYNCH_EFN          = 00000003
STATUS                  00000146  R  04
STR$UPCASE              ********  X  06
STS$K_ERROR           = 00000002
STS$K_INFO            = 00000003
STS$K_SUCCESS         = 00000001
STS$K_WARNING         = 00000000
STS$M_INHIB_MSG       = 10000000
STS$S_FAC_NO          = 0000000C
STS$S_SEVERITY        = 00000003
STS$V_FAC_NO          = 00000010
STS$V_SEVERITY        = 00000000
```

```
SUC_EXIT                        0000058B R      06
SUPDEV_GBLSEC                   00000020 R      03
SUP_FAB                         00000AD4 R      04
SYS$ASSIGN                      ******** GX     06
SYS$CLOSE                       ******** GX     06
SYS$CONNECT                     ******** GX     06
SYS$CREATE                      ******** GX     06
SYS$CRMPSC                      ******** GX     06
SYS$DCLEXH                      ******** GX     06
SYS$ERASE                       ******** GX     06
SYS$EXIT                        ******** GX     06
SYS$EXPREG                      ******** GX     06
SYS$EXTEND                      ******** GX     06
SYS$FAO                         ******** X      06
SYS$GET                         ******** GX     06
SYS$GETDEV                      ******** GX     06
SYS$GETDVI                      ******** GX     06
SYS$GETMSG                      ******** GX     06
SYS$HIBER                       ******** GX     06
SYS$INPUT                       00000061 R      03
SYS$MGBLSC                      ******** GX     06
SYS$OPEN                        ******** GX     06
SYS$PUTMSG                      ******** GX     06
SYS$QIOW                        ******** GX     06
SYS$READ                        ******** GX     06
SYS$SETAST                      ******** GX     06
SYS$SETIMR                      ******** GX     06
SYS$SETPRN                      ******** GX     06
SYS$SETSFM                      ******** GX     06
SYS$TRNLOG                      ******** GX     06
SYS$UPDATE                      ******** GX     06
SYS$WAIT                        ******** GX     06
SYS$WAKE                        ******** GX     06
SYS$WRITE                       ******** GX     06
SYS0_SYSTEST_DIR                00000253 R      03
SYSIN_FAB                       000009A4 R      04
SYSIN_RAB                       000009F4 R      04
SYSTEST_DIR                     00000249 R      03
TEST_FILE_ERROR                 00000B05 R      06
TEST_NAME                       0000000F R      03
TEST_OVER                       000005F7 R      06
TEST_OVERM                    = 00000002
TEST_OVERV                    = 00000001
TEST_OVER_STATUS                00000600 R      06
TEXT_BUFFER                   = 00000084
THREEMIN                        000001DD R      03
TIME_IT                         00000442 R      06
TIME_OUT                        00000992 R      06
TOP_PCT                       = 00004040
TTCHAN                          00000000 R      04
UETDISK00                       00000000 RG     06
UETP                          = 00740000
UETP$_ABENDD                  = 007410E0
UETP$_ABORTC                  = 0074832B
UETP$_BEGIND                  = 00741038
UETP$_DATADEVERR              = 00748018
UETP$_DENOSU                  = 00748333
UETP$_DESTP                   = 007481A2
UETP$_ENDEDD                  = 00741080
UETP$_ERBOXPROC               = 00748020
UETP$_FACILITY                = 00000074
UETP$_OPENIN                  = 00741098
UETP$_TEXT                    = 00741130
UETUNT$B_FLAGS                = 0000000B
UETUNT$B_TYPE                 = 00000008
UETUNT$C_DEVDEP               = 000001A4
UETUNT$C_FAB                  = 00000110
UETUNT$C_INDSIZ               = 000001A4
UETUNT$K_FAB                  = 00000110
UETUNT$K_RAB                  = 00000160
UETUNT$L_ITER                 = 00000010
UETUNT$M_TESTABLE             = 00000002
UETUNT$T_FILSPC               = 00000014
UETUNT$V_TESTABLE             = 00000001
UETUNT$W_FUNC                 = 0000000E
UETUNT$W_SIZE                 = 00000009
UNIT_DESC                       000001E5 R      03
UNIT_FILE_SETUP                 0000060B R      06
UNIT_LIST                       00000190 R      04
UNIT_LOOP                       00000C71 R      06
UNIT_NUMBER                     00000162 R      04
UPDATE_FAILED                   00000CE8 R      06
VERIFY_ERROR                    000008A8 R      06
WRITE_NEXT                      000006F0 R      06
WRITE_SIZE                    = 00000800
```

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
DEVDEP_STR_DEF	00000B1C (2844.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RODATA	00000276 (630.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RWDATA	00000C18 (3096.)	04 (4.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
$RMSNAM	00000023 (35.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
DISK	00000D45 (3397.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	27	00:00:00.12	00:00:00.58
Command processing	145	00:00:00.81	00:00:04.09
Pass 1	501	00:00:23.28	00:00:51.25
Symbol table sort	0	00:00:02.25	00:00:04.58
Pass 2	353	00:00:06.20	00:00:11.75
Symbol table output	36	00:00:00.26	00:00:00.47
Psect synopsis output	3	00:00:00.03	00:00:00.04
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1065	00:00:32.97	00:01:12.76

The working set limit was 2000 pages.
126800 bytes (248 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1518 non-local and 67 local symbols.
1991 source lines were read in Pass 1, producing 38 object records in Pass 2.
65 pages of virtual memory were used to define 58 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[UETP.OBJ]UETP.MLB;1	2
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	53
TOTALS (all libraries)	55

1859 GETS were required to define 55 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:UETDISK00/OBJ=OBJ$:UETDISK00 MSRC$:UETDISK00/UPDATE=(ENH$:UETDISK00)+EXECML$/LIB+LIB$:UETP/LIB
```

SATSSF18
LIS

SATSSS09
LIS

UETCOMS00
LIS

UETDISK00
LIS

SATSSS10
LIS

UETQMPF00
LIS