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

```
UU      UU  EEEEEEEEEE  TTTTTTTTTT  MM      MM    AAAAAA    77777777      888888      000000      000000
UU      UU  EEEEEEEEEE  TTTTTTTTTT  MM      MM    AAAAAA    77777777      888888      000000      000000
UU      UU  EE              TT      MMMM  MMMM  AA      AA          77  88      88  00      00  00      00
UU      UU  EE              TT      MMMM  MMMM  AA      AA          77  88      88  00      00  00      00
UU      UU  EE              TT      MM  MM  MM  AA      AA          77  88      88  00    0000  00    0000
UU      UU  EE              TT      MM  MM  MM  AA      AA          77  88      88  00    0000  00    0000
UU      UU  EEEEEEEE        TT      MM      MM  AA      AA        77      888888    00  00  00  00  00  00
UU      UU  EEEEEEEE        TT      MM      MM  AA      AA        77      888888    00  00  00  00  00  00
UU      UU  EE              TT      MM      MM  AAAAAAAAAA      77      88      88  0000    00  0000    00
UU      UU  EE              TT      MM      MM  AAAAAAAAAA      77      88      88  0000    00  0000    00
UU      UU  EE              TT      MM      MM  AA      AA    77        88      88  00      00  00      00      ....
UU      UU  EE              TT      MM      MM  AA      AA    77        88      88  00      00  00      00      ....
UUUUUUUUUU  EEEEEEEEEE      TT      MM      MM  AA      AA  77          888888      000000      000000          ....
UUUUUUUUUU  EEEEEEEEEE      TT      MM      MM  AA      AA  77          888888      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
```

Table of contents

```
0000     1          .TITLE UETMA7800 VAX/VMS UETP DEVICE TEST FOR MA780
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  ;       This program tests shared memory unit(s) (MA780) connected to one or
0000    36  ;       more VAX processors.  Should any other processors be running the same
0000    37  ;       test concurrently, the program takes advantage of that to use the
0000    38  ;       shared memory to communicate between them.  IT IS NOT THE INTENTION OF
0000    39  ;       THIS PROGRAM TO BE SELF DOCUMENTING, ALTHOUGH CONSIDERABLE EFFORT HAS
0000    40  ;       BEEN MADE TO PROVIDE USEFUL COMMENTS.  FOR AN EXPLANATION OF why THINGS
0000    41  ;       ARE DONE THE WAY THAT THEY ARE AND HOW VARIOUS DATA STRUCTURES AND
0000    42  ;       CODE INTERACT, PLEASE READ THE FUNCTIONAL SPEC.
0000    43  ;
0000    44  ; ENVIRONMENT:
0000    45  ;       This program will run in user access mode with the exception of two
0000    46  ;       routines which run in kernal mode.  The first routine finds memory
0000    47  ;       parameters associated with a given shared memory and the second returns
0000    48  ;       the operating system version and hardware ECO levels.  ASTs are
0000    49  ;       enabled except during error processing.  This program requires the
0000    50  ;       following privileges and quotas:
0000    51  ;               CMKRNL (to read shared memory parameters)
0000    52  ;               SHMEM
0000    53  ;               SYSGBL
0000    54  ;               PRMGBL
0000    55  ;       This module has one conditional assembly, FT_NO_INTERACT, which is
0000    56  ;       described where it is defined (or left undefined).
0000    57  ;       Successful linking requires SYS$SYSTEM:SYS.STB.
```

```
0000     58 ;       When this test was written, only 11/780 CPUs had shared memory.  Since
0000     59 ;       the test relies on the ECO level from a processor register, the test
0000     60 ;       must be modified should other CPUs eventually run it.
0000     61 ;
0000     62 ;--
0000     63 ;
0000     64 ; AUTHOR: Richard N. Holstein,  CREATION DATE: August, 1981
0000     65 ;
0000     66 ; MODIFIED BY:
0000     67 ;
0000     68 ;       V03-012 RNH0011         Richard N. Holstein,    13-Jul-1984
0000     69 ;               Include our node name in global section file names to allow for
0000     70 ;               simultaneous runs on a cluster with a common SYS$TEST.
0000     71 ;
0000     72 ;       V03-011 RNH0010         Richard N. Holstein,    15-Feb-1984
0000     73 ;               Take advantage of new UETP message codes.  Fix SSERROR
0000     74 ;               interaction with RMS_ERROR.  Account for the possibility of
0000     75 ;               11/785 systems with shared memory - modify what we believe the
0000     76 ;               ECO level field to be in the PR$_SID register.
0000     77 ;
0000     78 ;       V03-010 RNH0009         Richard N. Holstein,    19-Dec-1983
0000     79 ;               Give correct sentinels to Test Controller.
0000     80 ;
0000     81 ;       V03-009 RNH0008         Richard N. Holstein,    21-Nov-1983
0000     82 ;               Use decimal conversion routine for unit numbers.
0000     83 ;
0000     84 ;       V03-008 RNH0007         Richard N. Holstein,    11-Mar-1983
0000     85 ;               Don't signal ending message in EXIT_HANDLER.
0000     86 ;
0000     87 ;       V03-007 RNH0006         Richard N. Holstein,    25-Feb-1983
0000     88 ;               Allow for longer device names.
0000     89 ;
0000     90 ;       V03-006 RNH0005         Richard N. Holstein,    22-Dec-1982
0000     91 ;               Fixes from the V3B UETP Workplan: better message if sending
0000     92 ;               interprocessor mail fails, close INTERLOCKING_TEST window,
0000     93 ;               reduce kernel mode code.
0000     94 ;
0000     95 ;       V03-005 RNH0004         Richard N. Holstein,    15-Oct-1982
0000     96 ;               Miscellaneous fixes listed in the V3B UETP Workplan.
0000     97 ;
0000     98 ;       V03-004 LDJ0004         Larry D. Jones,         15-Apr-1982
0000     99 ;               Turn off interprocessor interaction - include definition of
0000    100 ;               FT_NO_INTERACT.
0000    101 ;
0000    102 ;       V03-003 RNH0003         Richard N. Holstein,    29-Mar-1982
0000    103 ;               Turn on interprocessor interaction - remove definition of
0000    104 ;               FT_NO_INTERACT.
0000    105 ;
0000    106 ;       V03-002 RNH0002         Richard N. Holstein,    05-Feb-1982
0000    107 ;               Don't let RCVMB_AST blithely continue if it sees a mailbox
0000    108 ;               before we're ready for one.  Don't $CMKRNL to do a $SETPRV,
0000    109 ;               SYSTEST now has SETPRV privilege.
0000    110 ;
0000    111 ;       V03-001 RNH0001         Richard N. Holstein,    28-Aug-1981
0000    112 ;               Final development for Edition A, Field Test 1.
0000    113 ;
0000    114 ;**
```

```
                        0000   116          .SBTTL  Declarations
                        0000   117 ;
                        0000   118 ; INCLUDE FILES:
                        0000   119 ;
                        0000   120 ;          SYS$LIBRARY:LIB.MLB      for general definitions
                        0000   121 ;          SHRLIB$:UETP.MLB         for UETP definitions
                        0000   122
                        0000   123 ;
                        0000   124 ; MACROS:
                        0000   125 ;
                        0000   126          $ADPDEF                          ; Adapter control block
                        0000   127          $CHFDEF                          ; Condition handler frame definitions
                        0000   128          $DEVDEF                          ; Device definitions
                        0000   129          $DIBDEF                          ;  evice Information Block
                        0000   130          $DVIDEF                          ; $GETDVI ITMLST item codes
                        0000   131          $FABDEF                          ; File Access Block
                        0000   132          $JPIDEF                          ; $GETJPI ITMLST item codes
                        0000   133          $MPMDEF                          ; Multiport memory adapter reg. offsets
                        0000   134          $NAMDEF                          ; NAM block offsets
                        0000   135          $PRDEF                           ; Internal processor registers
                        0000   136          $PRVDEF                          ; Privileges
                        0000   137          $RABDEF                          ; Record Access Block
                        0000   138          $SHBDEF                          ; Shared memory control block
                        0000   139          $SHDDEF                          ; Shared memory datapage
                        0000   140          $SHRDEF                          ; Shared messages
                        0000   141          $SSDEF                           ; System Service status codes
                        0000   142          $STSDEF                          ; Status return
                        0000   143          $SYIDEF                          ; $GETSYI ITMLST item codes
                        0000   144          $UETUNTDEF                       ; UETP unit block offset definitions
                        0000   145          $UETPDEF                         ; UETP
                        0000   146
                        0000   147 .MACRO   MAILBOX_MSG_TYPES                ; Define interprocessor mailbox codes
                        0000   148          X       HELLO,10                 ; Announce me to another processor
                        0000   149          X       ICU_UCME,10              ; I see you, do you see me?
                        0000   150          X       BYEFORNOW                ; We see each other
                        0000   151          X       BADCODE                  ; You need glasses to see me
                        0000   152          X       DOSET,10                 ; Please set a common event flag
                        0000   153          X       ISSET                    ; I set the requested common event flag
                        0000   154          X       DOCLEAR,10               ; Please clear a common event flag
                        0000   155          X       ISCLEAR                  ; I cleared the requested CEF
                        0000   156          X       INTERLOCK                ; Start memory interlock test
                        0000   157          X       ME2ME,2                  ; I'm trying to talk to myself
                        0000   158          X       ME2MEDONE                ; I can talk to myself
                        0000   159          X       ILOSTU                   ; I think you went away
                        0000   160          X       BYEBYE                   ; I've gone away, this is my last action
                        0000   161 .ENDM    MAILBOX_MSG_TYPES
                        0000   162
                        0000   163 ;
                        0000   164 ; EQUATED SYMBOLS:
                        0000   165 ;
                        0000   166 ;    Facility number definitions:
00000001                0000   167          RMS$_FACILITY = 1
                        0000   168
                        0000   169 ;    SHR message definitions:
00740000                0000   170          UETP = UETP$_FACILITY@STS$V_FAC_NO ; Define the UETP facility code
007410E0                0000   171          UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
00741038                0000   172          UETP$_BEGIND = UETP!SHR$_BEGIND
```

```
00741080  0000   173          UETP$_ENDEDD = UETP!SHR$_ENDEDD
00741098  0000   174          UETP$_OPENIN = UETP!SHR$_OPENIN
00741130  0000   175          UETP$_TEXT   = UETP!SHR$_TEXT
          0000   176
          0000   177 ;    Internal flag bits...:
00000001  0000   178          TEST_OVERV   = 1                ; Set when test is over
00000002  0000   179          SAFE_TO_UPDV = 2                ; Set if it's safe to update UETINIDEV
00000003  0000   180          BEGIN_MSGV   = 3                ; Set if "BEGIN" msg has been printed
00000004  0000   181          ONE_SHOTV    = 4                ; Set if running in one shot mode
          0000   182 ;    ...and corresponding masks:
00000002  0000   183          TEST_OVERM   = 1@TEST_OVERV
00000004  0000   184          SAFE_TO_UPDM = 1@SAFE_TO_UPDV
00000008  0000   185          BEGIN_MSGM   = 1@BEGIN_MSGV
00000010  0000   186          ONE_SHOTM    = 1@ONE_SHOTV
          0000   187
          0000   188 ;    Miscellany:
00000020  0000   189          LC_BITM        = ^X20           ; Mask to convert lower case to upper
00000028  0000   190          REC_SIZE       = 40             ; UETINIDEV.DAT record size
000000C8  0000   191          TEXT_BUFFER    = 200            ; Internal text buffer size
00000004  0000   192          EFN2           = 4              ; EFN used for three minute timer
00000003  0000   193          SS_SYNCH_EFN   = 3              ; Synch miscellaneous system services
0000000F  0000   194          MAX_PROC_NAME  = 15             ; Longest possible process name
0000000A  0000   195          MAX_DEV_DESIG  = 10             ; Longest possible controller name
00000005  0000   196          MAX_UNIT_DESIG= 5               ; Longest possible unit number
          0000   197 ;
          0000   198 ; To make it easy to switch between a version of this test which communicates
          0000   199 ; between processors connected to a memory and a version which is unaware of
          0000   200 ; other processors running UETP, we have the FT_NO_INTERACT assembly switch.
          0000   201 ; Communication between processors will work when each tries to reference
          0000   202 ; objects in shared memory by the same name.  Names of objects are formed by
          0000   203 ; concatenating the memory name, the object type, the memory unit number and
          0000   204 ; the port number by which the memory sees our processor into an ASCII string.
          0000   205 ; However, the "length" of the ASCII string is affected by the FT_NO_INTERACT
          0000   206 ; assembly switch.  If the switch is defined, include that last number and
          0000   207 ; prevent interaction.  If it's not defined, don't count the last number and
          0000   208 ; thereby create names which allow objects to be shared between processors.
          0000   209 ;
          0000   210          ;FT_NO_INTERACT= 0               ; If defined, assemble code which...
          0000   211                                          ; ...blocks interprocessor testing...
          0000   212                                          ; ...by creating unique GS and CEF
          0000   213 .IF NDF FT_NO_INTERACT                   ; Preventing interprocessor knowledge?
00000002  0000   214          OTHER_CHARS    = 2              ; No, GS and CEF names are short
          0000   215 .IFF
          0000   216          OTHER_CHARS    = 3              ; Yes, names include MPM port number
          0000   217 .ENDC    ; FT_NO_INTERACT
```

```
                    0000   219 ;    Multiport memory specific definitions:
00000004            0000   220          PROCESSOR_MAX    = 4                ; Max of processors connected to memory
00003039            0000   221          COMGS_VERSION    = 12345            ; GS version for consistency (16 bits)
30390004            0000   222          CONSIS_CHECK     = COMGS_VERSION@16!PROCESSOR_MAX
00000010            0000   223          COMGS_NAME_LEN   = 15+1             ; Space in interprocessor GS for ASCIC name
00000064            0000   224          COMGS_QUEUE_MAX  = 100              ; Queue entries in interlock tests
                    0000   225 .IIF GE  COMGS_QUEUE_MAX-65536, .ERROR 0     ; COMGS_QUEUE_MAX must fit into word
00000200            0000   226          COMGS_PVT_AREA   = 512              ; Size of each processor's private area
000001F8            0000   227          COMGS_PVT_CODE   = COMGS_PVT_AREA-8 ; Space at end reserved for "mycode"
00030D40            0000   228          ADAWI_LIMIT      = 200000           ; How high ADAWI interlock test counts
000000C8            0000   229          QUEUE_LIMIT      = 200              ; How many times queues are emptied and fill
0000000A            0000   230          INTERLOCK_LIMIT  = 10               ; How many seconds we'll stall until...
                    0000   231                                             ; ...all processors have finished...
                    0000   232                                             ; ...the interlock test
                    0000   233          ;COMGS_SIZE      = <GS_K_END+511>/512 ; Size of interprocessor GS, blocks
                    0000   234                                             ; (defined after GS_K_END defined)
00002000            0000   235          WRITE_SIZE       = 8192             ; Single processor GS size in bytes
00000010            0000   236          SPRGS_SIZE       = <WRITE_SIZE+511>/512 ; Single processor GS size, blocks
0000003E            0000   237          COMGS_EFN        = 62               ; Flag whilst $UPDSEC interprocessor GS
0000003F            0000   238          SPRGS_EFN        = 63               ; Flag to indicate that $UPDSEC of...
                    0000   239                                             ; ...the single proc GS is !NOT!...
                    0000   240                                             ; ...in progress.  This must be...
                    0000   241                                             ; ...set constantly except during...
                    0000   242                                             ; ...$UPDSEC to ensure that we don't...
                    0000   243                                             ; ...hang in CLEANUP.  It must also...
                    0000   244                                             ; ...be in a local cluster
00000040            0000   245          SPREF_EFN        = 64               ; Bit 0 of single processor CEF cluster
00000060            0000   246          COMEF_EFN        = 96               ; Bit 0 of interprocessor CEF cluster
                    0000   247
                    0000   248 ;
                    0000   249 ; Exactly one of the mailbox message codes must be in each mailbox sent for
                    0000   250 ; interprocessor communication.  Codes are defined sequentially via the mailbox
                    0000   251 ; message macro, starting with 1, so that one may neatly dispatch in the
                    0000   252 ; mailbox receipt routine and catch some errors if one gets a bogus mailbox.
                    0000   253 ; Associated with some mailbox types are timeouts.  For those where they exist,
                    0000   254 ; define a number of seconds equal to that timeout.
                    0000   255 ;
                    0000   256 .MACRO  X       MBCODE,SECONDS              ; Define mailbox type codes and timeouts
                    0000   257          MSG_'MBCODE = XX
                    0000   258          MAX_MSG_TYPE = XX-1                 ; Limiting value for CASE instructions
                    0000   259          XX = XX+1
                    0000   260          .IF NB SECONDS
                    0000   261                  MBCODE'_TIMEOUT = SECONDS * -10000000
                    0000   262          .ENDC   ; NB SECONDS
                    0000   263 .ENDM   X
00000001            0000   264          XX = 1
                    0000   265          MAILBOX_MSG_TYPES                   ; Define mailbox type codes and timeouts
```

```
                     0000   267 ;
                     0000   268 ; For each memory, there will be a data structure set up, called a node.
                     0000   269 ; These nodes will be linked together in a self-relative queue whose header is
                     0000   270 ; UNIT_LIST.  The first part of each node will be the standard definition from
                     0000   271 ; $UETUNTDEF.  Following that will come the device test dependent stuff,
                     0000   272 ; defined below.  NOTE THAT THIS DEFINITION IS DONE WITH AN ABSOLUTE PSECT.
                     0000   273 ; This means that what look like declarations are really definitions and the
                     0000   274 ; labels are really just offsets into a given node on the queue.  (A not
                     0000   275 ; necessarily obvious consequence of using an ABS PSECT is that space must be
                     0000   276 ; reserved with .BLKx operations, since .BYTE, etc., attempt to store data.)
                     0000   277 ;
                     0000   278         .PSECT  DEVDEP_STR_DEF,ABS,NOEXE,NOWRT,PAGE ; Note ABS attribute!
                     0000   279
000001A4             0000   280         .BLKB   UETUNT$C_DEVDEP         ; Skip over standard UETUNT block
                     01A4   281
                     01A4   282 MA_Q_DAYTIME:                        ; Code used to ensure interprocessor...
000001AC             01A4   283         .BLKQ   1                       ; ...communication
                     01AC   284
                     01AC   285 MA_B_MYPORT:                         ; My port number on this memory
000001AD             01AC   286         .BLKB   1                       ; Since BB%%I needs a longword, allow...
000001B0             01AD   287         .BLKB   3                       ; ...filler to save much code later on
                     01B0   288
                     01B0   289 ; The following all denote error conditions and may be accessed as a single
                     01B0   290 ; block; hence there is a reliance on their being contiguous.  There is no
                     01B0   291 ; restriction on their ordering.
                     01B0   292 MA_L_ERRBLK:                         ; Start of contiguous area
                     01B0   293
                     01B0   294 MA_L_SILOSTU:                        ; Bit mask. Bit lit if memory port...
000001B4             01B0   295         .BLKL   1                       ; ...was sent an ILOSTU mailbox
                     01B4   296
                     01B4   297 MA_L_RILOSTU:                        ; Bit mask. Bit lit if memory port...
000001B8             01B4   298         .BLKL   1                       ; ...sent me an ILOSTU mailbox
                     01B8   299
                     01B8   300 MA_L_RBADCODE:                       ; Bit mask. Bit lit if memory port...
000001BC             01B8   301         .BLKL   1                       ; ...sent me a BADCODE mailbox
                     01BC   302
                     01BC   303 MA_L_SBADCODE:                       ; Bit mask. Bit lit if memory port...
000001C0             01BC   304         .BLKL   1                       ; ...was sent a BADCODE mailbox
                     01C0   305
                     01C0   306 MA_L_DOSETFAIL:                      ; Bit mask. Bit lit if memory port...
000001C4             01C0   307         .BLKL   1                       ; ...failed to set requested bit
                     01C4   308
                     01C4   309 MA_L_DOCLEARFAIL:                    ; Bit mask. Bit lit if memory port...
000001C8             01C4   310         .BLKL   1                       ; ...failed to clear requested bit
                     01C8   311
                     01C8   312 MA_L_NOCAUSE:                        ; Bit mask. Bit lit if memory port...
000001CC             01C8   313         .BLKL   1                       ; ...sent me a mailbox for no cause
                     01CC   314
                     01CC   315 MA_L_OLDSTUFF:                       ; Bit mask. Bit lit if memory port...
000001D0             01CC   316         .BLKL   1                       ; ...sent me mailbox before I was ready
                     01D0   317
                     01D0   318 MA_L_INTLKTMO:                       ; Bit mask. Bit lit if memory port...
000001D4             01D0   319         .BLKL   1                       ; ...failed to finish interlock test
                     01D4   320
                     01D4   321 MA_L_IADAWI:                         ; Bit mask. Bit lit if memory port...
000001D8             01D4   322         .BLKL   1                       ; ...produced an inconsistent ADAWI sum
                     01D8   323
```

```
                     01D8   324 MA_L_IQUEUE:                          ; Bit mask. Bit lit if memory port...
000001DC             01D8   325          .BLKL   1                    ; ...left an inconsistent queue
                     01DC   326
0000002C             01DC   327 MA_K_ERRLEN = .-MA_L_ERRBLK           ; End of items which must be contiguous
                     01DC   328
                     01DC   329 MA_L_HELLO:                           ; Bit mask. Bit lit if memory port...
000001E0             01DC   330          .BLKL   1                    ; ...was sent HELLO mailbox, need reply
                     01E0   331
                     01E0   332 MA_L_ICU_UCME:                        ; Bit mask. Bit lit if memory port...
000001E4             01E0   333          .BLKL   1                    ; ...was sent ICU_UCME MB, need reply
                     01E4   334
                     01E4   335 MA_L_DOSET:                           ; Bit mask. Bit lit if memory port...
000001E8             01E4   336          .BLKL   1                    ; ...was sent DOSET MB, need reply
                     01E8   337
                     01E8   338 MA_L_DOCLEAR:                         ; Bit mask. Bit lit if memory port...
000001EC             01E8   339          .BLKL   1                    ; ...was sent DOCLEAR MB, need reply
                     01EC   340
                     01EC   341 MA_W_UNIT:                            ; Unit number of this memory
000001EE             01EC   342          .BLKW   1                    ; To parallel code in the exec, a word
000001F0             01EE   343          .BLKW   1                    ; is allocated, but really only a byte
                     01F0   344                                       ; is used.  Another word is allocated
                     01F0   345                                       ; to allow PUSHL for system service calls
                     01F0   346
                     01F0   347 MA_T_UNIT:                            ; Holds ASCII equivalent of hex unit
000001F1             01F0   348          .BLKB   1
                     01F1   349
                     01F1   350 MA_Q_MEMNAM:                          ; Descriptor for...
000001F9             01F1   351          .BLKQ   1
                     01F9   352 MA_T_MEMNAM:                          ; ...memory name from SYSGEN
00000208             01F9   353          .BLKB   15
                     0208   354
                     0208   355 MA_Q_COMGSNAM:                        ; Descriptor for...
00000210             0208   356          .BLKQ   1
                     0210   357 MA_T_COMGSNAM:                        ; ...interprocessor global section name
0000022F             0210   358          .BLKB   15+1+15              ; shared-mem-name:global-sec-name
                     022F   359
                     022F   360 MA_Q_COMEFNAM:                        ; Descriptor for...
00000237             022F   361          .BLKQ   1
                     0237   362 MA_T_COMEFNAM:                        ; ...interprocessor common EF cluster
00000256             0237   363          .BLKB   15+1+15              ; shared-mem-name:cluster-name
                     0256   364
                     0256   365 MA_W_COMMBCHN:                        ; Channel for another processor's MB
00000258             0256   366          .BLKW   1
                     0258   367
                     0258   368 MA_Q_COMMBNAM:                        ; Skeleton of descriptor for...
00000260             0258   369          .BLKQ   1
                     0260   370 MA_T_COMMBNAM:                        ; ...another processor's mailbox
0000027F             0260   371          .BLKB   15+1+15              ; shared-mem-name:mailbox-name
                     027F   372
                     027F   373 MA_Q_SPRGSNAM:                        ; Descriptor for...
00000287             027F   374          .BLKQ   1
                     0287   375 MA_T_SPRGSNAM:                        ; ...my single processor global section
000002A6             0287   376          .BLKB   15+1+15              ; shared-mem-name:global-sec-name
                     02A6   377
                     02A6   378 MA_Q_SPREFNAM:                        ; Descriptor for...
000002AE             02A6   379          .BLKQ   1
                     02AE   380 MA_T_SPREFNAM:                        ; ...Single processor CEF cluster
```

```
000002CD  02AE   381          .BLKB   15+1+15                  ; shared-mem-name:cluster-name
          02CD   382
          02CD   383 MA_L_SPREFSTATE:                          ; Single processor CEF state vector
000002D1  02CD   384          .BLKL   1
          02D1   385
          02D1   386 MA_W_SPRMBCHN:                            ; Channel for this processor's MB
000002D3  02D1   387          .BLKW   1
000002D5  02D3   388          .BLKW   1                       ; Filler allows easier use in SS calls
          02D5   389
          02D5   390 MA_Q_SPRMBNAM:                            ; Descriptor for...
000002DD  02D5   391          .BLKQ   1
          02DD   392 MA_T_SPRMBNAM:                            ; ...this processor's mailbox
000002FC  02DD   393          .BLKB   15+1+15                 ; shared-mem-name:mailbox-name
          02FC   394
00000014  02FC   395 MA_T_COMFAB = UETUNT$T_FILSPC           ; Define these symbols as the...
00000110  02FC   396 MA_K_COMFAB = UETUNT$K_FAB              ; ...standard UETUNT symbols...
00000160  02FC   397 MA_K_COMRAB = UETUNT$K_RAB              ; ...just so all structures look alike
          02FC   398
          02FC   399 MA_T_SPRFAB:                              ; .ASCII filespec for single processor GS
000003FB  02FC   400          .BLKB   NAM$C_MAXRSS
          03FB   401
          03FB   402          .ALIGN  LONG
          03FC   403 MA_K_SPRFAB:                              ; Space for single processor section FAB
0000044C  03FC   404          .BLKB   FAB$K_BLN               ; See note where COMGSFAB is defined
          044C   405
          044C   406 MA_K_SPRRAB:                              ; Space for single processor section RAB
00000490  044C   407          .BLKB   RAB$K_BLN               ; See note where COMGSFAB is defined
          0490   408
          0490   409 MA_T_CHKFAB:                              ; .ASCII filespec to check single proc GS
0000058F  0490   410          .BLKB   NAM$C_MAXRSS
          058F   411
          058F   412          .ALIGN  LONG
          0590   413 MA_K_CHKFAB:                              ; Space for single proc section check FAB
000005E0  0590   414          .BLKB   FAB$K_BLN               ; See note where COMGSFAB is defined
          05E0   415
          05E0   416 MA_K_CHKRAB:                              ; Space for single proc section check RAB
00000624  05E0   417          .BLKB   RAB$K_BLN               ; See note where COMGSFAB is defined
          0624   418
          0624   419 MA_Q_COMGSIAD:                            ; Suggested starting address for interproc G
0000062C  0624   420          .BLKL   2
          062C   421
          062C   422 MA_Q_COMGSRAD:                            ; Actual starting address for interproc GS
00000634  062C   423          .BLKL   2
          0634   424
          0634   425 MA_Q_SPRGSIAD:                            ; Suggested starting addr for single proc GS
0000063C  0634   426          .BLKL   2
          063C   427
          063C   428 MA_Q_SPRGSRAD:                            ; Actual starting addr for single proc GS
00000644  063C   429          .BLKL   2
          0644   430
          0644   431 MA_Q_SPRGSISB:                            ; IOSB for $UPDSEC of single processor GS
0000064C  0644   432          .BLKQ   1
          064C   433
000004A8  064C   434          DEVDEP_SIZE    = .-UETUNT$C_DEVDEP ; Device dependent part of UETUNT
          064C   435
          064C   436 MA_K_SPRBUF:                              ; Buffer to store single processor GS data
0000264C  064C   437          .BLKB   WRITE_SIZE
```

```
                     264C    438
                     264C    439 MA_K_CHKBUF:                                ; Buffer to read check single processor GS
0000464C             264C    440         .BLKB   WRITE_SIZE
                     464C    441
                     464C    442         PAGES = <<UETUNT$C_INDSIZ+-         ; Add together all of the pieces...
                     464C    443                   DEVDEP_SIZE+-             ; ...which make up a UETP unit block...
                     464C    444                   WRITE_SIZE+-              ; ...to give to the $EXPREG service...
                     464C    445                   WRITE_SIZE+-              ; ...below
00000024             464C    446                   511>/512>
```

```
                        464C   448 ;
                        464C   449 ; There is a standard format for messages passed via interprocessor mailboxes.
                        464C   450 ; See the functional spec and the mailbox AST routine for summaries of what
                        464C   451 ; is actually passed.
                        464C   452 ;
                        464C   453         .PSECT  INTER_PROC_MB,ABS,NOEXE,NOWRT,PAGE ; Note ABS attribute!
                        0000   454
                        0000   455 MB_B_SENDER:                             ; Port number of sender
00000001                0000   456         .BLKB   1
                        0001   457
                        0001   458 MB_B_RECEIVER:                           ; Port number of receiver
00000002                0001   459         .BLKB   1
                        0002   460
                        0002   461 MB_B_UNIT:                               ; Shared memory unit number
00000003                0002   462         .BLKB   1
                        0003   463
                        0003   464 MB_W_WHY:                                ; Why sent (one of MSG_* codes)
00000005                0003   465         .BLKW   1
                        0005   466
                        0005   467 MB_K_MISC:                               ; Other misc info, dependent on MB_W_WHY
0000001F                0005   468         .BLKB   26
                        001F   469
                        001F   470 MB_K_END:                                ; End of mailbox
                        001F   471
                        001F   472
                        001F   473
                        001F   474
                        001F   475 ;
                        001F   476 ; Whenever a timer is associated with a mailbox, the REQIDT parameter is
                        001F   477 ; defined as:
                        001F   478 ;
                        001F   479 ;       .BYTE   shared-memory-unit-number (MA_W_UNIT)
                        001F   480 ;       .BYTE   receiver's-port-number (MB_B_RECEIVER)
                        001F   481 ;       .WORD   reason-for-sending-mail (MB_W_WHY)
                        001F   482 ;
```

```
                     001F    484 ;
                     001F    485 ; There will be an interprocessor global section accessed by all processors
                     001F    486 ; attached to a shared memory.  To get a good description of how the section
                     001F    487 ; is used, see the functional spec for this test.
                     001F    488 ;
                     001F    489         .PSECT  INTERPROC_GS_DEF,ABS,NOEXE,NOWRT,PAGE ; Note ABS attribute!
                     0000    490
                     0000    491 GS_T_SECT_NAME:                          ; ASCIC global section name
00000010             0000    492         .BLKB   COMGS_NAME_LEN           ; Use this for consistency checking
                     0010    493
                     0010    494 GS_L_VERSION:                            ; UETMA7800 GS version, max ports
00000014             0010    495         .BLKL   1                        ; (Interprocessor consistency check)
                     0014    496
                     0014    497 GS_Q_SYSGQVERSION:                       ; VMS .ASCII version number
0000001C             0014    498         .BLKQ   1
                     001C    499
                     001C    500 GS_L_ECOLEVEL:                           ; Hardware ECO level of owner CPU
00000020             001C    501         .BLKL   1
                     0020    502
                     0020    503 GS_L_OWNER:                              ; Port number of creating processor
00000024             0020    504         .BLKL   1
                     0024    505
                     0024    506 GS_L_AVAILABLE:                          ; Bit mask.  Bit lit if memory port...
00000028             0024    507         .BLKL   1                        ; ...has cooperating processor
                     0028    508
                     0028    509 GS_L_CEF_FLAG:                           ; Flag which marks interprocessor...
0000002C             0028    510         .BLKL   1                        ; CEF test in progress (flag and port)
                     002C    511
                     002C    512 GS_L_PARTICIPATING:                      ; Bit mask of procs starting interlock test
00000030             002C    513         .BLKL   1                        ; Must immediately preceed GS_L_FINISHED
                     0030    51.
                     0030    515 GS_L_FINISHED:                           ; Bit mask of procs finishing interlock test
00000034             0030    516         .BLKL   1                        ; Must immediately follow GS_L_PARTICIPATING
                     0034    517
                     0034    518 GS_L_INTLK_FLAG:                         ; Flag which marks interlocking...
00000038             0034    519         .BLKL   1                        ; ...test in progress (flag and port)
                     0038    520
                     0038    521         .ALIGN  WORD                     ; ADAWI must be word aligned
                     0038    522 GS_L_ADAWI_COM:                          ; Common ADAWI count word - grand total
0000003C             0038    523         .BLKW   1+1                      ; Includes overflow word, as well
                     003C    524
                     003C    525 GS_K_ADAWI_PVT:                          ; Private ADAWI count per processor
0000004C             003C    526         .BLKW   PROCESSOR_MAX*<1+1>      ; ADAWI, overflow. Adjacent to ADAWI_COM
                     004C    527
                     004C    528 GS_K_Q_COUNT:                            ; Per processor count of "busy"...
00000054             004C    529         .BLKW   PROCESSOR_MAX            ; ...queue entries
                     0054    530
                     0054    531         .ALIGN  QUAD
                     0058    532 GS_Q_FREE:                               ; "Free" queue header
00000060             0058    533         .BLKQ   1                        ; Keep adjacent to "busy" queue header
                     0060    534
                     0060    535 GS_Q_BUSY:                               ; "Busy" queue header
00000068             0060    536         .BLKQ   1                        ; Keep adjacent to queue entries
                     0068    537
                     0068    538 GS_K_Q_ENTRIES:                          ; Busy and free queue entries
000006A8             0068    539         .BLKL   COMGS_QUEUE_MAX*<1+1+1+1> ; Flink, blink, port number, filler
                     06A8    540
```

```
                06A8    541 GS_K_PVT_AREAS:                                     ; Private areas for each processor. Each...
00000EA8        06A8    542         .BLKB   PROCESSOR_MAX*COMGS_PVT_AREA ; ...last quadword gets MA_Q_DAYTIME
                0EA8    543
                0EA8    544 GS_K_END:                                           ; End of global section
00000008        0EA8    545 COMGS_SIZE = <GS_K_END+511>/512                     ; Size of interprocessor GS, blocks
```

```
                                0EA8    547          .SBTTL  Read-Only Data
                            00000000    548          .PSECT  RODATA,NOEXE,NOWRT,PAGE
                                0000    549
                                0000    550 ACNT_NAME:                              ; Process name on exit
53 45 54 53 59 53 00000008'010E0000'    551          .ASCID  /SYSTEST/
                                  54    000E
                                000F    552
                                000F    553 TEST_NAME:                              ; This test name
37 41 4D 54 45 55 00000017'010E0000'    554          .ASCID  /UETMA7800/
                             30 30 38   001D
                                0020    555
                                0020    556 SUPDEV_GBLSEC:                          ; How we access UETSUPDEV.DAT
50 55 53 54 45 55 00000028'010E0000'    557          .ASCID  /UETSUPDEV/
                             56 45 44   002E
                                0031    558
                                0031    559 CONTROLLER:                             ; Logical name of controller
41 4E 4C 52 54 43 00000039'010E0000'    560          .ASCID  /CTRLNAME/
                                 45 4D  003F
                                0041    561
                                0041    562 MODE:                                   ; Run mode logical name
      45 44 4F 4D 00000049'010E0000'    563          .ASCID  /MODE/
                                004D    564
                                004D    565 NO_RMS_AST_TABLE:                       ; List of errors for which...
                        00000000'       566          .LONG   RMS$_BLN               ; ...RMS cannot deliver an AST...
                        00000000'       567          .LONG   RMS$_BUSY              ; ...even if one has an ERR= arg
                        00000000'       568          .LONG   RMS$_CDA               ; Note that we can search table...
                        00000000'       569          .LONG   RMS$_FAB               ; ...via MATCHC since <31:16>...
                        00000000'       570          .LONG   RMS$_RAB               ; ...pattern can't be in <15:0>
                        00000014        571 NRAT_LENGTH = .-NO_RMS_AST_TABLE
                                0061    572
                                0061    573 SYS$INPUT:                              ; Name of device from which...
4E 49 24 53 59 53 00000069'010E0000'    574          .ASCID  /SYS$INPUT/            ; ...the test can be aborted
                             54 55 50   006F
                                0072    575
                                0072    576 INPUT_ITMLST:                           ; $GETDVI arg list for SYS$INPUT
                          0020 0040     577          .WORD   64,DVI$_DEVNAM         ; We need the equivalence name
                    0000000C'00000014'  0076    578          .LONG   BUFFER,BUFFER_PTR
                          00000000      007E    579          .LONG   0              ; Terminate the list
                                0082    580
                                0082    581 CNTRLCMSG:
65 74 72 6F 62 41 0000008A'010E0000'    582          .ASCID  \Aborted via a user CTRL/C\
72 65 73 75 20 61 20 61 69 76 20 64     0090
                  43 2F 4C 52 54 43 20  009C
                                00A3    583
                                00A3    584 NO_CTRLNAME:
6E 6F 63 20 6F 4E 000000AB'010E0000'    585          .ASCID  /No controller specified./
63 65 70 73 20 72 65 6C 6C 6F 72 74     00B1
                     2E 64 65 69 66 69  00BD
                                00C3    586
                                00C3    587 DEAD_CTRLNAME:
20 74 27 6E 61 43 000000CB'010E0000'    588          .ASCID  /Can't test controller !AS, marked as unusable in UETINIDEV.DAT./
6C 6F 72 74 6E 6F 63 20 74 73 65 74     00D1
72 61 6D 20 2C 53 41 21 20 72 65 6C     00DD
61 73 75 6E 75 20 73 61 20 64 65 6B     00E9
4E 49 54 45 55 20 6E 69 20 65 6C 62     00F5
         2E 54 41 44 2E 56 45 44 49     0101
                                010A    589
```

```
                                                010A   590 NOUNIT_SELECTED:
69 6E 75 20 6F 4E 00000112'010E0000'            010A   591         .ASCID  /No units selected for testing./
20 64 65 74 63 65 6C 65 73 20 73 74             0118
2E 67 6E 69 74 73 65 74 20 72 6F 66             0124
                                                0130   592
                                                0130   593 ILLEGAL_REC:
61 67 65 6C 6C 49 00000138'010E0000'            0130   594         .ASCID  /Illegal record format in file UETINIDEV.DAT!/
72 6F 66 20 64 72 6F 63 65 72 20 6C             013E
20 65 6C 69 66 20 6E 69 20 74 61 6D             014A
41 44 2E 56 45 44 49 4E 49 54 45 55             0156
                                  21 54         0162
                                                0164   595
                                                0164   596 PASS_MSG:
66 6F 20 64 6E 45 0000016C'010E0000'            0164   597         .ASCID  /End of pass !UL with !UL iterations at !%D./
69 77 20 4C 55 21 20 73 73 61 70 20             0172
61 72 65 74 69 20 4C 55 21 20 68 74             017E
44 25 21 20 74 61 20 73 6E 6F 69 74             018A
                                  2E            0196
                                                0197   598
                                                0197   599 INIDEV_UPDERR:                      ; Error during exit handler
20 72 6F 72 72 45 0000019F'010E0000'            0197   600         .ASCID  /Error updating UETINIDEV.DAT./
54 45 55 20 67 6E 69 74 61 64 70 75             01A5
   2E 54 41 44 2E 56 45 44 49 4E 49             01B1
                                                01BC   601
                                                01BC   602 THREEMIN:                            ; 3 minute delta time
                  FFFFFFFF 94B62E00             01BC   603         .LONG   -10*1000*1000*180,-1
                                                01C4   604
                                                01C4   605 UNIT_DESC:                           ; Descriptor used to convert unit #
                           00000005             01C4   606         .LONG   5
                           0000001A'            01C8   607         .ADDRESS BUFFER+6
                                                01CC   608
                                                01CC   609 CONT_DESC:                           ; Descriptor used to convert controller...
                         0000 0028              01CC   610         .WORD   REC_SIZE,0           ; ...from lowercase to uppercase
                           00000014'            01D0   611         .ADDRESS BUFFER
                                                01D4   612
                                                01D4   613 FILE:                                ; Fills in RMS_ERR_STRING
      65 6C 69 66 000001DC'010E0000'            01D4   614         .ASCID  /file/
                                                01E0   615
                                                01E0   616 RECORD:                              ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 000001E8'010E0000'            01E0   617         .ASCID  /record/
                                                01EE   618
                                                01EE   619 RMS_ERR_STRING:                      ; Announces an RMS error
41 21 20 53 4D 52 000001F6'010E0000'            01FE   620         .ASCID  /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53             01FC
                  44 41 21 20 65 6C 69          0208
                                                020F   621
                                                020F   622 PROMPT:
64 20 72 65 6C 6F 72 74 6E 6F 43                020F   623         .ASCII  /Controller designation?: /
3A 3F 6E 6F 69 74 61 6E 67 69 73 65             021B
                                  20            0227
                           00000019             0228   624         PMTSIZ = .-PROMPT
                                                0228   625
                                                0228   626 NEEDED_PRIVS:                       ; Arglist for $SETPRV
                           0B000000             0228   627         .LONG   <1@PRV$V_SHMEM>!<1@PRV$V_PRMGBL>!<1@PRV$V_SYSGBL>
                           00000000             022C   628         .LONG   0
                                                0230   629
                                                0230   630 GETSYI_ITMLST:                      ; Return info about our processor
```

```
                     1001 0004  0230   631          .WORD   4,SYI$_SID                  ; Read PR$_SID for me
                      00000224' 0234   632          .ADDRESS COMGS_BUF+GS_L_ECOLEVEL
                      00000000  0238   633          .LONG   0
                     1000 0008  023C   634          .WORD   8,SYI$_VERSION              ; What version of VMS is running?
                      0000021C' 0240   635          .ADDRESS COMGS_BUF+GS_Q_SYSGQVERSION
                      00000000  0244   636          .LONG   0
                     1067 0006  0248   637          .WORD   6,SYI$_SCSNODE              ; What's our cluster node name?
                      000001F4' 024C   638          .ADDRESS SCSNODE
                      000001FA' 0250   639          .ADDRESS SCSNODE_LENGTH
                      00000000  0254   640          .LONG   0                           ; End of list of processor info items
                                0258   641
                                0258   642 GETJPI_ITMLST:                              ; Return timer request info
                     0315 0004  0258   643          .WORD   4,JPI$_TQCNT
                      0000114B' 025C   644          .ADDRESS TQCNT
                00000000 00000000  0260   645          .LONG   0,0
                                0268   646
                                0268   647 MPM_LITERAL:                               ; Literal string needed to check that...
        4D 50 4D 00000270'010E0000'  0268   648          .ASCID  /MPM/                       ; ...we're really given a shared memory
                                0273   649
                                0273   650 CS2:                                       ; No device type, any device class...
20 2A 2A 20 20 20 0000027B'010E0000'  0273   651          .ASCID  /   ** /                    ; ...to check UETSUPDEV.DAT
                                0281   652
                                0281   653 UETCOMGS:                                 ; Constant part of interproc GS name
4D 4F 43 54 45 55 00000289'010E0000'  0281   654          .ASCID  /UETCOMGS/                  ; Must be able to accomodate unit
                          53 47  028F
                                0291   655
                                0291   656 UETSPRGS:                                 ; Constant part of single processor GS name
52 50 53 54 45 55 00000299'010E0000'  0291   657          .ASCID  /UETSPRGS/                  ; Must be able to accomodate unit
                          53 47  029F
                                02A1   658
                                02A1   659 UETMB:                                    ; Constant part of mailbox names
   42 4D 54 45 55 000002A9'010E0000'  02A1   660          .ASCID  /UETMB/                     ; Must be able to accomodate unit and port
                                02AE   661
                                02AE   662 UETCOMEF:                                 ; Constant part of interproc common EF
4D 4F 43 54 45 55 000002B6'010E0000'  02AE   663          .ASCID  /UETCOMEF/                  ; Must be able to accomodate unit
                          46 45  02BC
                                02BE   664
                                02BE   665 UETSPREF:                                 ; Constant part of single processor EF
52 50 53 54 45 55 000002C6'010E0000'  02BE   666          .ASCID  /UETSPREF/                  ; Must be able to accomodate unit
                          46 45  02CC
                                02CE   667
                                02CE   668 SYS$TEST:                                 ; Logical name for [SYSTEST] on system disk
45 54 24 53 59 53 000002D6'010E0000'  02CE   669          .ASCID  /SYS$TEST:/
                       3A 54 53  02DC
                                02DF   670
                                02DF   671 DOTDAT:                                   ; File type of files used for disk sections
       54 41 44 2E 000002E7'010E0000'  02DF   672          .ASCID  /.DAT/
                                02EB   673
                                02EB   674 ONE_SECOND:                               ; 1-second delta time
                   FFFFFFFF FF676980  02EB   675          .LONG   -10000000*1,-1
                                02F3   676
                                02F3   677 ALREADY_HERE:                             ; Possible consistency error message
61 63 69 64 6E 49 000002FB'010E0000'  02F3   678          .ASCID  \Indications are that this processor was already connected\-
68 74 20 65 72 61 20 73 6E 6F 69 74   0301
63 6F 72 70 20 73 69 68 74 20 74 61   030D
6C 61 20 73 61 77 20 72 6F 73 73 65   0319
63 65 6E 6E 6F 63 20 79 64 61 65 72   0325
```

```
                                                64 65 74  0331
6E 69 20 65 68 74 20 6F 74 09 2F 21  0334   679                 \!/     to the interprocessor global section for !AS, MPM:ZB.\
72 6F 73 73 65 63 6F 72 70 72 65 74  0340
74 63 65 73 20 6C 61 62 6F 6C 67 20  034C
2C 53 41 21 20 72 6F 66 20 6E 6F 69  0358
                2E 42 5A 21 4D 50 4D 20  0364
                                                036C   680
                                                036C   681 INCONSISTENCY:                                ; We got the wrong interprocessor GS
64 65 70 70 61 4D 00000374'010E0000'  036C   682         .ASCID  /Mapped to wrong interprocessor global section for !AS, MPM!ZB./
6E 69 20 67 6E 6F 72 77 20 6F 74 20  037A
72 6F 73 73 65 63 6F 72 70 72 65 74  0386
74 63 65 73 20 6C 61 62 6F 6C 67 20  0392
2C 53 41 21 20 72 6F 66 20 6E 6F 69  03AE
                2E 42 5A 21 4D 50 4D 20  03AA
                                                03B2   683
                                                03B2   684 OLD_ONEPROC:                                 ; We found an existing single processor GS
64 65 70 70 61 4D 000003BA'010E0000'  03B2   685         .ASCID  \Mapped to an existing single processor global section\-
74 73 69 78 65 20 6E 61 20 6F 74 20  03C0
70 20 65 6C 67 6E 69 73 20 67 6E 69  03CC
6F 6C 67 20 72 6F 73 73 65 63 6F 72  03D8
   6E 6F 69 74 63 65 73 20 6C 61 62  03E4
20 2C 53 41 21 20 72 6F 66 09 2F 21  03EF   686                 \!/     for !AS, MPM!ZB.\
                2E 42 5A 21 4D 50 4D  03FB
                                                0402   687
                                                0402   688 BAD_MOVC3:                                   ; Couldn't write to single processor GS
6E 69 74 69 72 57 0000040A'010E0000'  0402   689         .ASCID  \Writing to single processor global section in memory\-
20 65 6C 67 6E 69 73 20 6F 74 20 67  0410
6C 67 20 72 6F 73 73 65 63 6F 72 70  041C
6E 6F 69 74 63 65 73 20 6C 61 62 6F  0428
      74 72 6F 6D 65 6D 20 6E 69 20  0434
72 6F 63 6E 69 20 74 6F 67 09 0A 0D  043E   690                 <13><10>\       got incorrect results.\
73 74 6C 75 73 65 72 20 74 63 65 72  044A
                                2E  0456
                                                0457   691
                                                0457   692 BAD_UPDATE:                                  ; Couldn't update single processor GS
69 74 61 64 70 55 0000045F'010E0000'  0457   693         .ASCID  \Updating single processor global section to disk\-
72 70 20 65 6C 67 6E 69 73 20 67 6E  0465
62 6F 6C 67 20 72 6F 73 73 65 63 6F  0471
74 20 6E 6F 69 74 63 65 73 20 6C 61  047D
                6B 73 69 64 20 6F  0489
72 6F 63 6E 69 20 74 6F 67 09 0A 0D  048F   694                 <13><10>\       got incorrect results.\
73 74 6C 75 73 65 72 20 74 63 65 72  049B
                                2E  04A7
                                                04A8   695
                                                04A8   696 UPDSEC_HWE:                                  ; Hardware write error during $UPDSEC
61 77 64 72 61 48 000004B0'010E0000'  04A8   697         .ASCID  /Hardware write error during $UPDSEC./
72 72 65 20 65 74 69 72 77 20 65 72  04B6
55 24 20 67 6E 69 72 75 64 20 72 6F  04C2
                2E 43 45 53 44 50  04CE
                                                04D4   698
                                                04D4   699 UPDSEC_FAILED:                               ; Miscellaneous error during $UPDSEC
45 53 44 50 55 24 000004DC'010E0000'  04D4   700         .ASCID  \$UPDSEC failed for !AS, MPM!ZB.\-
72 6F 66 20 64 65 6C 69 61 66 20 43  04E2
42 5A 21 4D 50 4D 20 2C 53 41 21 20  04EE
                                2E  04FA
61 20 6C 61 75 74 72 69 56 09 2F 21  04FB   701                 \!/     Virtual address of first page not written was: !XL.\
69 66 20 66 6F 20 73 73 65 72 64 64  0507
```

```
74 6F 6E 20 65 67 61 70 20 74 73 72  0513
73 61 77 20 6E 65 74 74 69 72 77 20  051F
                  2E 4C 58 21 20 3A  052B
                                     0531    702
                                     0531    703 ME2ME_FAILED:                       ; Could not send myself some mail
20 64 6C 75 6F 43 00000539'010E0000' 0531    704         .ASCID  \Could not send mail to myself for !AS, MPM.ZB.\
69 61 6D 20 64 6E 65 73 20 74 6F 6E  053F
20 66 6C 65 73 79 6D 20 6F 74 20 6C  054B
4D 50 4D 20 2C 53 41 21 20 72 6F 66  0557
                  2E 42 5A 21        0563
                                     0567    705
                                     0567    706 BAD_CEF:                            ; Wrong results with common event flags
65 6C 67 6E 69 53 0000056F'010E0000' 0567    707         .ASCID  \Single processor common event flag failed for .AS, MPM!ZB:\-
63 20 72 6F 73 73 65 63 6F 72 70 20  0575
20 74 6E 65 76 65 20 6E 6F 6D 6D 6F  0581
20 64 65 6C 69 61 66 20 67 61 6C 66  058D
4D 50 4D 20 2C 53 41 21 20 72 6F 66  0599
                  3A 42 5A 21        05A5
20 64 65 74 63 65 70 78 45 09 2F 21  05A9    708                 \!/     Expected flags = !XL, returned flags = !XL, XOR = !XL.\
2C 4C 58 21 20 3D 20 73 67 61 6C 66  05B5
6C 66 20 64 65 6E 72 75 74 65 72 20  05C1
58 20 2C 4C 58 21 20 3D 20 73 67 61  05CD
         2E 4C 58 21 20 3D 20 52 4F  05D9
                                     05E2    709
                                     05E2    710 BAD_MSG_TYPE:                       ; Inconsistent interprocessor mailbox
73 6E 6F 63 6E 49 000005EA'010E0000' 05E2    711         .ASCID  \Inconsistent or otherwise obviously wrong information in\-
74 6F 20 72 6F 20 74 6E 65 74 73 69  05F0
69 76 62 6F 20 65 73 69 77 72 65 68  05FC
20 67 6E 6F 72 77 20 79 6C 73 75 6F  0608
20 6E 6F 69 74 61 6D 72 6F 66 6E 69  0614
                              6E 69  0620
63 6F 72 70 72 65 74 6E 69 09 2F 21  0622    712                     \!/     interprocessor mailbox for !AS, MPM!ZB.\
6F 62 6C 69 61 6D 20 72 6F 73 73 65  062E
4D 20 2C 53 41 21 20 72 6F 66 20 78  063A
                  2E 42 5A 21 4D 50  0646
                                     064C    713
                                     064C    714 BAD_MB_QIO:                         ; IOSB tells the story of $QIO gone bad
6F 72 72 45 5F 21 00000654'010E0000' 064C    715         .ASCID \!_Error writing to interprocessor mailbox.!/\-
6F 74 20 67 6E 69 74 69 72 77 20 72  065A
73 65 63 6F 72 70 72 65 74 6E 69 20  0666
2E 78 6F 62 6C 69 61 6D 20 72 6F 73  0672
                              2F 21  067E
69 6E 55 20 79 72 6F 6D 65 4D 5F 21  0680    716         \!_Memory Unit: !XB, Receiver's port: !XB, Message Type: !XW,\-
65 63 65 52 20 2C 42 58 21 20 3A 74  068C
3A 74 72 6F 70 20 73 27 72 65 76 69  0698
67 61 73 73 65 4D 20 2C 42 58 21 20  06A4
2C 57 58 21 20 3A 65 70 79 54 20 65  06B0
74 73 20 6C 61 6E 69 46 5F 21 2F 21  06BC    717         \!/!_Final status: !XL.\
      2E 4C 58 21 20 3A 73 .5 74 61  06C8
                                     06D2    718
                                     06D2    719 BAD_REQIDT:                         ; Timeout for MB got phoney AST
20 72 65 6D 69 54 000006DA'010E0000' 06D2    720         .ASCID  /Timer request for a mailbox resulted in an inconsistent/-
20 72 6F 66 20 74 73 65 75 71 65 72  06E0
65 72 20 78 6F 62 6C 69 61 6D 20 61  06EC
6E 61 20 6E 69 20 64 65 74 6C 75 73  06F8
6E 65 74 73 69 7' 6E 6F 63 6E 69 20  0704
                                 74  0710
```

```
2E 54 44 49 51 45 52 20 54 53 41 20  0711   721                 / AST REQIDT./
                                     071D   722
                                     071D   723 BAD_ECO:                                ; My CPU differs from yours
43 45 20 55 50 43 00000725'010E0000' 071D   724         .ASCID  /CPU ECO level differs from another CPU's for !AS, MPM.ZB./
66 66 69 64 20 6C 65 76 65 6C 20 4F  072B
6F 6E 61 20 6D 6F 72 66 20 73 72 65  0737
66 20 73 27 55 50 43 20 72 65 68 74  0743
21 4D 50 4D 20 2C 53 41 21 20 72 6F  074F
                  2E 42 5A           075B
                                     075E   725
                                     075E   726 WRONG_VERSION:                            ; My VMS differs from yours
6F 69 73 72 65 56 00000766'010E0000' 075E   727         .ASCID  /Version of VMS differs from another CPU's for !AS, MPM!ZB./
66 69 64 20 53 4D 56 20 66 6F 20 6E  076C
6E 61 20 6D 6F 72 66 20 73 72 65 66  0778
20 73 27 55 50 43 20 72 65 68 74 6F  0784
4D 50 4D 20 2C 53 41 21 20 72 6F 66  0790
                  2E 42 5A 21        079C
                                     07A0   728
                                     07A0   729 SUMMARY_HEADER:                          ; First line when summarizing errors
72 61 6D 6D 75 53 000007A8'010E0000' 07A0   730         .ASCID  \Summary of recoverable errors on !AS, MPM!ZB:\
72 65 76 6F 63 65 72 20 66 6F 20 79  07AE
20 73 72 6F 72 72 65 20 65 6C 62 61  07BA
21 4D 50 4D 20 2C 53 41 21 20 6E 6F  07C6
                  3A 42 5A           07D2
                                     07D5   731
                                     07D5   732 RILOSTU_MSG:                            ; Processors who think we went away
73 28 55 50 43 09 000007DD'010E0000' 07D5   733         .ASCID  \       CPU(s) on port(s) !#(3UB) timed out before this\-
29 73 28 74 72 6F 70 20 6E 6F 20 29  07E3
6D 69 74 20 29 42 55 33 28 23 21 20  07EF
72 6F 66 65 62 20 74 75 6F 20 64 65  07FB
                  73 69 68 74 20 65  0807
6F 73 73 65 63 6F 72 70 09 09 2F 21  080D   734                 \!/             processor could send a reply mailbox.\
64 6E 65 73 20 64 6C 75 6F 63 20 72  0819
69 61 6D 20 79 6C 70 65 72 20 61 20  0825
                  2E 78 6F 62 6C     0831
                                     0836   735
                                     0836   736 SILOSTU_MSG:                            ; Processors we think went away
73 28 55 50 43 09 0000083E'010E0000' 0836   737         .ASCID  \       CPU(s) on port(s) !#(3UB) didn't reply to some\-
29 73 28 74 72 6F 70 20 6E 6F 20 29  0844
64 69 64 20 29 42 55 33 28 23 21 20  0850
6F 74 20 79 6C 70 65 72 20 74 27 6E  085C
                  65 6D 6F 73 20     0868
20 78 6F 62 6C 69 61 6D 09 09 2F 21  086D   738                 \!/             mailbox this processor sent.\
73 73 65 63 6F 72 70 20 73 69 68 74  0879
               2E 74 6E 65 73 20 72 6F  0885
                                     088D   739
                                     088D   740 RBADCODE_MSG:                           ; Procs whose date-time we read wrong
73 28 55 50 43 09 00000895'010E0000' 088D   741         .ASCID  \       CPU(s) on port(s) !#(3UB) wrote different info to the\-
29 73 28 74 72 6F 70 20 6E 6F 20 29  089B
6F 72 77 20 29 42 55 33 28 23 21 20  08A7
74 6E 65 72 65 66 66 69 64 20 65 74  08B3
65 68 74 20 6F 74 20 6F 66 6E 69 20  08BF
6F 72 70 72 65 74 6E 69 09 09 2F 21  08CB   742                 \!/             interprocessor global section than what\-
61 62 6F 6C 67 20 72 6F 73 73 65 63  08D7
68 74 20 6E 6F 69 74 63 65 73 20 6F  08E3
               74 61 68 77 20 6E 61  08EF
65 72 20 55 50 43 20 73 69 68 74 20  08F6   743                 \ this CPU read.\
```

```
                                                2E 64 61  0902
                                                          0905   744
                                                          0905   745 SBADCODE_MSG:                       ; Procs who read our date-time wrong
73 28 55 50 43 09 0000090D'010E0000'              0905   746         .ASCID  \       CPU(s) on port(s) !#(3UB) couldn't read what this\-
29 73 28 74 72 6F 70 20 6E 6F 20 29               0913
75 6F 63 20 29 42 55 33 28 23 21 20               091F
77 20 64 61 65 72 20 74 27 6E 64 6C               092B
                  73 69 68 74 20 74 61 68         0937
6F 73 73 65 63 6F 72 70 09 09 2F 21               093F   747                 \!/             processor wrote to the interprocessor global\-
74 20 6F 74 20 65 74 6F 72 77 20 72               094B
63 6F 72 70 72 65 74 6E 69 20 65 68               0957
6C 61 62 6F 6C 67 20 72 6F 73 73 65               0963
         2E 6E 6F 69 74 63 65 73 20               096F   748                 \ section.\
                                                0978   749
                                                0978   750 DOSET_MSG:                                ; Processors which didn't set a CEF
73 28 55 50 43 09 00000980'010E0000'              0978   751         .ASCID  \       CPU(s) on port(s) !#(3UB) weren't able to set a\-
29 73 28 74 72 6F 70 20 6E 6F 20 29               0986
72 65 77 20 29 42 55 33 28 23 21 20               0992
6F 74 20 65 6C 62 61 20 74 27 6E 65               099E
                  61 20 74 65 73 20               09AA
6E 65 76 65 20 6E 6F 6D 6D 6F 63 20               09B0   752                 \ common event flag.\
                  2E 67 61 6C 66 20 74            09BC
                                                09C3   753
                                                09C3   754 DOCLEAR_MSG:                              ; Processors which didn't clear a CEF
73 28 55 50 43 09 000009CB'010E0000'              09C3   755         .ASCID  \       CPU(s) on port(s) !#(3UB) weren't able to clear a\-
29 73 28 74 72 6F 70 20 6E 6F 20 29               09D1
72 65 77 20 29 42 55 33 28 23 21 20               09DD
6F 74 20 65 6C 62 61 20 74 27 6E 65               09E9
                  61 20 72 61 65 6C 63 20         09F5
6E 65 76 65 20 6E 6F 6D 6D 6F 63 20               09FD   756                 \ common event flag.\
                  2E 67 61 6C 66 20 74            0A09
                                                0A10   757
                                                0A10   758 NOCAUSE_MSG:                              ; Msg from processors for no cause
73 28 55 50 43 09 00000A18'010E0000'              0A10   759         .ASCID  \       CPU(s) on port(s) !#(3UB) sent this processor a\-
29 73 28 74 72 6F 70 20 6E 6F 20 29               0A1E
6E 65 73 20 29 42 55 33 28 23 21 20               0A2A
65 63 6F 72 70 20 73 69 68 74 20 74               0A36
                  61 20 72 6F 73 73               0A42
20 78 6F 62 6C 69 61 6D 09 09 2F 21               0A48   760                 \!/             mailbox message we were not expecting.\
77 20 65 77 20 65 67 61 73 73 65 6D               0A54
65 70 78 65 20 74 6F 6E 20 65 72 65               0A60
                  2E 67 6E 69 74 63               0A6C
                                                0A72   761
                                                0A72   762 OLDSTUFF_MSG:                            ; Msg from processor before we're ready
73 28 55 50 43 09 00000A7A'010E0000'              0A72   763         .ASCID  \       CPU(s) on port(s) !#(3UB) sent this processor a\-
29 73 28 74 72 6F 70 20 6E 6F 20 29               0A80
6E 65 73 20 29 42 55 33 28 23 21 20               0A8C
65 63 6F 72 70 20 73 69 68 74 20 74               0A98
                  61 20 20 6F 73 73               0AA4
20 78 6F 62 6C 69 61 6D 09 09 2F 21               0AAA   764                 \!/             mailbox message before we announced we were\-
6F 66 65 62 20 65 67 61 73 73 65 6D               0AB6
6E 75 6F 6E 6E 61 20 65 77 20 65 72               0AC2
   65 72 65 77 20 65 77 20 64 65 63               0ACE
6F 74 09 09 2F 21 79 64 61 65 72 20               0AD9   765                 \ ready!/               to receive it.\
2E 74 69 20 65 76 69 65 63 65 72 20               0AE5
                                                0AF1   766
                                                0AF1   767 INTLKTMO_MSG:                             ; GS_L_FINISHED .NE. GS_L_PARTICIPATING
```

```
                                                                                
73 28 55 50 43 09 00000AF9'010E0000' 0AF1    768          .ASCID  \       CPU(s) on port(s) !#(3UB) did not indicate that they\-
29 73 28 74 72 6F 70 20 6E 6F 20 29  0AFF
64 69 64 20 29 42 55 33 28 23 21 20  0B0B
74 61 63 69 64 6E 69 20 74 6F 6E 20  0B17
   79 65 68 74 20 74 61 68 74 20 65  0B23
64 65 68 73 69 6E 69 66 09 09 6F 21  0B2E    769                  \!/             finished the memory interlock test.\
20 79 72 6F 6D 65 6D 20 65 68 74 20  0B3A
65 74 20 6B 63 6F 6C 72 6F 74 6E 69  0B46
                     2E 74 73        0B52
                                     0B55    770
                                     0B55    771 IADAWI_MSG:                 ; Inconsistent ADAWI sums
73 28 55 50 43 09 00000B5D'010E0000' 0B55    772          .ASCID  \       CPU(s) on port(s) !#(3UB) did not produce consistent\-
29 73 28 74 72 6F 70 20 6E 6F 20 29  0B63
64 69 64 20 29 42 55 33 28 23 21 20  0B6F
65 63 75 64 6F 72 70 20 74 6F 6E 20  0B7B
   74 6E 65 74 73 69 73 6E 6F 63 20  0B87
75 73 20 49 57 41 44 41 09 09 2F 21  0B92    773                  \!/             ADAWI sums when all were added together.\
20 6C 6C 61 20 6E 65 68 77 20 73 6D  0B9E
74 20 64 65 64 64 61 20 65 72 65 77  0BAA
            2E 72 65 68 74 65 67 6F  0BB6
                                     0BBE    774
                                     0BBE    775 IQUEUE_MSG:                 ; "Free" and "busy" queues inconsistent
73 28 55 50 43 09 00000BC6'010E0000' 0BBE    776          .ASCID  \       CPU(s) on port(s) !#(3UB) had inconsistent queues or\-
29 73 28 74 72 6F 70 20 6E 6F 20 29  0BCC
64 61 68 20 29 42 55 33 28 23 21 20  0BD8
6E 65 74 73 69 73 6E 6F 63 6E 69 20  0BE4
   72 6F 20 73 65 75 65 75 71 20 74  0BF0
6F 63 20 65 75 65 75 71 09 09 2F 21  0BFB    777                  \!/             queue counts when testing interlocking queue\-
65 74 20 6E 65 68 77 20 73 74 6E 75  0C07
6C 72 65 74 6E 69 20 67 6E 69 74 73  0C13
65 75 65 75 71 20 67 6E 69 6B 63 6F  0C1F
6E 6F 69 74 63 75 72 74 73 6E 69 20  0C2B    778                  \ instructions!/                against one another.\
74 73 6E 69 61 67 61 09 09 2F 21 73  0C37
72 65 68 74 6F 6E 61 20 65 6E 6F 20  0C43
                        2E           0C4F
```

```
                                0C50   780          .SBTTL  Read/Write Data
                         00000000   781          .PSECT  RWDATA,WRT,NOEXE,PAGE
                                0000   782
                                0000   783 TTCHAN:                                ; Channel associated with ctrl. term.
                           0000 0000   784          .WORD   0
                                0002   785
                                0002   786 FLAG:                                  ; Miscellaneous flag bits
                           0000 0002   787          .WORD   0                      ; (See Equated Symbols for definitions)
                                0004   788
                                0004   789 FAO_BUF:                               ; FAO output string descriptor
                      0000 00C8 0004   790          .WORD   TEXT_BUFFER,0
                       00000014' 0008   791          .ADDRESS BUFFER
                                000C   792
                                000C   793 BUFFER_PTR:                           ; Fake .ASCID buffer for misc. strings
                      0000 00C8 000C   794          .WORD   TEXT_BUFFER,0          ; A word for length, a word for desc.
                       00000014' 0010   795          .ADDRESS BUFFER
                                0014   796
                                0014   797 BUFFER:                               ; FAO output and other misc. buffer
                         000000DC 0014   798          .BLKB   TEXT_BUFFER
                                00DC   799
                                00DC   800 DEVDSC:                               ; Device name descriptor
                      0000 000A 00DC   801          .WORD   MAX_DEV_DESIG,0
                       000000FB' 00E0   802          .ADDRESS DEV_NAME
                                00E4   803
                                00E4   804 PROCESS_NAME:                         ; Process name
      38 37 41 4D 000000EC'010E0000' 00E4   805          .ASCID  /MA78/
                         0000000B 00F0   806          PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
                         000000FB 00F0   807          .BLKB   PROCESS_NAME_FREE
                                00FB   808
                                00FB   809 DEV_NAME:                             ; Device name buffer
                         0000010A 00FB   810          .BLKB   MAX_DEV_DESIG+MAX_UNIT_DESIG
                         0000000F 010A   811          NAME_LEN = .-DEV_NAME
                                010A   812
                                010A   813 DIB:                                 ; Device Information Block
                      0000 0074 010A   814          .WORD   DIB$K_LENGTH,0
                       00000112' 010E   815          .ADDRESS DIBBUF
                                0112   816 DIBBUF:
                         00000186 0112   817          .BLKB   DIB$K_LENGTH
                                0186   818
                                0186   819 ERROR_COUNT:                           ; Cumulative error count at runtime
                         00000000 0186   820          .LONG   0
                                018A   821
                                018A   822 STATUS:                              ; Status value on program exit
                         00000000 018A   823          .LONG   0
                                018E   824
                                018E   825 QUAD_STATUS:                         ; IO status block for misc sys. svcs.
            00000000 00000000   018E   826          .QUAD   0
                                0196   827
                                0196   828 INADDRESS:                           ; $CRMPSC address storage
            00000000 00000000   0196   829          .LONG   0,0
                                019E   830 OUTADDRESS:
            00000000 00000000   019E   831          .LONG   0,0
                                01A6   832
                                01A6   833 UNIT_NUMBER:                         ; Current dev unit number
                           0000 01A6   834          .WORD   0
                                01A8   835
                                01A8   836 DEVNAM_LEN:                          ; Current device name length
```

```
                     0000  01A8   837          .WORD   0
                           01AA   838
                           01AA   839 RANDOM1:                                ; Random word #1
                AAAAAAAA   01AA   840          .LONG   ^XAAAAAAAA
                           01AE   841
                           01AE   842 RANDOM2:                                ; Random word #2
                A72EA72E   01AE   843          .LONG   ^XA72EA72E
                           01B2   844
                           01B2   845 ITERATION:                              ; # of times all tests were executed
                00000000   01B2   846          .LONG   0
                           01B6   847
                           01B6   848 PASS:                                   ; Pass count
                00000000   01B6   849          .LONG   0
                           01BA   850
                           01BA   851 MSG_BLOCK:                              ; Auxiliary $GETMSG info
                000001BE   01BA   852          .BLKB   4
                           01BE   853
                           01BE   854 EXIT_DESC:                              ; Exit handler descriptor
                00000000   01BE   855          .LONG   0
                00002125'  01C2   856          .ADDRESS EXIT_HANDLER
                00000001   01C6   857          .LONG   1
                0000018A'  01CA   858          .ADDRESS STATUS
                           01CE   859
                           01CE   860 ARG_COUNT:                              ; Argument counter used by ERROR_EXIT
                00000000   01CE   361          .LONG   0
                           01D2   862
                           01D2   863          .ALIGN QUAD                    ; Self-relative queue must be aligned
                           01D8   864 UNIT_LIST:                              ; Head of unit block circular queue
00000000 00000000          01D8   865          .QUAD   0
                           01E0   866
                           01E0   867 SHM_INFO_ARGS:                          ; Arg list for $CMKRNL SHM_INFO
                00000004   01E0   868          .LONG   4
                000001A6'  01E4   869          .ADDRESS UNIT_NUMBER
                000001F4   01E8   870          .BLKL   3
                           01F4   871
                           01F4   872 SCSNODE:                                ; The name of our node in a cluster
                000001FA   01F4   873          .BLKB   6
                           01FA   874
                           01FA   875 SCSNODE_LENGTH:                         ; Length of our cluster node name
                000001FE   01FA   876          .BLKL   1
                           01FE   877
                           01FE   878 NEW_NODE:                               ; Newly acquired node address
00000000 00000000          01FE   879          .QUAD   0
                           0206   880
                           0206   881          .ALIGN  QUAD                   ; Needed for interlocked queue s.uff
                           0208   882 COMGS_BUF:                             ; Buffer for creating interprocessor GS
                000010B0   0208   883          .BLKB   GS_K_END
                           10B0   884
                           10B0   885 SYNCH_MBBUF:                            ; Buffer to create/receive MB during...
                000010CF   10B0   886          .BLKB   MB_K_END               ; ...synchronous operations
                           10CF   887
                           10CF   888 SYNCH_MBCHN:                            ; Channel to access random MB during...
                000010D1   10CF   889          .BLKW   1                      ; ...synchronous operations
                000010D3   10D1   890          .BLKW   1                      ; Filler to make SS calls easier
                           10D3   891
                           10D3   892 SYNCH_MBISB:                            ; IOSB for synchronous QIO's to MBs
                000010DB   10D3   893          .BLKQ   1
```

```
                10DB   894
                10DB   895 SYNCH_MBPTR:                    ; String descriptor for...
000010E3        10DB   896          .BLKQ   1
                10E3   897
                10E3   898 SYNCH_MBNAM:                    ; ...buffer to form logical name for...
00001102        10E3   899          .BLKB   15+1+15        ; ...MB during synchronous operations
                1102   900
                1102   901 SYNCH_MBDAY:                    ; DAYTIM parameter if reply timer set
0000110A        1102   902          .BLKQ   1
                110A   903
                110A   904 AST_MBBUF:                      ; Buffer to create/receive MB during...
00001129        110A   905          .BLKB   MB_K_END       ; ...AST operations
                1129   906
                1129   907 AST_MBCHN:                      ; Channel to access random MB during...
0000112B        1129   908          .BLKW   1              ; ...AST operations
0000112D        112B   909          .BLKW   1              ; Filler to make SS calls easier
                112D   910
                112D   911 AST_MBISB:                      ; IOSB for AST level QIO's to MBs
00001135        112D   912          .BLKQ   1
                1135   913
                1135   914 AST_MBPTR:                      ; String descriptor for...
0000113D        1135   915          .BLKQ   1
                113D   916
                113D   917 AST_MBNAM:                      ; ...buffer to form logical name for...
0000115C        113D   918          .BLKB   15+1+15        ; ...MB during AST operations
                115C   919
                115C   920 AST_MBDAY:                      ; DAYTIM parameter if reply timer set
00001164        115C   921          .BLKQ   1
                1164   922
                1164   923 INTERLOCK_SAVED_R6:            ; Save R6 here during interlocking...
00001168        1164   924          .BLKL   1              ; ...instructions test
                1168   925
                1168   926 TQCNT:                         ; Count of remaining timer queue...
00000000        1168   927          .LONG   0              ; ...entries when doing EFN test fixup
```

```
                     116C   929          .SBTTL  RMS-32 Data Structures
                     116C   930          .ALIGN  LONG
                     116C   931
                     116C   932 SYSIN_FAB:                                ; Allocate FAB for SYS$INPUT
                     116C   933          $FAB-
                     116C   934          FNM = <SYS$INPUT>
                     11BC   935
                     11BC   936 SYSIN_RAB:                                ; Allocate RAB for SYS$INPUT
                     11BC   937          $RAB-
                     11BC   938          FAB = SYSIN_FAB,-
                     11BC   939          ROP = PMT,-
                     11BC   940          PBF = PROMPT,-
                     11BC   941          PSZ = PMTSIZ,-
                     11BC   942          UBF = DEV_NAME,-
                     11BC   943          USZ = NAME_LEN
                     1200   944
                     1200   945 INI_FAB:                                 ; Allocate FAB for UETINIDEV
                     1200   946          $FAB-
                     1200   947          FAC = <GET,PUT,UPD>,-
                     1200   948          RAT = CR,-
                     1200   949          SHR = <GET,PUT,UPI>,-
                     1200   950          FNM = <UETINIDEV.DAT>
                     1250   951
                     1250   952 INI_RAB:                                 ; Allocate RAB for UETINIDEV
                     1250   953          $RAB-
                     1250   954          FAB = INI_FAB,-
                     1250   955          RBF = BUFFER,-
                     1250   956          UBF = BUFFER,-
                     1250   957          USZ = REC_SIZE
                     1294   958
                     1294   959 DDB_RFA:                                 ; RFA storage for INI_RAB
0000129A             1294   960          .BLKB   6
                     129A   961
                     129A   962          .ALIGN  LONG
                     129C   963 SUP_FAB:                                 ; Allocate FAB for UETSUPDEV
                     129C   964          $FAB-
                     129C   965          FAC = GET,-
                     129C   966          SHR = <UPI,GET>,-
                     129C   967          RAT = CR,-
                     129C   968          FOP = UFO,-
                     129C   969          FNM = <UETSUPDEV.DAT>
                     12EC   970
                     12EC   971 ;
                     12EC   972 ; The dummy FABs and RABs defined below will be copied into their respective
                     12EC   973 ; areas in the node set up for a particular shared memory.  A single MOVC is
                     12EC   974 ; used for each FAB/RAB set, so the sets must remain with the RAB immediately
                     12EC   975 ; following the FAB.
                     12EC   976 ;
                     12EC   977 COMGSFAB:                                ; Dummy FAB used for interproc GS
                     12EC   978          $FAB-                           ; FAB for interproc GS is in MA_K_COMFAB
                     12EC   979          ALQ = COMGS_SIZE,-
                     12EC   980          FAC = <BIO,PUT>,-
                     12EC   981          FOP = SUP,-
                     12EC   982          MRS = GS_K_END,-
                     12EC   983          RFM = FIX
                     133C   984
                     133C   985 COMGSRAB:                                ; Dummy RAB used for interproc GS
```

```
133C   986          $RAB-                                   ; RAB for interproc GS is in MA_K_COMRAB
133C   987                  RAC = SEQ,-
133C   988                  RBF = COMGS_BUF,-
133C   989                  ROP = BIO,-
133C   990                  RSZ = GS_K_END
1380   991
1380   992 SPRGSFAB:                                        ; Dummy FAB used for single processor GS
1380   993          $FAB-                                   ; FAB for single proc GS is MA_K_SPRFAB
1380   994                  ALQ = SPRGS_SIZE,-
1380   995                  FAC = <BIO,PUT>,-
1380   996                  FOP = SUP,-
1380   997                  MRS = WRITE_SIZE,-
1380   998                  RFM = FIX
13D0   999
13D0  1000 SPRGSRAB:                                        ; Dummy RAB used for single processor GS
13D0  1001          $RAB-                                   ; RAB for single proc GS is MA_K_SPRRAB
13D0  1002                  RAC = SEQ,-
13D0  1003                  ROP = BIO,-
13D0  1004                  RSZ = WRITE_SIZE
1414  1005
1414  1006 CHKGSFAB:                                       ; Dummy FAB used to check single proc GS
1414  1007          $FAB-                                   ; FAB to check single proc GS is MA_K_CHKFAB
1414  1008                  ALQ = SPRGS_SIZE,-
1414  1009                  FAC = <BIO,GET>,-
1414  1010                  FOP = SUP,-
1414  1011                  MRS = WRITE_SIZE,-
1414  1012                  RFM = FIX
1464  1013
1464  1014 CHKGSRAB:                                       ; Dummy RAB used to check single proc GS
1464  1015          $RAB-                                   ; RAB to check single proc GS is MA_K_CHKRAB
1464  1016                  RAC = SEQ,-
1464  1017                  ROP = BIO,-
1464  1018                  USZ = WRITE_SIZE
```

```
                                                N 1
                     14A8  1020          .SBTTL  Test and Device Initialization
                 00000000  1021          .PSECT  MA780,EXE,NOWRT,PAGE
                     0000  1022
                     0000  1023          .DEFAULT DISPLACEMENT,WORD
                     0000  1024
                     0000  1025 ;+
                     0000  1026 ;       Start up the multiport memory test.  This entails some overhead
                     0000  1027 ;       necessary to cope with both expected and unforseen conditions, figuring
                     0000  1028 ;       out just what devices are to be tested, making sure we can test the
                     0000  1029 ;       indicated devices and setting up writeable space for each device to
                     0000  1030 ;       be tested.
                     0000  1031 ;-
                     0000  1032
                0000 0000  1033 .ENTRY UETMA7800,^M<>                           ; Entry mask
                     0002  1034
      6D  1F17'CF    DE    0002  1035          MOVAL   SSERROR,(FP)              ; Declare exception handler
                           0007  1036          $SETSFM_S ENBFLG = #1             ; Enable system service failure mode
                           0010  1037          $SETEF_S EFN = #SPRGS_EFN         ; (see note where SPRGS_EFN defined)
                           0019  1038          $DCLEXH_S DESBLK = EXIT_DESC      ; Declare an exit handler
                           0024  1039
                           0024  1040          $OPEN   FAB = SYSIN_FAB,-         ; Open SYS$INPUT
                           0024  1041                  ERR = RMS_ERROR
                           0033  1042          $CONNECT RAB = SYSIN_RAB,-        ; Connect RAB to SYS$INPUT
                           0033  1043                   ERR = RMS_ERROR
            02     E1     0042  1044          BBC     S^#DEV$V_TRM,-            ; BR if SYS$INPUT is NOT a terminal
    1E 11AC'CF            0044  1045                  SYSIN_FAB+FAB$L_DEV,10$
                           0048  1046          $TRNLOG_S LOGNAM = CONTROLLER,-   ; Get the name of our memory
                           0048  1047                    RSLLEN = DEVNAM_LEN,-
                           0048  1048                    RSLBUF = DEVDSC
        01     50     D1    0061  1049          CMPL    R0,#SS$_NORMAL            ; Was a controller specified?
               2E     13    0064  1050          BEQL    PROC_CONT_NAME            ; BR if it was - go process it
                            0066  1051 10$:
                           0066  1052          $GET    RAB = SYSIN_RAB,-         ; Read SYS$INPUT...
                           0066  1053                  ERR = RMS_ERROR           ; ...for the controller name
            11DE'CF    B0    0075  1054          MOVW    SYSIN_RAB+RAB$W_RSZ,-     ; Save the name length
            01A8'CF          0079  1055                  DEVNAM_LEN
                  16   12    007C  1056          BNEQ    PROC_CONT_NAME            ; BR if we got something
         018A'CF    14   D0    007E  1057          MOVL    #SS$_BADPARAM,STATUS      ; Save an exit status if not
            00A3'CF    01   DF    0083  1058          PUSHAL  NO_CTRLNAME               ; Prepare for message...
                       01   DD    0087  1059          PUSHL   #1                        ; ...
                 00741132   8F    DD    0089  1060          PUSHL   #UETP$_TEXT!STS$K_ERROR   ; ...
                       03   DD    008F  1061          PUSHL   #3                        ; ...
                     200C   31    0091  1062          BRW     ERROR_EXIT                ; ...to tell of bad setup
                            0094  1063
                            0094  1064 PROC_CONT_NAME:
   00DC'CF  01A8'CF    3C    0094  1065          MOVZWL  DEVNAM_LEN,DEVDSC         ; Set the device name length
            00DC'CF    DF    009B  1066          PUSHAL  DEVDSC                    ; Make sure...
            00DC'CF    DF    009F  1067          PUSHAL  DEVDSC                    ; ...that the specified controller...
     00000000'GF       02    FB    00A3  1068          CALLS   #2,G^STR$UPCASE           ; ...is all uppercase for later comaparison
52     00DC'CF    01    C1    00AA  1069          ADDL3   #1,DEVDSC,R2              ; Estimate the eventual...
         00E4'CF    52    A0    00B0  1070          ADDW2   R2,PROCESS_NAME           ; ...process name length (incl. "_")
                          DE    00B5  1071          MOVAL   PROCESS_NAME+8-           ; Locate first available byte...
                           00B6  1072                  +MAX_PROC_NAME-           ; ...in process name handle...
         50    00F0'CF          00B6  1073                  -PROCESS_NAME_FREE,R0     ; ...for device name
                     0B   C3    00BA  1074          SUBL3   #PROCESS_NAME_FREE,-      ; Will the device name fit...
               51    52         00BC  1075                  R2,R1                     ; ...in the remaining space?
                     08   15    00BE  1076          BLEQ    10$                       ; BR if it will
```

```
                                                B 2
                50    51    C2  00C0  1077          SUBL2   R1,R0                           ; Overwrite handle otherwise...
            00E4'CF   0F    B0  00C3  1078          MOVW    #MAX_PROC_NAME,PROCESS_NAME ; ...and define the maximum length
                                00C8  1079 10$:
              80    5F  8F    90  00C8  1080          MOVB    #^A/ /,(R0)+                    ; Separate handle from device name
        60  00FB'CF   00DC'CF   28  00CC  1081          MOVC3   DEVDSC,DEV_NAME,(R0)    ; Concatenate handle with device name
                        7E    D4  00D4  1082          CLRL    -(SP)                           ; Set the time stamp flag
                    000F'CF   DF  00D6  1083          PUSHAL  TEST_NAME                       ; Set the test name
                          02    DD  00DA  1084          PUSHL   #2                              ; Push the argument count
                    00741039  8F  DD  00DC  1085          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
                00000000'GF   04  FB  00E2  1086          CALLS   #4,G^LIB$SIGNAL         ; Print the startup message
                    0002'CF   08  A8  00E9  1087          BISW2   #BEGIN_MSGM,FLAG        ; Set flag so we don't print it again
                                      00EE  1088          $SETPRN_S PRCNAM = PROCESS_NAME ; Set the process name to UETMA7800_x
                                      00F9  1089
                          02    E1  00F9  1090          BBC     S^#DEV$V_TRM,-          ; BR if SYS$INPUT is NOT a terminal
                    66 11AC'CF        00FB  1091                  SYSIN_FAB+FAB$L_DEV,20$
                                      00FF  1092          $GETDVI_S DEVNAM = SYS$INPUT,-  ; Get the name of...
                                      00FF  1093                    EFN    = #SS_SYNCH_EFN,- ; ...device which may abort test
                                      00FF  1094                    ITMLST = INPUT_ITMLST,-
                                      00FF  1095                    IOSB   = QUAD_STATUS
                    45 018E'CF    E9  011B  1096          BLBC    QUAD_STATUS,20$         ; Avoid CTRL/C handler if any error
                                      0120  1097          $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handler
                                      0120  1098                    CHAN   = TTCHAN
                                      0131  1099          $QIOW_S CHAN    = TTCHAN,-      ; Enable CTRL/C AST's...
                                      0131  1100                  FUNC    = #IO$_SETMODE!IO$M_CTRLCAST,-
                                      0131  1101                  P1      = CCASTHAND
                    00E4'CF   DF  0152  1102          PUSHAL  PROCESS_NAME                    ; ...and tell the user...
                          01    DD  0156  1103          PUSHL   #1                              ; ...
                    0074832B  8F  DD  0158  1104          PUSHL   #UETP$_ABORTC!STS$K_SUCCESS ; ...how to abort gracefully...
                00000000'GF   03  FB  015E  1105          CALLS   #3,G^LIB$SIGNAL         ; ...
                                      0165  1106 20$:
                                      0165  1107          $TRNLOG_S LOGNAM = MODE,-       ; Get the run mode
                                      0165  1108                    RSLLEN = BUFFER_PTR,-
                                      0165  1109                    RSLBUF = FAO_BUF
                  0014'CF     20  8A  017E  1110          BICB2   #LC_BITM,BUFFER         ; Convert to upper case
            0014'CF     4F  8F  91  0183  1111          CMPB    #^A/O/,BUFFER           ; Is this a one shot?
                          05    12  0189  1112          BNEQ    25$                     ; BR if not
                    0002'CF   12  A8  018B  1113          BISW2   #ONE_SHOTM!TEST_OVERM,FLAG ; Set flags for one-shot mode
                                      0190  1114 25$:
```

```
                                0190  1116 ;
                                0190  1117 ; From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller
                                0190  1118 ; and unit configuration and lets us know if the setup to run this test was
                                0190  1119 ; done correctly.
                                0190  1120 ;
                                0190  1121         $OPEN   FAB = INI_FAB,-         ; Open file 'UETINIDEV.DAT'
                                0190  1122                 ERR = RMS_ERROR
                                019F  1123         $CONNECT RAB = INI_RAB,-        ; Connect the RAB and FAB
                                019F  1124                  ERR = RMS_ERROR
                                01AE  1125         $MGBLSC_S  INADR = INADDRESS,-  ; Connect to UETSUPDEV global section
                                01AE  1126                    RETADR = OUTADDRESS,-
                                01AE  1127                    GSDNAM = SUPDEV_GBLSEC,-
                                01AE  1128                     FLAGS = #SEC$M_EXPREG
        00000978 8F    50   D1  01CD  1129         CMPL    R0,#SS$_NOSUCHSEC       ; Was the section already there?
                       37   12  01D4  1130         BNEQ    30$                     ; BR if it was...
                                01D6  1131         $OPEN   FAB = SUP_FAB,-         ; ...else open 'UETSUPDEV.DAT'
                                01D6  1132                 ERR = RMS_ERROR
                                01E5  1133         $CRMPSC_S CHAN = SUP_FAB+FAB$L_STV,- ; Create the global section
                                01E5  1134                   INADR = INADDRESS,-
                                01E5  1135                  RETADR = OUTADDRESS,-
                                01E5  1136                  GSDNAM = SUPDEV_GBLSEC,-
                                01E5  1137                   FLAGS = #SEC$M_EXPREG!SEC$M_GBL
                                020D  1138 30$:
    56  01A2'CF   019E'CF   C3  020D  1139         SUBL3   OUTADDRESS,OUTADDRESS+4,R6 ; Compute global section length
                                0215  1140
                                0215  1141 FIND_IT:
                                0215  1142         $GET    RAB = INI_RAB,-         ; Get the first record
                                0215  1143                 ERR = RMS_ERROR
                01CC'CF    DF   0224  1144         PUSHAL  CONT_DESC               ; Make sure...
                01CC'CF    DF   0228  1145         PUSHAL  CONT_DESC               ; ...that the controller name...
          00000000'GF      02   FB  022C  1146         CALLS   #2,G^STR$UPCASE         ; ...is all uppercase letters
           0014'CF    44 8F    91  0233  1147         CMPB    #^A/D/,BUFFER           ; Is this a DDB?
                         0A    13  0239  1148         BEQL    10$                     ; Go on if not
           0014'CF    45 8F    91  023B  1149         CMPB    #^A/E/,BUFFER           ; Is this the end of the file?
                         D2    12  0241  1150         BNEQ    FIND_IT                 ; Continue on if not
                         75    11  0243  1151         BRB     30$                     ; Use common error message if it is
                                0245  1152 10$:
  00FB'CF   001A'CF   01A8'CF    29  0245  1153         CMPC    DEVNAM_LEN,BUFFER+6,DEV_NAME ; Is this the right controller?
                         C4    12  024F  1154         BNEQ    FIND_IT                 ; BR if not
      1294'CF   1260'CF      06   28  0251  1155         MOVC3   #6,INI_RAB+RAB$W_RFA,DDB_RFA ; Save the Record File Address
           0018'CF    54 8F    91  0259  1156         CMPB    #^A/T/,BUFFER+4         ; Can we test this controller?
                         2F    13  025F  1157         BEQL    20$                     ; BR if we can...
                                0261  1158         $FAO_S  CTRSTR = DEAD_CTRLNAME,- ; ...and yell at user if we can't
                                0261  1159                 OUTLEN = BUFFER_PTR,-
                                0261  1160                 OUTBUF = FAO_BUF,-
                                0261  1161                 P1     = #DEVDSC
                018A'CF    14   D0  0[illegible]  1162         MOVL    #SS$_BADPARAM,STATUS    ; Set return status
                     000C'CF    DF  027F  1163         PUSHAL  BUFFER_PTR              ; ...
                                01  DD  0283  1164         PUSHL   #1                      ; ...
                 00741132 8F    DD  0285  1165         PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...
                                03  DD  028B  1166         PUSHL   #3                      ; ...
                       1E10     31  028D  1167         BRW     ERROR_EXIT              ; We can't test what we can't test
```

```
                                0290  1169 ;
                                0290  1170 ; Check here (only once) to see if we're the correct test for the
                                0290  1171 ; device we were given to check.  Note that this section differs substantially
                                0290  1172 ; from the standard UETP Device Test Template because there is no way to get
                                0290  1173 ; the "device characteristics" of a multiport memory.
                                0290  1174 ;
                                0290  1175 20$:
      00  00FB'CF    00DC'CF   2D  0290  1176          CMPC5   DEVDSC,DEV_NAME,#0,-    ; Match controller name vs. 'MPM'
          0270'CF    0268'CF         0298  1177                  MPM_LITERAL,MPM_LITERAL+8
                           1A   12  029E  1178          BNEQ    30$                     ; BR if another device was given
      56  027B'CF    0273'CF   39  02A0  1179          MATCHC  CS2,CS2+8,R6,@OUTADDRESS ; Find our line in UETSUPDEV.DAT
                     019E'DF         02A8
                           0D   12  02AB  1180          BNEQ    30$                     ; BR if we're not there
                55    000F'CF   9A  02AD  1181          MOVZBL  TEST_NAME,R5            ; Get the test name length
          0017'CF   63    55   29  02B2  1182          CMPC3   R5,(R3),TEST_NAME+8     ; Are we the right test?
                           1F   13  02B8  1183          BEQL    40$                     ; BR if we are
                                02BA  1184 30$:
                     00DC'CF    DF  02BA  1185          PUSHAL  DEVDSC                  ; Push device not supported message
                     00E4'CF    DF  02BE  1186          PUSHAL  PROCESS_NAME            ; Parameters on the stack
                           02   DD  02C2  1187          PUSHL   #2                      ; Push the argument count
                00748333   8F   DD  02C4  1188          PUSHL   #UETP$_DENOSU
                           02   F0  02CA  1189          INSV    #STS$K_ERROR,-          ; Set the severity code...
                           00       02CC  1190                  #STS$V_SEVERITY,-
                        6E 03       02CD  1191                  #STS$S_SEVERITY,(SP)
                018A'CF    6E   D0  02CF  1192          MOVL    (SP),STATUS             ; ...and save it as the exit status
                           04   DD  02D4  1193          PUSHL   #4                      ; Push the partial arg count...
                      1DC7      31  02D6  1194          BRW     ERROR_EXIT              ; ...and split this scene
                                02D9  1195
                                02D9  1196 40$:
                                02D9  1197          $GETSYIW_S  EFN = #SS_SYNCH_EFN,- ; Get our processor's ECO level...
                                02D9  1198                      ITMLST = GETSYI_ITMLST,-  ; ...the version of VMS and...
                                02D9  1199                      IOSB   = QUAD_STATUS      ; ...our SCS node name
01F4'CF    06   20    3A  02F0  1200          LOCC    #^A/ /,#6,SCSNODE       ; Remove the blanks...
01FA'CF    06   50    C3  02F6  1201          SUBL3   R0,#6,SCSNODE_LENGTH    ; ...that pad out the node name
                                02FC  1202
                                02FC  1203 FOUND_IT:
                                02FC  1204          $GET    RAB = INI_RAB,-         ; Get a record
                                02FC  1205                  ERR = RMS_ERROR
                     01CC'CF    DF  030B  1206          PUSHAL  CONT_DESC               ; Make sure...
                     01CC'CF    DF  030F  1207          PUSHAL  CONT_DESC               ; ...that this line...
 00000000'GF       02   FB  0313  1208          CALLS   #2,G^STR$UPCASE         ; ...is all uppercase letters
  0014'CF    55 8F   91  031A  1209          CMPB    #^A/U/,BUFFER           ; Is this a UCB?
                           24   13  0320  1210          BEQL    30$                     ; BR if it is
  0014'CF    44 8F   91  0322  1211          CMPB    #^A/D/,BUFFER           ; Is this a DDB?
                           19   13  0328  1212          BEQL    20$                     ; BR if yes
  0014'CF    45 8F   91  032A  1213          CMPB    #^A/E/,BUFFER           ; Is this the end?
                           11   13  0330  1214          BEQL    20$                     ; BR if yes
                                0332  1215 10$:
                0130'CF    DF  0332  1216          PUSHAL  ILLEGAL_REC             ; Then this is an error in the record
                           01   DD  0336  1217          PUSHL   #1                      ; Push the error message
             00741132   8F   DD  0338  1218          PUSHL   #UETP$_TEXT!STS$K_ERROR ; Push the signal name
                           03   DD  033E  1219          PUSHL   #3                      ; Push the temp arg count
                      1D5D      31  0340  1220          BRW     ERROR_EXIT              ; Finish for good
                                0343  1221 20$:
                      02C6      31  0343  1222          BRW     ALL_SET                 ; Found DDB or END
                                0346  1223 30$:
  0018'CF    54 8F   91  0346  1224          CMPB    #^A/T/,BUFFER+4         ; Is the unit testable?
```

```
                     AE   12  034C  1225          BNEQ    FOUND_IT                ; BR if not
                     01   DD  034E  1226          PUSHL   #1                      ; Flag to ignore blanks when converting
                     02   DD  0350  1227          PUSHL   #2                      ; Set byte size of results
           01A6'CF   DF  0352  1228          PUSHAL  UNIT_NUMBER             ; Set address to receive word
           01C4'CF   DF  0356  1229          PUSHAL  UNIT_DESC               ; Push string address
00000000'GF    04   FB  035A  1230          CALLS   #4,G^OTS$CVT_TI_L       ; Convert ASCII unit # to decimal
          CE 50   E9  0361  1231          BLBC    R0,10$                  ; Don't allow bogus unit to pass
                      0364  1232          ;BRB    60$                     ; Sh. mems have no device chars, so...
                      0364  1233                                          ; ...skip usual $GETDEV/UETSUPDEV check
```

```
                     0364  1235 ;
                     0364  1236 ; We've found a valid memory to test.  Allocate space to store unit-specific
                     0364  1237 ; information and initialize the information.  Interprocessor data structures
                     0364  1238 ; are unique between memories; single processor data structures are unique
                     0364  1239 ; both between memories and within a shared memory.  See the FT_NO_INTERACT
                     0364  1240 ; definition for a discussion of the use of OTHER_CHARS.
                     0364  1241 ;
                     0364  1242 60$:
                     0364  1243         $EXPREG_S PAGCNT = #PAGES,-      ; Get a new node of demand zero memory
                     0364  1244                   RETADR = NEW_NODE
01D8'CF  01FE'DF  5D 0375  1245         INSQTI  @NEW_NODE,UNIT_LIST      ; Put the new node in the unit list
     56  01FE'CF  D0 037C  1246         MOVL    NEW_NODE,R6              ; Save a copy of its address
       08 A6   01  90 0381  1247         MOVB    #1,UETUNT$B_TYPE(R6)     ; Set the structure type
          064C 8F  B0 0385  1248         MOVW    #UETUNT$C_INDSIZ+DEVDEP_SIZE,-
            09 A6      0389  1249                 UETUNT$W_SIZE(R6)        ; Set the structure size
                       038B  1250         $GETTIM_S TIMADR = MA_Q_DAYTIME(R6) ; Save time as interprocessor code
01EC C6  01A6'CF  B0 0396  1251         MOVW    UNIT_NUMBER,MA_W_UNIT(R6)        ; Given memory unit number...
01E8'CF  01AC C6  9E 039D  1252         MOVAB   MA_B_MYPORT(R6),SHM_INFO_ARGS+08 ; ...find my port on it,...
01EC'CF  01F9 C6  9E 03A4  1253         MOVAB   MA_T_MEMNAM(R6),SHM_INFO_ARGS+12 ; ...the memory name...
01F0'CF  01F1 C6  7E 03AB  1254         MOVAQ   MA_Q_MEMNAM(R6),SHM_INFO_ARGS+16 ; ...and the name length...
                       03B2  1255         $CMKRNL_S ROUTIN = SHM_INFO,-    ; ...
                       03B2  1256                   ARGLST = SHM_INFO_ARGS
            01F9 C6  9E 03C1  1257         MOVAB   MA_T_MEMNAM(R6),-        ; Finish .ASCID pointer to memory name
            01F5 C6    03C5  1258                 MA_Q_MEMNAM+4(R6)
                 02  88 03C8  1259         BISB2   #UETUNT$M_TESTABLE,-     ; Assume for now that MPM is testable
              0B A6      03CA  1260                 UETUNT$B_FLAGS(R6)
                         03CC  1261         ; Note that the unit number is treated as a word above, but as a byte
                         03CC  1262         ; forevermore.  (The current hardware uses a 2-bit unit number.)
     50   01EC C6  90 03CC  1263         MOVB    MA_W_UNIT(R6),R0         ; Get hex ASCII unit number...
             0E5D  30 03D1  1264         BSBW    HEXUNIT                  ; ...
     01F0 C6   50  90 03D4  1265         MOVB    R0,MA_T_UNIT(R6)         ; Put it away for safekeeping...
          59   50  90 03D9  1266         MOVB    R0,R9                    ; ...and for convenience
                   03DC  1267
                   03DC  1268
          01F1 C6  28 03DC  1269         MOVC3   MA_Q_MEMNAM(R6),-        ; Interproc GS name is memory-name...
0210 C6   01F9 C6    03E0  1270                 MA_T_MEMNAM(R6),MA_T_COMGSNAM(R6)
          83   3A  90 03E6  1271         MOVB    #^A/:/,(R3)+             ; ...colon...
63  0289'CF   0281'CF  28 03E9  1272         MOVC3   UETCOMGS,UETCOMGS+8,(R3) ; ...section-name...
          83   59  90 03F1  1273         MOVB    R9,(R3)+                 ; ...and unit to make it unique
    63  01AC C6   30 03F4  1274         ADDB3   #^A/0/,MA_B_MYPORT(R6),(R3) ; Append port in case FT_NO_INTERACT
57  0281'CF  01F1 C6  81 03FA  1275         ADDB3   MA_Q_MEMNAM(R6),UETCOMGS,R7 ; Figure total name length...
0208 C6   57   02  81 0402  1276         ADDB3   #OTHER_CHARS,R7,-           ; ...
                   0408  1277                 MA_Q_COMGSNAM(R6)
          0210 C6  DE 0408  1278         MOVAL   MA_T_COMGSNAM(R6),-      ; Finish .ASCID pointer
          020C C6    040C  1279                 MA_Q_COMGSNAM+4(R6)
```

```
                01F1 C6   28  040F  1281          MOVC3   MA_Q_MEMNAM(R6),-           ; Single proc GS name is memory-name...
        0287 C6 01F9 C6       0413  1282                  MA_T_MEMNAM(R6),MA_T_SPRGSNAM(R6)
                83   3A   90  0419  1283          MOVB    #^A/:/,(R3)+                ; ...colon...
    63  0299'CF 0291'CF   28  041C  1284          MOVC3   UETSPRGS,UETSPRGS+8,(R3)    ; ...section-name...
                83   59   90  0424  1285          MOVB    R9,(R3)+                    ; ...unit...
        63  01AC C6   30   81  0427  1286          ADDB3   #^A/0/,MA_B_MYPORT(R6),(R3) ; ...and port to make it unique
    57  0291'CF 01F1 C6   81  042D  1287          ADDB3   MA_Q_MEMNAM(R6),UETSPRGS,R7 ; Figure total name length...
        027F C6  57   03   81  0435  1288          ADDB3   #3,R7,MA_Q_SPRGSNAM(R6)     ; ...
                 0287 C6  DE  043B  1289          MOVAL   MA_T_SPRGSNAM(R6),-     ; Finish .ASCID pointer
                 0283 C6      043F  1290                  MA_Q_SPRGSNAM+4(R6)
                              0442  1291
                              0442  1292
                01F1 C6   28  0442  1293          MOVC3   MA_Q_MEMNAM(R6),-           ; Interproc CEF name is memory-name...
        0237 C6 01F9 C6       0446  1294                  MA_T_MEMNAM(R6),MA_T_COMEFNAM(R6)
                83   3A   90  044C  1295          MOVB    #^A/:/,(R3)+                ; ...colon...
    63  02B6'CF 02BA'CF   28  044F  1296          MOVC3   UETCOMEF,UETCOMEF+8,(R3)    ; ...cluster-name...
                83   59   90  0457  1297          MOVB    R9,(R3)+                    ; ...and unit to make it unique
        63  01AC C6   30   81  045A  1298          ADDB3   #^A/0/,MA_B_MYPORT(R6),(R3) ; Append port in case FT_NO_INTERACT
    57  02AE'CF 01F1 C6   81  0460  1299          ADDB3   MA_Q_MEMNAM(R6),UETCOMEF,R7 ; Figure total name length...
        022F C6  57   02   81  0468  1300          ADDB3   #OTHER_CHARS,R7,-            ; ...
                              046E  1301                  MA_Q_COMEFNAM(R6)
                 0237 C6  DE  046E  1302          MOVAL   MA_T_COMEFNAM(R6),-     ; Finish .ASCID pointer
                 0233 C6      0472  1303                  MA_Q_COMEFNAM+4(R6)
                              0475  1304
                              0475  1305
                01F1 C6   28  0475  1306          MOVC3   MA_Q_MEMNAM(R6),-           ; Single proc CEF name is memory-name...
        02AE C6 01F9 C6       0479  1307                  MA_T_MEMNAM(R6),MA_T_SPREFNAM(R6)
                83   3A   90  047F  1308          MOVB    #^A/:/,(R3)+                ; ...colon...
    63  02C6'CF 02BE'CF   28  0482  1309          MOVC3   UETSPREF,UETSPREF+8,(R3)    ; ...cluster-name...
                83   59   90  048A  1310          MOVB    R9,(R3)+                    ; ...unit...
        63  01AC C6   30   81  048D  1311          ADDB3   #^A/0/,MA_B_MYPORT(R6),(R3) ; ...and port to make it unique
    57  02BE'CF 01F1 C6   81  0493  1312          ADDB3   MA_Q_MEMNAM(R6),UETSPREF,R7 ; Figure total name length...
        02A6 C6  57   03   81  049B  1313          ADDB3   #3,R7,MA_Q_SPREFNAM(R6)     ; ...
                 02AE C6  DE  04A1  1314          MOVAL   MA_T_SPREFNAM(R6),-     ; Finish .ASCID pointer
                 02AA C6      04A5  1315                  MA_Q_SPREFNAM+4(R6)
```

```
                01F1 C6    28  04AB  1317          MOVC3   MA_Q_MEMNAM(R6),-          ; Other CPU's MB name is memory-name...
          0260 C6   01F9 C6      04AC  1318                  MA_T_MEMNAM(R6),MA_T_COMMBNAM(R6)
                    83   3A    90  04B2  1319          MOVB    #^A/:/,(R3)+                ; ...colon...
          63  02A9'CF   02A1'CF  28  04B5  1320          MOVC3   UETMB,UETMB+8,(R3)          ; ...mailbox-name...
                    63   59    90  04BD  1321          MOVB    R9,(R3)                     ; ...and unit, to form skeleton of name
          57  02A1'CF   01F1 C6  81  04C0  1322          ADDB3   MA_Q_MEMNAM(R6),UETMB,R7 ; Figure total name length...
          0258 C6   57   03    81  04C8  1323          ADDB3   #3,R7,MA_Q_COMMBNAM(R6)     ; ...(include extra char for port)
                                   04CE  1324          ;MOVAL  MA_T_COMMBNAM(R6),-         ; Don't bother finishing skeleton
                                   04CE  1325          ;       MA_Q_COMMBNAM+4(R6)
                                   04CE  1326
                                   04CE  1327
                01F1 C6    28  04CE  1328          MOVC3   MA_Q_MEMNAM(R6),-           ; My CPU's MB name's memory-name...
          02DD C6   01F9 C6      04D2  1329                  MA_T_MEMNAM(R6),MA_T_SPRMBNAM(R6)
                    83   3A    90  04D8  1330          MOVB    #^A/:/,(R3)+                ; ...colon...
          63  02A9'CF   02A1'CF  28  04DB  1331          MOVC3   UETMB,UETMB+8,(R3)          ; ...mailbox-name...
                    83   59    90  04E3  1332          MOVB    R9,(R3)+                    ; ...unit...
          63   01AC C6    30    81  04E6  1333          ADDB3   #^A/0/,MA_B_MYPORT(R6),(R3) ; ...and port to make it unique
          57  02A1'CF   01F1 C6  81  04EC  1334          ADDB3   MA_Q_MEMNAM(R6),UETMB,R7 ; Figure total name length...
          02D5 C6   57   03    81  04F4  1335          ADDB3   #3,R7,MA_Q_SPRMBNAM(R6)    ; ...
                    02DD C6    DE  04FA  1336          MOVAL   MA_T_SPRMBNAM(R6),-         ; Finish .ASCID pointer
                    02D9 C6        04FE  1337                  MA_Q_SPRMBNAM+4(R6)
                                   0501  1338
                                   0501  1339
             57     0110 C6    DE  0501  1340          MOVAL   MA_K_COMFAB(R6),R7          ; Point to interprocessor GS FAB
                    0094 8F    28  0506  1341          MOVC3   #FAB$K_BLN+RAB$K_BLN,-      ; Set up interproc GS skeleton FAB & RAB
             67     12EC'CF        050A  1342                  COMGSFAB,(R7)
             019C C6    57  D0  050E  1343          MOVL    R7,MA_K_COMRAB+RAB$L_FAB(R6) ; Set the FAB address in the RAB
          02D6'CF   02CE'CF  28  0513  1344          MOVC3   SYS$TEST,SYS$TEST+8,-       ; Set up constant part of name...
                      14 A6        051A  1345                  MA_T_COMFAB(R6)
          63  0289'CF   0281'CF  28  051C  1346          MOVC3   UETCOMGS,UETCOMGS+8,(R3) ; ...
          63  01F4'CF   01FA'CF  28  0524  1347          MOVC3   SCSNODE_LENGTH,SCSNODE,(R3) ; Allow for SYS$TEST in SYS$COMMON
                    83   59    90  052C  1348          MOVB    R9,(R3)+                    ; Append the unit number for uniqueness
          63  02E7'CF   02DF'CF  28  052F  1349          MOVC3   DOTDAT,DOTDAT+8,(R3)       ; Give the file a .DAT type
                58   0281'CF   01  81  0537  1350          ADDB3   #1,UETCOMGS,R8              ; Set the FNS field in the FAB...
                     58   01FA'CF  80  053D  1351          ADDB2   SCSNODE_LENGTH,R8           ; ...
                     58   02CE'CF  80  0542  1352          ADDB2   SYS$TEST,R8                 ; ...
            34 A7   02DF'CF   58  81  0547  1353          ADDB3   R8,DOTDAT,FAB$B_FNS(R7)     ; ...
                          14 A6   DE  054E  1354          MOVAL   MA_T_COMFAB(R6),-           ; Set the FNA field in the FAB
                          2C A7       0551  1355                  FAB$C_FNA(R7)
```

```
                57   03FC C6   DE  0553  1357          MOVAL   MA_K_SPRFAB(R6),R7       ; Point to single processor GS FAB
                     0094 8F   28  0558  1358          MOVC3   #FAB$K_BLN+RAB$K_BLN,-  ; Set up skeleton FAB and RAB
                67   1380'CF        055C  1359                  SPRGSFAB,(R7)
                0488 C6   57   D0  0560  1360          MOVL    R7,MA_K_SPRRAB+RAB$L_FAB(R6) ; Have the RAB point to the FAB
          02D6'CF   02CE'CF   28  0565  1361          MOVC3   SYS$TEST,SYS$TEST+8,-   ; Set up constant part of name...
                02FC C6        056C  1362                  MA_T_SPRFAB(R6)
     63   0299'CF   0291'CF   28  056F  1363          MOVC3   UETSPRGS,UETSPRGS+8,(R3) ; ...
     63   01F4'CF   01FA'CF   28  0577  1364          MOVC3   SCSNODE_LENGTH,SCSNODE,(R3) ; Allow for SYS$TEST in SYS$COMMON
                83   59   90  057F  1365          MOVB    R9,(R3)+                ; Append the unit number for uniqueness
     63   02E7'CF   02DF'CF   28  0582  1366          MOVC3   DOTDAT,DOTDAT+8,(R3)    ; Give the file a .DAT type
          58   0291'CF   01   81  058A  1367          ADDB3   #1,UETSPRGS,R8          ; Set the FNS field in the FAB...
               58   01FA'CF   80  0590  1368          ADDB2   SCSNODE_LENGTH,R8       ; ...
               58   02CE'CF   80  0595  1369          ADDB2   SYS$TEST,R8             ; ...
     34 A7     02DF'CF   58   81  059A  1370          ADDB3   R8,DOTDAT,FAB$B_FNS(R7) ; ...
                02FC C6   DE  05A1  1371          MOVAL   MA_T_SPRFAB(R6),-       ; Set the FNA field in the FAB
                  2C A7        05A5  1372                  FAB$L_FNA(R7)
                064C C6   DE  05A7  1373          MOVAL   MA_K_SPRBUF(R6),-       ; Set RBF field
                0474 C6        05AB  1374                  MA_K_SPRRAB+RAB$L_RBF(R6)
                              05AE  1375
                              05AE  1376
                57   0590 C6   DE  05AE  1377          MOVAL   MA_K_CHKFAB(R6),R7      ; Point to single processor GS check FAB
                     0094 8F   28  05B3  1378          MOVC3   #FAB$K_BLN+RAB$K_BLN,-  ; Set up skeleton FAB and RAB
                67   1414'CF        05B7  1379                  CHKGSFAB,(R7)
                061C C6   57   D0  05BB  1380          MOVL    R7,MA_K_CHKRAB+RAB$L_FAB(R6) ; Have the RAB point to the FAB
          02D6'CF   02CE'CF   28  05C0  1381          MOVC3   SYS$TEST,SYS$TEST+8,-   ; Set up constant part of name...
                0490 C6        05C7  1382                  MA_T_CHKFAB(R6)
     63   0299'CF   029?'CF   28  05CA  1383          MOVC3   UETSPRGS,UETSPRGS+8,(R3) ; ...
     63   01F4'CF   01FA'CF   28  05D2  1384          MOVC3   SCSNODE_LENGTH,SCSNODE,(R3) ; Allow for SYS$TEST in SYS$COMMON
                83   59   90  05DA  1385          MOVB    R9,(R3)+                ; Append the uni* number for uniqueness
     63   02E7'CF   02DF'CF   28  05DD  1386          MOVC3   DOTDAT,DOTDAT+8,(R3)    ; Give the file a .DAT type
          58   0291'CF   01   81  05E5  1387          ADDB3   #1,UETSPRGS,R8          ; Set the FNS field in the FAB...
               58   01FA'CF   80  05EB  1388          ADDB2   SCSNODE_LENGTH,R8       ; ...
               58   02CE'CF   80  05F0  1389          ADDB2   SYS$TEST,R8             ; ...
     34 A7     02DF'CF   58   81  05F5  1390          ADDB3   R8,DOTDAT,FAB$B_FNS(R7) ; ...
                0490 C6   DE  05FC  1391          MOVAL   MA_T_CHKFAB(R6),-       ; Set the FNA field in the FAB
                  2C A7        0600  1392                  FAB$L_FNA(R7)
                264C C6   DE  0602  1393          MOVAL   MA_K_CHKBUF(R6),-       ; Set UBF field
                0604 C6        0606  1394                  MA_K_CHKRAB+RAB$L_UBF(R6)
                              0609  1395
                              0609  1396
                   FCF0   31  0609  1397          BRW     FOUND_IT                ; Do the next UCB
```

```
                                060C  1399 ;+
                                060C  1400 ;       Arrive here when we have the device configuration.  Set up those things
                                060C  1401 ;       which need to be done before actual testing begins (file creation).
                                060C  1402 ;       In normal or loop forever mode, set a timer far enough in the future
                                060C  1403 ;       such that we can do a reasonable set of tests before the timer expires,
                                060C  1404 ;       but if our device gets hung, the program won't waste too much time
                                060C  1405 ;       before noticing.  Let one-shot mode be a special case.
                                060C  1406 ;-
                                060C  1407
                                060C  1408 ALL_SET:
          01D8'CF    D5         060C  1409         TSTL    UNIT_LIST                ; Anything to test?
              16     12         0610  1410         BNEQ    10$                      ; BR if yes
          010A'CF    DF         0612  1411         PUSHAL  NOUNIT_SELECTED          ; Else set up the error message...
              01     DD         0616  1412         PUSHL   #1                       ; ...argument count...
        00741132     8F   DD    0618  1413         PUSHL   #UETP$_TEXT!STS$K_ERROR  ; ...signal name...
              03     DD         061E  1414         PUSHL   #3                       ; ...and parameter count
      018A'CF    14     D0      0620  1415         MOVL    #SS$_BADPARAM,STATUS     ; Set return status
            1A78     31         0625  1416         BRW     ERROR_EXIT               ; ...and give up, complaining
                                0628  1417 10$:
      0002'CF    04     A8      0628  1418         BISW2   #SAFE_TO_UPDM,FLAG       ; OK safe to update UETINIDEV.DAT now
                                062D  1419         $SETPRV_S ENBFLG = #1,-          ; Set up needed privileges
                                062D  1420                   PRVADR = NEEDED_PRIVS
                                063E  1421
                                063E  1422 ;
                                063E  1423 ; There are some fields in the interprocessor global section which will be the
                                063E  1424 ; same for each shared memory, whatever port we're connected to.  Set those
                                063E  1425 ; up first.
                                063E  1426 ;
    0289'CF    0281'CF    28    063E  1427         MOVC3   UETCOMGS,UETCOMGS+8,-    ; Set up those things in the...
                  0209'CF       0645  1428                 COMGS_BUF+GS_T_SECT_NAME+1 ; (section name)
          0281'CF    01     81  0648  1429         ADDB3   #1,UETCOMGS,-            ; ...interproc GS buffer which are...
                  0208'CF       064D  1430                 COMGS_BUF+GS_T_SECT_NAME ; (section name length)
           30390004  8F     D0  0650  1431         MOVL    #CONSIS_CHECK,-          ; ...the same for all memories
                  0218'CF       0656  1432                 COMGS_BUF+GS_L_VERSION   ; (version consistency)
                                0659  1433 ; NOTE:  When this test was written, only 11/780 CPUs had shared memory.  Since
                                0659  1434 ; we are here extracting the ECO level from a processor dependent register, the
                                0659  1435 ; test must be modified should other CPUs eventually run it.  It has already
                                0659  1436 ; been modified for 11/785 CPUs.
    0224'CF    08     0F     EF  0659  1437         EXTZV   #15,#8,COMGS_BUF+GS_L_ECOLEVEL,- ; Get 780 processor ECO level
                  0224'CF       065F  1438                 COMGS_BUF+GS_L_ECOLEVEL
                                0662  1439
                                0662  1440 ;
                                0662  1441 ; For each memory that we're to test, fill in the fields in the interprocessor
                                0662  1442 ; global section that are memory and port specific.  Create the data structures
                                0662  1443 ; that will always be present, regardless of the presence of other cooperating
                                0662  1444 ; processors on that memory.
                                0662  1445 ;
56  01D8'CF     000001D8'8F    C1  0662  1446         ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; R6 points to node for current memory
                                066C  1447 20$:
          57    0208'CF    9A    066C  1448         MOVZBL  COMGS_BUF+GS_T_SECT_NAME,R7 ; Tack on the last byte...
                01F0 C6    90  0671  1449         MOVB    MA_T_UNIT(R6),-          ; ...to the global section name
                  0208'C7       0675  1450                 COMGS_BUF+GS_T_SECT_NAME(R7)
                01AC C6    9A  0678  1451         MOVZBL  MA_B_MYPORT(R6),-        ; Indicate which processor created...
                  0228'CF       067C  1452                 COMGS_BUF+GS_L_OWNER     ; ...this section in this memory
          01    0228'CF    78  067F  1453         ASHL    COMGS_BUF+GS_L_OWNER,#1,- ; Set the bit which says that...
                  022C'CF       0684  1454                 COMGS_BUF+GS_L_AVAILABLE ; ...we're a cooperating processor
           00000200  8F     7A  0687  1455         EMUL    #COMGS_PVT_AREA,-        ; Point to start of my private area...
```

```
                0228'CF        068D  1456                    COMGS_BUF+GS_L_OWNER,-
            57  000008B0'8F    0690  1457                    #COMGS_BUF+GS_R_PVT_AREAS,R7
                  51    57  D0  0696  1458          MOVL      R7,R1
            50  0000007E 8F    D0  0699  1459          MOVL      #COMGS_PVT_CODE/4,R0    ; ...and fill it...
                      0B9A  30    06A0  1460          BSBW      RANDOM_FILL             ; ...with random trash
                   01A4 C6  7D    06A3  1461          MOVQ      MA_Q_DAYTIME(R6),-      ; Set the special message, used as...
                   01F8 C7        06A7  1462                    COMGS_PVT_CODE(R7)      ; ...a code to identify my processor
                                  06AA  1463
                                  06AA  1464          $CREATE   FAB = MA_K_COMFAB(R6),- ; Set up file for interprocessor GS
                                  06AA  1465                    ERR = RMS_ERROR
                                  06B9  1466          $CONNECT  RAB = MA_K_COMRAB(R6),-
                                  06B9  1467                    ERR = RMS_ERROR
                                  06C8  1468          $WRITE    RAB = MA_K_COMRAB(R6),-
                                  06C8  1469                    ERR = RMS_ERROR
                                  06D7  1470          $CLOSE    FAB = MA_K_COMFAB(R6),-
                                  06D7  1471                    ERR = RMS_ERROR
                00020000 8F    C8 06E6  1472          BISL2     #FAB$M_UFO,-            ; Turn on the UFO bit...
                   0114 C6        06EC  1473                    MA_K_COMFAB+FAB$L_FOP(R6) ; ...so RMS sets us up OK later
                                  06EF  1474
                                  06EF  1475          $CREATE   FAB = MA_K_SPRFAB(R6),- ; Set up file for single processor GS
                                  06EF  1476                    ERR = RMS_ERROR
                                  06FE  1477          $CONNECT  RAB = MA_K_SPRRAB(R6),-
                                  06FE  1478                    ERR = RMS_ERROR
                                  070D  1479          $WRITE    RAB = MA_K_SPRRAB(R6),-
                                  070D  1480                    ERR = RMS_ERROR
                                  071C  1481          $CLOSE    FAB = MA_K_SPRFAB(R6),-
                                  071C  1482                    ERR = RMS_ERROR
                00020000 8F    C8 072B  1483          BISL2     #FAB$M_UFO,-            ; Turn on the UFO bit...
                   0400 C6        0731  1484                    MA_K_SPRFAB+FAB$L_FOP(R6) ; ...so RMS sets us up OK later
                                  0734  1485
                                  0734  1486          $CREMBX_S CHAN   = MA_W_SPRMBCHN(R6),- ; Create my mailbox
                                  0734  1487                    LOGNAM = MA_Q_SPRMBNAM(R6)
                                  074B  1488          $QIO_S    CHAN   = MA_W_SPRMBCHN(R6),- ; Be ready to read if...
                                  074B  1489                    FUNC   = #IO$_READVBLK,-    ; ...someone writes...
                                  074B  1490                    IOSB   = AST_MBISB,-        ; ...to my mailbox
                                  074B  1491                    ASTADR = RCVMB_AST,-
                                  074B  1492                    ASTPRM = R6,-
                                  074B  1493                    P1     = AST_MBBUF,-
                                  074B  1494                    P2     = #MB_K_END
                                  0772  1495
                      56  66    C0  0772  1496          ADDL2     (R6),R6                 ; Point to node for next memory
                56  000001D8'8F    D1  0775  1497          CMPL      #UNIT_LIST,R6           ; Back at the start of the queue?
                         03      13  077C  1498          BEQL      30$                     ; BR if we are - we're finished
                       FEEB      31  077E  1499          BRW       20$                     ; More to go so loop
                                  0781  1500 30$:
             13 0002'CF    04    E0  0781  1501          BBS       #ONE_SHOTV,FLAG,RESTART ; In one-shot mode start testing now...
                                  0787  1502                                            ; ...else fall into TIME_IT
                                  0787  1503 TIME_IT:
                                  0787  1504          $SETIMR_S DAYTIM = THREEMIN,-      ; Set timer AST to 3 minutes
                                  0787  1505                    ASTADR = TIME_OUT,-
                                  0787  1506                    EFN    = #EFN2
```

```
                                079A  1508          .SBTTL  Test the Multiport Memory
                                079A  1509
                                079A  1510 RESTART:
                                079A  1511 ;+
                                079A  1512 ;       Start the normal test by mapping to the interprocessor global section
                                079A  1513 ;       and announcing our presence.
                                079A  1514 ;-
                                079A  1515
          03 0002'CF    04    E1  079A  1516          BBC     #ONE_SHOTV,FLAG,10$     ; Skip all interprocessor stuff...
                        00E7    31  07A0  1517          BRW     ONEPROC_GS              ; ...if we are in one-shot mode
                                07A3  1518 10$:
    56  01D8'CF    000001D8'8F    C1  07A3  1519          ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; R6 points to node for current memory
                                07AD  1520 20$:
                                07AD  1521 ;
                                07AD  1522 ; Map to the interprocessor global section for this memory.  If it turns out we
                                07AD  1523 ; also create it, we're ready to handle messages from other processors which
                                07AD  1524 ; may subsequently map to it.
                                07AD  1525 ;
                                07AD  1526          $OPEN   FAB = MA_K_COMFAB(R6),- ; Reopen its interprocessor GS file
                                07AD  1527                  ERR = RMS_ERROR
              0624 C6   01    D0  07BC  1528          MOVL    #1,MA_Q_COMGSIAD(R6)    ; Say that file maps to P0 space
                                07C1  1529          $CRMPSC_S INADR = MA_Q_COMGSIAD(R6),- ; Map that file to a GS
                                07C1  1530                    RETADR = MA_Q_COMGSRAD(R6),-
                                07C1  1531                    FLAGS  = #SEC$M_GBL!SEC$M_EXPREG!SEC$M_WRT!SEC$M_PERM,-
                                07C1  1532                    PAGCNT = #<GS_K_END+511>/512,-
                                07C1  1533                    GSDNAM = MA_Q_COMGSNAM(R6),-
                                07C1  1534                    CHAN   = MA_K_COMFAB+FAB$L_STV(R6)
          57   062C C6    D0  07EB  1535          MOVL    MA_Q_COMGSRAD(R6),R7    ; Save GS's starting address in our P0
        50    00000619 8F    D1  07F0  1536          CMPL    #SS$_CREATED,R0         ; Did we create-and-map or just map?
                        03    12  07F7  1537          BNEQ    30$
                      007F    31  07F9  1538          BRW     90$                     ; BR if we created
                                07FC  1539 30$:
                                07FC  1540 ;
                                07FC  1541 ; We've mapped to an existing section.  Check that we don't already appear in
                                07FC  1542 ; the mask of cooperating processors.  Consistency check that we've got a good
                                07FC  1543 ; section.  Check for matching ECO levels of our CPU versus the CPU which
                                07FC  1544 ; created the global section.  Set up our private area and announce us to the
                                07FC  1545 ; world.
                                07FC  1546 ;
              01AC C6   E7      07FC  1547          BBCCI   MA_B_MYPORT(R6),-       ; BR if we're not already here
                    08 24 A7    0800  1548                  GS_L_AVAILABLE(R7),40$
          50    02F3'CF   DE    0803  1549          MOVAL   ALREADY_HERE,R0
                      0A7B    30  0808  1550          BSBW    DO_FAO_SIGNAL           ; Warn of possible consistency problems
                                080B  1551 40$:
    0289'CF     0281'CF   29    080B  1552          CMPC3   UETCOMGS,UETCOMGS+8,-    ; Did we find the correct...
                      01 A7     0812  1553                  GS_T_SECT_NAME+1(R7)     ; ...global section name?
                         11     12  0814  1554          BNEQ    50$                     ; BR if not
          63    01F0 C6   91    0816  1555          CMPB    MA_T_UNIT(R6),(R3)      ; More checking of GS name
                         0A     12  081B  1556          BNEQ    50$                     ; Again, BR if not as expected
    10 A7    30390004 8F    D1  081D  1557          CMPL    #CONSIS_CHECK,GS_L_VERSION(R7) ; Version and max ports match?
                         08     13  0825  1558          BEQL    60$                     ; BR if they do - we're consistent
                                0827  1559 50$:
          50    036C'CF   DE    0827  1560          MOVAL   INCONSISTENCY,R0
                      0A77    30  082C  1561          BSBW    DO_FAO_EXIT             ; We got the wrong GS - die
                                082F  1562 60$:
                0224'CF   D1    082F  1563          CMPL    COMGS_BUF+GS_L_ECOLEVEL,- ; Do we have the same CPU...
                      1C A7     0833  1564                  GS_L_ECOLEVEL(R7)        ; ...as the CPU which created GS?
```

```
                08   13  0835  1565          BEQL    70$                             ; BR if we do
      50  071D'CF    DE  0837  1566          MOVAL   BAD_ECO,R0
            0A47   30  083C  1567          BSBW    DO_FAO_SIGNAL                   ; Yell if we don't
                       083F  1568 70$:
      021C'CF   08   29  083F  1569          CMPC3   #8,COMGS_BUF+GS_Q_SYSGQVERSION,-
           14 A7       0844  1570                  GS_Q_SYSGQVERSION(R7)           ; Do we match on VAX/VMS, too?
                08   13  0846  1571          BEQL    80$                             ; BR if we do
      50  075E'CF   DE  0848  1572          MOVAL   WRONG_VERSION,R0
            0A56   30  084D  1573          BSBW    DO_FAO_EXIT                     ; Die if we don't
                       0850  1574 80$:
      58  06A8 C7   DE  0850  1575          MOVAL   GS_K_PVT_AREAS(R7),R8
       00000200 8F   7A  0855  1576          EMUL    #COMGS_PVT_AREA,-               ; Point to start of my private area...
58    58  01AC C6       085B  1577                  MA_B_MYPORT(R6),R8,R8
          51   58   D0  0860  1578          MOVL    R8,R7
 50    0000007E 8F   D0  0863  1579          MOVL    #COMGS_PVT_CODE/4,R0            ; ...and fill it...
            09D0   30  086A  1580          BSBW    RANDOM_FILL                     ; ...with random trash
           01A4 C6   7D  086D  1581          MOVQ    MA_Q_DAYTIME(R6),-              ; Set the special message, used as...
           01F8 C8       0871  1582                  COMGS_PVT_CODE(R8)              ; ...a code to identify my processor
           01AC C6   E6  0874  1583          BBSSI   MA_B_MYPORT(R6),-               ; Let other processors know of us!
          00 24 A7       0878  1584                  GS_L_AVAILABLE(R7),90$
                       087B  1585 90$:
          56   66   C0  087B  1586          ADDL2   (R6),R6                         ; Point to node for next memory
56    000001D8'8F   D1  087E  1587          CMPL    #UNIT_LIST,R6                   ; Back at the start of the queue?
                03   13  0885  1588          BEQL    100$                            ; BR if we are - we're finished
            FF23   31  0887  1589          BRW     20$                             ; More to go so loop
                       088A  1590 100$:
```

```
                                088A  1592 ONEPROC_GS:
                                088A  1593 ;+
                                088A  1594 ;       Play with the single processor global section.  Create and map to a
                                088A  1595 ;       global section in each shared memory which is used exclusively by our
                                088A  1596 ;       processor.  Modify the contents and update each section back to disk.
                                088A  1597 ;       At AST level, confirm that the updates succeeded and delete each
                                088A  1598 ;       section and its address space.
                                088A  1599 ;-
                                088A  1600
    56  01D8'CF   000001D8'8F   C1  088A  1601         ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; R6 points to node for current memory
                                0894  1602 10$:
                                0894  1603 ;
                                0894  1604 ; Map to the single processor global section in this memory.  If we only map,
                                0894  1605 ; but not create it, give a warning.  To prevent possible conflicts with
                                0894  1606 ; previous passes through this code, make sure that the $UPDSEC-in-progress
                                0894  1607 ; flag is set to allow us to proceed.
                                0894  1608 ;
                                0894  1609         $WAITFR_S EFN = #SPRGS_EFN      ; Allow possible $UPDSEC to finish
                                089D  1610         $OPEN   FAB = MA_K_SPRFAB(R6),- ; Reopen single processor GS file
                                089D  1611                 ERR = RMS_ERROR
              0634 C6   01      D0  08AC  1612         MOVL    #1,MA_Q_SPRGSIAD(R6)    ; Say that file maps to P0 space
                                08B1  1613         $CRMPSC_S INADR = MA_Q_SPRGSIAD(R6),- ; Map that file to a GS
                                08B1  1614                   RETADR = MA_Q_SPRGSRAD(R6),-
                                08B1  1615                   FLAGS  = #SEC$M_GBL!SEC$M_EXPREG!SEC$M_WRT!SEC$M_PERM,-
                                08B1  1616                   PAGCNT = #SPRGS_SIZE,-
                                08B1  1617                   GSDNAM = MA_Q_SPRGSNAM(R6),-
                                08B1  1618                   CHAN   = MA_K_SPRFAB+FAB$L_STV(R6)
              57  063C C6      D0  08DB  1619         MOVL    MA_Q_SPRGSRAD(R6),R7    ; Save GS's starting address in our P0
           50  00000619 8F      D1  08E0  1620         CMPL    #SS$_CREATED,R0         ; Did we create-and-map or just map?
                        08      13  08E7  1621         BEQL    20$                     ; BR if we created
              50  03B2'CF       DE  08E9  1622         MOVAL   OLD_ONEPROC,R0
                  0995          30  08EE  1623         BSBW    DO_FAO_SIGNAL           ; We've a rusty old single proc GS - warn
                                08F1  1624 20$:
                                08F1  1625 ;
                                08F1  1626 ; Use a bufferful of pseudo-random data as a means to verify that the section
                                08F1  1627 ; can be written to correctly and then updated correctly to disk.  Only issue
                                08F1  1628 ; the $UPDSEC here; the actual verification will take place at AST level.  That
                                08F1  1629 ; will allow us to test the next memory or otherwise proceed with the test.
                                08F1  1630 ;
              51  064C C6       DE  08F1  1631         MOVAL   MA_K_SPRBUF(R6),R1      ; Fill a buffer...
        50  00000800 8F          D0  08F6  1632         MOVL    #SPRGS_SIZE*128,R0      ; ...of longwords...
                  093D          30  08FD  1633         BSBW    RANDOM_FILL             ; ...with random trash
                2000 8F         28  0900  1634         MOVC3   #SPRGS_SIZE*512,-       ; Move that trash to the...
              67  064C C6             0904  1635                 MA_K_SPRBUF(R6),(R7)    ; ...single processor global section
                2000 8F         29  0908  1636         CMPC3   #SPRGS_SIZE*512,-       ; Make sure that the pages...
              67  064C C6             090C  1637                 MA_K_SPRBUF(R6),(R7)    ; ...all get updated
                        49      13  0910  1638         BEQL    30$                     ; BR if all data transferred
                        02      8A  0912  1639         BICB2   #UETUNT$M_TESTABLE,-    ; The MPM is not testable...
                     0B A6              0914  1640                 UETUNT$B_FLAGS(R6)      ; ...(as far as one-shot mode thinks)
                7E      63      9A  0916  1641         MOVZBL  (R3),-(SP)              ; Not written OK. Save the bad byte...
                7E      61      9A  0919  1642         MOVZBL  (R1),-(SP)              ; ...the corresponding good byte...
              7E  53    57      C3  091C  1643         SUBL3   R7,R3,-(SP)             ; ...where in the buffer it was...
                01F1 C6         DF  0920  1644         PUSHAL  MA_Q_MEMNAM(R6)         ; ...the memory name...
                        04      DD  0924  1645         PUSHL   #4                      ; ...the argument count...
                 0074801A 8F    DD  0926  1646         PUSHL   #UETP$_DATADEVERR!STS$K_ERROR ; ...and the error type
                    0402'CF     DF  092C  1647         PUSHAL  BAD_MOVC3               ; Tell what the error was from
                 000F0001 8F    DD  0930  1648         PUSHL   #^XF0001
```

```
                                                B 3
                00741132 8F    DD  0936  1649          PUSHL   #UETP$_TEXT!STS$K_ERROR
                    0186'CF    D6  093C  1650          INCL    ERROR_COUNT             ; Keep a running count...
                    0186'CF    DD  0940  1651          PUSHL   ERROR_COUNT             ; ...of the errors we've gotten
                    00E4'CF    DF  0944  1652          PUSHAL  PROCESS_NAME
                00010002 8F    DD  0948  1653          PUSHL   #^X10002
                00748022 8F    DD  094E  1654          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; Have the error stand out
            00000000'GF    0D  FB  0954  1655          CALLS   #13,G^LIB$SIGNAL
                               095B  1656 30$:
                               095B  1657          $UPDSEC_S INADR = MA_Q_SPRGSRAD(R6),- ; Update the section to disk
                               095B  1658                    UPDFLG = #1,-
                               095B  1659                    EFN    = #SPRGS_EFN,-       ; (see note where SPRGS_EFN defined)
                               095B  1660                    IOSB   = MA_Q_SPRGSISB(R6),-
                               095B  1661                    ASTADR = SPRGS_AST,-
                               095B  1662                    ASTPRM = R6
                    56    66   C0  0978  1663          ADDL2   (R6),R6                 ; Point to node for next memory
                56  000001D8'8F   D1  097B  1664          CMPL    #UNIT_LIST,R6           ; Back at the start of the queue?
                          03   13  0982  1665          BEQL    40$                     ; BR if we are - we're finished
                        FFOD   31  0984  1666          BRW     10$                     ; More to go so loop
                               0987  1667 40$:
```

```
                                        0987  1669 ONEPROC_MB:
                                        0987  1670 ;+
                                        0987  1671 ;       Send mail to my own processor.  Synchronize to ensure that it gets
                                        0987  1672 ;       received properly.
                                        0987  1673 ;-
                                        0987  1674
    56   01DB'CF   000001D8'8F   C1     0987  1675          ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; R6 points to node for current memory
                                        0991  1676 10$:
                 5A   01AC C6   9A     0991  1677          MOVZBL  MA_B_MYPORT(R6),R10              ; Start filling in message...
                 10B0'CF   5A   90     0996  1678          MOVB    R10,SYNCH_MBBUF+MB_B_SENDER      ; ...sender's port number...
                 10B1'CF   5A   90     099B  1679          MOVB    R10,SYNCH_MBBUF+MB_B_RECEIVER    ; ...receiver's port number...
                      01EC C6   90     09A0  1680          MOVB    MA_W_UNIT(R6),-                  ; ...unit no. of shared memory...
                      10B2'CF          09A4  1681                  SYNCH_MBBUF+MB_B_UNIT
                           0A   B0     09A7  1682          MOVW    #MSG_ME2ME,-                     ; ...reason for the message
                      10B3'CF          09A9  1683                  SYNCH_MBBUF+MB_W_WHY
                                       09AC  1684          $CREMBX_S CHAN = SYNCH_MBCHN,-  ; Establish another link to my MB
                                       09AC  1685                    LOGNAM = MA_Q_SPRMBNAM(R6)
          1102'CF    FECED300 8F   D0     09C3  1686          MOVL    #ME2ME_TIMEOUT,SYNCH_MBDAY ; $SETIMR parameters - DAYTIM...
                 1106'CF   00   D2     09CC  1687          MCOML   #0,SYNCH_MBDAY+4                 ; ...
               54   10   10   0A   F0     09D1  1688          INSV    #MSG_ME2ME,#16,#16,R4            ; ...REQIDT...
               54   08   08   5A   F0     09D6  1689          INSV    R10,#8,#8,R4                     ; ...
                    54   01EC C6   90     09DB  1690          MOVB    MA_W_UNIT(R6),R4                 ; ...
                                       09E0  1691          $SETIMR_S DAYTIM = SYNCH_MBDAY,-     ; A timer will let us know...
                                       09E0  1692                    ASTADR = MB_TIMEOUT_AST,- ; ...if my processor...
                                       09E0  1693                    REQIDT = R4               ; ...doesn't reply to the following
                                       09F3  1694          $QIO_S  CHAN    = SYNCH_MBCHN,-  ; Talk to myself
                                       09F3  1695                  FUNC    = #IO$_WRITEVBLK!IO$M_NOW,-
                                       09F3  1696                  IOSB    = SYNCH_MBISB,-
                                       09F3  1697                  P1      = SYNCH_MBBUF,-
                                       09F3  1698                  P2      = #MB_K_END
                          54   DD     0A16  1699          PUSHL   R4
                 1422'CF   01   FB     0A18  1700          CALLS   #1,QIOMB_SYNCH_CHECK     ; Check the completion status
                                       0A1D  1701          $DASSGN_S CHAN = SYNCH_MBCHN     ; We no longer need the MB
                                       0A29  1702 ;
                                       0A29  1703 ; In order to make sure that we have time to receive the mail, we'll hibernate
                                       0A29  1704 ; for a while.  If the mail is received correctly, the mailbox receipt routine
                                       0A29  1705 ; will change the MB_W_WHY field to say so.  If the buffer is unchanged,
                                       0A29  1706 ; assume the transfer failed and we timed out.
                                       0A29  1707 ;
                                       0A29  1708          $HIBER_S                         ; Wait for mailbox AST or time out
                           0B   B1     0A30  1709          CMPW    #MSG_ME2MEDONE,-         ; Message acknowledged?
                      10B3'CF          0A32  1710                  SYNCH_MBBUF+MB_W_WHY
                           08   13     0A35  1711          BEQL    30$                      ; BR if it was
                50   0531'CF   DE     0A37  1712          MOVAL   ME2ME_FAILED,R0
                        0867   30     0A3C  1713          BSBW    DO_FAO_EXIT              ; Die if it wasn't
                                       0A3F  1714 30$:
                    56   66   C0     0A3F  1715          ADDL2   (R6),R6                  ; Point to node for next memory
          56   000001D8'8F   D1     0A42  1716          CMPL    #UNIT_LIST,R6            ; Back at the start of the queue?
                           03   13     0A49  1717          BEQL    40$                      ; BR if we are - we're finished
                      FF43   31     0A4B  1718          BRW     10$                      ; More to go so loop
                                       0A4E  1719 40$:
```

```
                        0A4E  1721 ONEPROC_EF:
                        0A4E  1722 ;+
                        0A4E  1723 ;       Play with the single processor common event flag cluster.  Associate
                        0A4E  1724 ;       with a cluster in each shared memory and test to see that all bits in
                        0A4E  1725 ;       each can be set and cleared when used exclusively by our processor.
                        0A4E  1726 ;       Disassociate from the clusters.
                        0A4E  1727 ;-
                        0A4E  1728
    56  01D8'CF  000001D8'8F  C1  0A4E  1729          ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; R6 points to node for current memory
                        0A58  1730 10$:
                        0A58  1731          $ASCEFC_S EFN = #SPREF_EFN,-     ; Associate with a CEF cluster
                        0A58  1732                    NAME = MA_Q_SPREFNAM(R6)
                        0A6D  1733 ;
                        0A6D  1734 ; Set and clear each flag in sequence.
                        0A6D  1735 ;
            57  00000040 8F  D0  0A6D  1736          MOVL    #SPREF_EFN,R7           ; Initialize EF position counter
                  58    01  D0  0A74  1737          MOVL    #1,R8                   ; Initialize EF mask
                        0A77  1738 20$:
                        0A77  1739          $SETEF_S  EFN   = R7             ; Set a common event flag
                        0A80  1740          $READEF_S EFN   = R7,-          ; Read back the cluster
                        0A80  1741                    STATE = MA_L_SPREFSTATE(R6)
                     00AA  30  0A8D  1742          BSBW    100$                    ; Check that exactly our flags are set
                        0A90  1743          $CLREF_S  EFN   = R7             ; Clear what we just set
                        0A99  1744          $READEF_S EFN   = R7,-          ; Read back the cluster
                        0A99  1745                    STATE = MA_L_SPREFSTATE(R6)
                     009D  30  0AA6  1746          BSBW    200$                    ; Check that our flag is clear
                       57  D6  0AA9  1747          INCL    R7                      ; Point to the next flag in the cluster
               58   58    01  78  0AAB  1748          ASHL    #1,R8,R8                ; Mask the next flag in the cluster
                       C6  12  0AAF  1749          BNEQ    20$                     ; BR if there are more bits to test
                        0AB1  1750 ;
                        0AB1  1751 ; Set each flag, one at a time, until the entire cluster is set.
                        0AB1  1752 ;
    57  00000040 8F  D0  0AB1  1753          MOVL    #SPREF_EFN,R7           ; Initialize EF position counter
                  58    D4  0AB8  1754          CLRL    R8                      ; Initialize total EF mask
                  59    01  D0  0ABA  1755          MOVL    #1,R9                   ; Initialize individual EF mask
                        0ABD  1756 30$:
                  58    59  C8  0ABD  1757          BISL2   R9,R8                   ; Mask next flag to set in the cluster
                        0AC0  1758          $SETEF_S  EFN   = R7             ; Set a common event flag
                        0AC9  1759          $READEF_S EFN   = R7,-          ; Read back the cluster
                        0AC9  1760                    STATE = MA_L_SPREFSTATE(R6)
                     0061  30  0AD6  1761          BSBW    100$                    ; Check that exactly our flags are set
                       57  D6  0AD9  1762          INCL    R7                      ; Point to the next flag in the cluster
           59   59    01  78  0ADB  1763          ASHL    #1,R9,R9                ; Mask the next flag in the cluster
                       DC  12  0ADF  1764          BNEQ    30$                     ; BR if there are more bits to test
                        0AE1  1765 ;
                        0AE1  1766 ; Clear each flag, one at a time, until the entire cluster is clear.
                        0AE1  1767 ;
    57  00000040 8F  D0  0AE1  1768          MOVL    #SPREF_EFN,R7           ; Initialize EF position counter
                  58    00  D2  0AE8  1769          MCOML   #0,R8                   ; Initialize total EF mask
                  59    01  D0  0AEB  1770          MOVL    #1,R9                   ; Initialize individual EF mask
                        0AEE  1771 40$:
                  58    59  CA  0AEE  1772          BICL2   R9,R8                   ; Mask all flags set in the cluster
                        0AF1  1773          $CLREF_S  EFN   = R7             ; Clear a common event flag
                        0AFA  1774          $READEF_S EFN   = R7,-          ; Read back the cluster
                        0AFA  1775                    STATE = MA_L_SPREFSTATE(R6)
                     0030  30  0B07  1776          BSBW    100$                    ; Check that exactly our flags are clear
                       57  D6  0B0A  1777          INCL    R7                      ; Point to the next flag in the cluster
```

```
                59   59    01    78  0B0C  1778          ASHL    #1,R9,R9                ; Mask the next flag in the cluster
                           DC    12  0B10  1779          BNEQ    40$                     ; BR if there are more bits to test
                                     0B12  1780
                                     0B12  1781          $DACEFC_S EFN = #SPREF_EFN      ; Disassociate (and delete) our CEF cluster
                     56    66    C0  0B1F  1782          ADDL2   (R6),R6                 ; Point to node for next memory
              56  000001D8'8F    D1  0B22  1783          CMPL    #UNIT_LIST,R6           ; Back at the start of the queue?
                           03    13  0B29  1784          BEQL    50$                     ; BR if we are - we're finished
                        FF2A    31  0B2B  1785          BRW     10$                     ; More to go so loop
                                     0B2E  1786 50$:
              03 0002'CF    04    E0  0B2E  1787          BBS     #ONE_SHOTV,FLAG,60$     ; BR if in one-shot mode
                        005D    31  0B34  1788          BRW     INTERPROC_GS            ; Skip over local subroutines to next test
                                     0B37  1789 60$:
                        0656    31  0B37  1790          BRW     SUC_EXIT                ; Only test single proc stuff if one-shot
```

```
                                0B3A  1792 ;
                                0B3A  1793 ; Subroutine to check that R8 matches exactly the common event flags in our
                                0B3A  1794 ; cluster.
                                0B3A  1795 ;
                                0B3A  1796 100$:
                     54   58    D0  0B3A  1797          MOVL    R8,R4                           ; Set up if error message needed
          5A   58    02CD C6   CD  0B3D  1798          XORL3   MA_L_SPREFSTATE(R6),R8,R10 ; Exact match?
                          11    12  0B43  1799          BNEQ    300$                            ; BR if not - issue error message
                          05    05  0B45  1800          RSB
                                0B46  1801
                                0B46  1802
                                0B46  1803 ;
                                0B46  1804 ; Subroutine to check that the flags masked in R8 are clear.
                                0B46  1805 ;
                                0B46  1806 200$:
                     54   D4    0B46  1807          CLRL    R4                              ; Set up if error message needed...
          5A   58    02CD C6   CD  0B48  1808          XORL3   MA_L_SPREFSTATE(R6),R8,R10 ; ...
               02CD C6    58   D3  0B4E  1809          BITL    R8,MA_L_SPREFSTATE(R6)  ; Is requested bit clear?
                          01    12  0B53  1810          BNEQ    300$                            ; BR if not - issue error message
                          05    05  0B55  1811          RSB
                                0B56  1812
                                0B56  1813
                                0B56  1814 ;
                                0B56  1815 ; Subroutine which, given expected flags to be set in R4, actual flags set in
                                0B56  1816 ; MA_L_SPREFSTATE(R6) and the difference between the two in R10, prints an
                                0B56  1817 ; error message.
                                0B56  1818 ;
                                0B56  1819 300$:
                          02    8A  0B56  1820          BICB2   #UETUNT$M_TESTABLE,-    ; The MPM is not testable...
                       0B A6        0B58  1821                  UETUNT$B_FLAGS(R6)      ; ...(as far as one-shot mode thinks)
               55   01F1 C6    DE  0B5A  1822          MOVAL   MA_Q_MEMNAM(R6),R5      ; Form $FAO parameter
                                0B5F  1823          $FAO_S  CTRSTR = BAD_CEF,-      ; Form an error message
                                0B5F  1824                  OUTLEN = BUFFER_PTR,-
                                0B5F  1825                  OUTBUF = FAO_BUF,-
                                0B5F  1826                  P1     = R5,-
                                0B5F  1827                  P2     = MA_W_UNIT(R6),-
                                0B5F  1828                  P3     = R4,-
                                0B5F  1829                  P4     = MA_L_SPREFSTATE(R6),-
                                0B5F  1830                  P5     = R10
                     000C'CF    DF  0B80  1831          PUSHAL  BUFFER_PTR
                          01    DD  0B84  1832          PUSHL   #1
                00741132  8F    DD  0B86  1833          PUSHL   #UETP$_TEXT!STS$K_ERROR
          00000000'GF   03    FB  0B8C  1834          CALLS   #3,G^LIB$SIGNAL
                          05    05  0B93  1835          RSB
```

```
                                0B94  1837 INTERPROC_GS:
                                0B94  1838 ;+
                                0B94  1839 ;       Start the interprocessor communications part of the shared memory test.
                                0B94  1840 ;       Mail to each processor which appears in the "available" mask in the
                                0B94  1841 ;       interprocessor global section, a message which includes the date-time
                                0B94  1842 ;       quadword at the end of that section's private area in the
                                0B94  1843 ;       interprocessor global section.  We will expect a reply to our mailbox,
                                0B94  1844 ;       to be received at AST level.  Set a timer to catch the case where
                                0B94  1845 ;       another processor doesn't reply.
                                0B94  1846 ;-
                                0B94  1847
                                0B94  1848 ; NOTE WELL!  Within the following loop, registers R6 through R10 are set and
                                0B94  1849 ; assumed to remain unchanged.
    56  01D8'CF   00000'D8'8F   C1  0B94  1850         ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; R6 points to node for current memory
                                    0B9E  1851 10$:
                57  062C C6     D0  0B9E  1852         MOVL    MA_Q_COMGSRAD(R6),R7     ; R7 will point into the interproc GS
                       58        D4  0BA3  1853         CLRL    R8                       ; R8 is STRTPOS for FFS operations
                    59   04      D0  0BA5  1854         MOVL    #PROCESSOR_MAX,R9        ; R9 is SIZE for FFS operations
                                    0BA8  1855 20$:
     5A   24 A7     59   58      EA  0BA8  1856         FFS     R8,R9,GS_L_AVAILABLE(R7),R10 ; Search for a willing processor
                       03        12  0BAE  1857         BNEQ    30$                      ; BR if we found one
                          00E2   31  0BB0  1858         BRW     50$                      ; BR if there are none left
                                    0BB3  1859 30$:
                01AC C6     90  0BB3  1860         MOVB    MA_B_MYPORT(R6),-                ; Start filling in message...
                10B0'CF             0BB7  1861                 SYNCH_MBBUF+MB_B_SENDER  ; ...sender's port number...
            10B1'CF    5A   90  0BBA  1862         MOVB    R10,SYNCH_MBBUF+MB_B_RECEIVER ; ...receiver's port number...
                01EC C6     90  0BBF  1863         MOVB    MA_W_UNIT(R6),-          ; ...unit number...
                10B2'CF             0BC3  1864                 SYNCH_MBBUF+MB_B_UNIT    ; ...of the memory we share...
                   01       B0  0BC6  1865         MOVW    #MSG_HELLO,-             ; ...reason for the message...
                10B3'CF             0BC8  1866                 SYNCH_MBBUF+MB_W_WHY
            5B  06A8 C7     DE  0BCB  1867         MOVAL   GS_K_PVT_AREAS(R7),R11
5B  5B  5A   00000200 8F    7A  0BD0  1868         EMUL    #COMGS_PVT_AREA,R10,R11,R11  ; ...(point to)...
                01F8 CB     7D  0BD9  1869         MOVQ    COMGS_PVT_CODE(R11),-    ; ...and the other processor's...
                10B5'CF             0BDD  1870                 SYNCH_MBBUF+MB_K_MISC    ; ...date/time
                0258 C6     28  0BE0  1871         MOVC3   MA_Q_COMMBNAM(R6),-      ; Form the logical name...
        10E3'CF 0260 C6         0BE4  1872                 MA_T_COMMBNAM(R6),SYNCH_MBNAM ; ...of other processor's MB
        10DB'CF 0258 C6     D0  0BEA  1873         MOVL    MA_Q_COMMBNAM(R6),SYNCH_MBPTR ; Form descriptor to...
        10DF'CF 10E3'CF     DE  0BF1  1874         MOVAL   SYNCH_MBNAM,SYNCH_MBPTR+4    ; ...the logical name
                   50   5A   D0  0BF8  1875         MOVL    R10,R0                   ; Include the port number...
                      0633   30  0BFB  1876         BSBW    HEXUNIT                  ; ...
                FF A3    50   90  0BFE  1877         MOVB    R0,-1(R3)                ; ...in the name
                                0C02  1878         $CREMBX_S CHAN = SYNCH_MBCHN,-   ; Establish a link to other proc's MB
                                0C02  1879                   LOGNAM = SYNCH_MBPTR
    1102'CF   FA0A1F00 8F       D0  0C19  1880         MOVL    #HELLO_TIMEOUT,SYNCH_MBDAY ; $SETIMR parameters - DAYTIM...
            1106'CF    00       D2  0C22  1881         MCOML   #0,SYNCH_MBDAY+4         ; ...
        54  10   10    01       F0  0C27  1882         INSV    #MSG_HELLO,#16,#16,R4    ; ...REQIDT...
        54  08   08    5A       F0  0C2C  1883         INSV    R10,#8,#8,R4             ; ...
            54   01EC C6    90  0C31  1884         MOVB    MA_W_UNIT(R6),R4         ; ...
                                0C36  1885         $SETIMR_S DAYTIM = SYNCH_MBDAY,-     ; A timer will let us know...
                                0C36  1886                   ASTADR = MB_TIMEOUT_AST,- ; ...if the other processor...
                                0C36  1887                   REQIDT = R4             ; ...doesn't reply to the following
     00 01DC C6    5A   E2  0C49  1888         BBSS    R10,MA_L_HELLO(R6),40$   ; Log that we expect reply to our mail
                                0C4F  1889 40$:
                                0C4F  1890         $QIO_S  CHAN   = SYNCH_MBCHN,-   ; Say "Hi!" to the other processor
                                0C4F  1891                 FUNC   = #IO$_WRITEVBLK!IO$M_NOW,-
                                0C4F  1892                 IOSB   = SYNCH_MBISB,-
                                0C4F  1893                 P1     = SYNCH_MBBUF,-
```

```
                     0C4F  1894                    P2      = #MB_K_END
                54   DD   0C72  1895          PUSHL   R4
      1422'CF    01   FB   0C74  1896          CALLS   #1,QIOMB_SYNCH_CHECK    ; Check the completion status
                           0C79  1897          $DASSGN_S CHAN = SYNCH_MBCHN    ; We no longer need the MB
   5B   5A   58   C3   0C85  1898          SUBL3   R8,R10,R11              ; Get number of bits we searched
             5B   D6   0C89  1899          INCL    R11                     ; Correct off-by-one (inclusive search)
        59   5B   C2   0C8B  1900          SUBL2   R11,R9                  ; Adjust number of bits left to search
   58   5A   01   C1   0C8E  1901          ADDL3   #1,R10,R8               ; Point to new first bit to search
           FF13   31   0C92  1902          BRW     20$                     ; Loop thru all procs on this memory
                      0C95  1903 50$:
        56   66   C0   0C95  1904          ADDL2   (R6),R6                 ; Point to node for next memory
56   000001D8'8F   D1   0C98  1905          CMPL    #UNIT_LIST,R6           ; Back at the start of the queue?
              03   13   0C9F  1906          BEQL    60$                     ; BR if we are - we're finished
            FEFA   31   0CA1  1907          BRW     10$                     ; More to go so loop
                      0CA4  1908 60$:
```

```
                        0CA4  1910 INTERPROC_EF:
                        0CA4  1911 ;+
                        0CA4  1912 ;       Start the shared common event flag test.  For this test to work, one
                        0CA4  1913 ;       processor must have control of the test across all processors.  To
                        0CA4  1914 ;       ensure that, there is a lock in the interprocessor global section which
                        0CA4  1915 ;       can be owned by only one process.  For each shared memory, try to grab
                        0CA4  1916 ;       the lock.  If that fails, try to grab the next memory's lock or proceed
                        0CA4  1917 ;       to the next part of the test.  If we do grab a lock, associate with a
                        0CA4  1918 ;       common event flag cluster in that memory.  For each processor connected
                        0CA4  1919 ;       to the memory, choose a flag in that cluster and send the process a
                        0CA4  1920 ;       mailbox message requesting the processor to set that bit in the
                        0CA4  1921 ;       cluster.  Allow a reasonable timeout for that to occur.  If the
                        0CA4  1922 ;       processor responds by setting the bit, pick another processor to clear
                        0CA4  1923 ;       the bit, again, allowing a reasonable timeout for that action.  Verify
                        0CA4  1924 ;       that the bits get set and cleared.  Release the lock in the shared
                        0CA4  1925 ;       memory.
                        0CA4  1926 ;-
                        0CA4  1927
                        0CA4  1928 ; NOTE WELL!  Within the following loop, registers R6 through R10 are set and
                        0CA4  1929 ; assumed to remain unchanged.
 56  01D8'CF  000001D8'8F  C1  0CA4  1930         ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; R6 points to node for current memory
                        0CAE  1931 10$:
           57  062C C6  D0  0CAE  1932         MOVL    MA_Q_COMGSRAD(R6),R7    ; R7 will point into the interproc GS
        08 28 A7   00  E6  0CB3  1933         BBSSI   #0,GS_L_CEF_FLAG(R7),20$ ; Try to grab the interlock
              01AC C6  90  0CB8  1934         MOVB    MA_B_MYPORT(R6),-       ; Bit was clear...
                 2B A7       0CBC  1935                 GS_L_CEF_FLAG+3(R7)
                    03  11  0CBE  1936         BRB     30$                     ; ...my port has the interlock
                        0CC0  1937 20$:
                  0278  31  0CC0  1938         BRW     150$                    ; Bit was set - someone else has it
                        0CC3  1939 30$:
                        0CC3  1940         $ASCEFC_S EFN = #COMEF_EFN,-     ; Associate with interprocessor cluster
                        0CC3  1941                   NAME = MA_Q_COMEFNAM(R6)
                     58  D4  0CD8  1942         CLRL    R8                      ; R8 is STRTPOS for FFS operations
                 59  04  D0  0CDA  1943         MOVL    #PROCESSOR_MAX,R9       ; R9 is SIZE for FFS operations
                        0CDD  1944 40$:
5A  24 A7        59  58  EA  0CDD  1945         FFS     R8,R9,GS_L_AVAILABLE(R7),R10 ; Search for a willing processor
                     03  12  0CE3  1946         BNEQ    50$                     ; BR if we found one
                   0243  31  0CE5  1947         BRW     140$                    ; BR if there are none left
                        0CE8  1948 50$:
              01AC C6  90  0CE8  1949         MOVB    MA_B_MYPORT(R6),-               ; Start filling in message...
              10B0'CF        0CEC  1950                 SYNCH_MBBUF+MB_B_SENDER         ; ...sender's port number...
        10B1'CF  5A  90  0CEF  1951         MOVB    R10,SYNCH_MBBUF+MB_B_RECEIVER ; ...receiver's port number...
              01EC C6  90  0CF4  1952         MOVB    MA_W_UNIT(R6),-                 ; ...unit no. of shared memory...
              10B2'CF        0CF8  1953                 SYNCH_MBBUF+MB_B_UNIT
                 05  B0  0CFB  1954         MOVW    #MSG_DOSET,-                    ; ...reason for the message...
              10B3'CF        0CFD  1955                 SYNCH_MBBUF+MB_W_WHY
   01AA'CF    01AE'CF  C0  0D00  1956         ADDL2   RANDOM2,RANDOM1                 ; ...
   01AA'CF    05   06  EF  0D07  1957         EXTZV   #6,#5,RANDOM1,-                 ; ...and a random bit to set
              10B5'CF        0D0D  1958                 SYNCH_MBBUF+MB_K_MISC
              0258 C6  28  0D10  1959         MOVC3   MA_Q_COMMBNAM(R6),-             ; Form the logical name...
   10E3'CF    0260 C6        0D14  1960                 MA_T_COMMBNAM(R6),SYNCH_MBNAM ; ...of other processor's MB
   10DB'CF    0258 C6  D0  0D1A  1961         MOVL    MA_Q_COMMBNAM(R6),SYNCH_MBPTR ; Form descriptor to...
   10DF'CF    10E3'CF  DE  0D21  1962         MOVAL   SYNCH_MBNAM,SYNCH_MBPTR+4       ; ...the logical name
              50   5A  D0  0D28  1963         MOVL    R10,R0                  ; Include the port number...
                 0503  30  0D2B  1964         BSBW    HEXUNIT                 ; ...
           FF A3   50  90  0D2E  1965         MOVB    R0,-1(R3)               ; ...in the name
```

```
                                 0D32  1967          $CREMBX_S CHAN = SYNCH_MBCHN,-  ; Establish a link to other proc's MB
                                 0D32  1968                    LOGNAM = SYNCH_MBPTR
      1102'CF    FA0A1F00 8F    D0  0D49  1969          MOVL    #DOSET_TIMEOUT,SYNCH_MBDAY ; $SETIMR parameters - DAYTIM...
               1106'CF    00    D2  0D52  1970          MCOML   #0,SYNCH_MBDAY+4           ; ...
           54    10    10    05  F0  0D57  1971          INSV    #MSG_DOSET,#16,#16,R4      ; ...REQIDT...
           54    08    08    5A  F0  0D5C  1972          INSV    R10,#8,#8,R4               ; ...
                 54    01EC C6   90  0D61  1973          MOVB    MA_W_UNIT(R6),R4           ; ...
                                 0D66  1974          $SETIMR_S DAYTIM = SYNCH_MBDAY,-    ; A timer will let us know...
                                 0D66  1975                    ASTADR = MB_TIMEOUT_AST,- ; ...if the other processor...
                                 0D66  1976                    REQIDT = R4               ; ...doesn't reply to the following
            00 01E4 C6    5A   E2  0D79  1977          BBSS    R10,MA_L_DOSET(R6),60$  ; Log that we expect reply to our mail
                                 0D7F  1978 60$:
                                 0D7F  1979          $QIO_S  CHAN    = SYNCH_MBCHN,-  ; Tell the other processor to set a bit
                                 0D7F  1980                  FUNC    = #IO$_WRITEVBLK!IO$M_NOW,-
                                 0D7F  1981                  IOSB    = SYNCH_MBISB,-
                                 0D7F  1982                  P1      = SYNCH_MBBUF,-
                                 0D7F  1983                  P2      = #MB_K_END
                       54    DD  0DA2  1984          PUSHL   R4
                1422'CF    01    FB  0DA4  1985          CALLS   #1,@IOMB_SYNCH_CHECK    ; Check the completion status
                                 0DA9  1986          $DASSGN_S CHAN = SYNCH_MBCHN    ; We no longer need the MB
                                 0DB5  1987 ;
                                 0DB5  1988 ; Now we must wait.  The logic of the interprocessor CEF test dictates that we
                                 0DB5  1989 ; know for sure that either the other processor successfully set the bit or we
                                 0DB5  1990 ; know for sure that it did not.  We will wait for either the receipt of a
                                 0DB5  1991 ; mailbox in reply (must happen at AST level) or the expiration of the timer.
                                 0DB5  1992 ; Note that on return, we cannot assume the contents of the mailbox received
                                 0DB5  1993 ; at AST level because another mailbox may have been received as well, however
                                 0DB5  1994 ; the mailbox receipt routine is smart enough to realize this and will move
                                 0DB5  1995 ; the contents of the reply we want into the synchronous mailbox buffer.  If
                                 0DB5  1996 ; the buffer is unchanged, assume the transfer failed and we timed out.
                                 0DB5  1997 ;
                                 0DB5  1998          $HIBER_S                        ; Wait for mailbox AST or time out
                       06    B1  0DBC  1999          CMPW    #MSG_ISSET,-            ; Message acknowledged?
                10B3'CF          0DBE  2000                  SYNCH_MBBUF+MB_W_WHY
                          13    13  0DC1  2001          BEQL    70$                     ; BR if it was
              00000060 8F    C1  0DC3  2002          ADDL3   #COMEF_EFN,-            ; Timed out. Event already logged,...
           50   10B5'CF          0DC9  2003                  SYNCH_MBBUF+MB_K_MISC,R0 ; ...
                                 0DCD  2004          $SETEF_S EFN = R0               ; ...so just recover
                                 0DD6  2005 70$:
                                 0DD6  2006          $READEF_S EFN = #COMEF_EFN,-        ; Check to see...
                                 0DD6  2007                    STATE = SYNCH_MBBUF+MB_K_MISC+4
                10B5'CF    9C    0DE7  2008          ROTL    SYNCH_MBBUF+MB_K_MISC,-     ; ...if the bit we wanted...
                5B    01         0DEB  2009                  #1,R1T
           5B   10B9'CF    D1    0DED  2010          CMPL    SYNCH_MBBUF+MB_K_MISC+4,R1T ; ...got set
                          19    13  0DF2  2011          BEQL    80$                     ; BR if it did
        00 01C0 C6    5A   E2  0DF4  2012          BBSS    R10,MA_L_DOSETFAIL(R6),75$ ; Bit wasn't set, log the error
                                 0DFA  2013 75$:
              00000060 8F    C1  0DFA  2014          ADDL3   #COMEF_EFN,-            ; Event is now logged,...
           50   10B5'CF          0E00  2015                  SYNCH_MBBUF+MB_K_MISC,R0 ; ...
                                 0E04  2016          $SETEF_S EFN = R0               ; ...so just recover
                                 0E0D  2017 80$:
```

```
                                0E0D  2019 ;
                                0E0D  2020 ; We're going through the available processors one-by-one to set random bits
                                0E0D  2021 ; in the common event flag.  Now use the flag bit in a function to determine
                                0E0D  2022 ; a random processor to clear that bit.
                                0E0D  2023 ;
                  10B5'CF   9C  0E0D  2024          ROTL    SYNCH_MBBUF+MB_K_MISC,-         ; This nonsense...
                     5B   24 A7 0E11  2025                  GS_L_AVAILABLE(R7),R11
              5B    5B   20   00 EA 0E14  2026          FFS     #0,#32,R11,R11                  ; ...allows us...
                             08 12 0E19  2027          BNEQ    90$                             ; (Catch pathological case -)
                  10B5'CF   C1  0E1B  2028          ADDL3   SYNCH_MBBUF+MB_K_MISC,-         ; ( - prevent hassles )
                  5B   01AC C6  0E1F  2029                  MA_B_MYPORT(R6),R11             ; ( if no one can send mail!)
                                0E23  2030 90$:
                  5B   10B5'CF   C2 0E23  2031          SUBL2   SYNCH_MBBUF+MB_K_MISC,R11       ; ...to pick...
                               03 18 0E28  2032          BGEQ    100$                            ; ...a random processor...
                  5B   20   C0  0E2A  2033          ADDL2   #32,R11                         ; ...to...
                                0E2D  2034 100$:
                  10B1'CF   5B  90 0E2D  2035          MOVB    R11,SYNCH_MBBUF+MB_B_RECEIVER ; ...
                               07 B0 0E32  2036          MOVW    #MSG_DOCLEAR,-                  ; ...clear the bit just set
                  10B3'CF         0E34  2037                  SYNCH_MBBUF+MB_W_WHY
                  50   5B   D0  0E37  2038          MOVL    R11,R0                  ; Include the port number...
                     03F4   30  0E3A  2039          BSBW    HEXUNIT                 ; ...
                  54   10DB'CF   3C  0E3D  2040          MOVZWL  SYNCH_MBPTR,R4          ; ...
                  10E2'C4   50  90 0E42  2041          MOVB    R0,SYNCH_MBNAM-1(R4)    ; ...in the mailbox name
                                0E47  2042          $CREMBX_S CHAN = SYNCH_MBCHN,-  ; Establish a link to other proc's MB
                                0E47  2043                  LOGNAM = SYNCH_MBPTR
        1102'CF    FA0A1F00 BF  D0 0E5E  2044          MOVL    #DOCLEAR_TIMEOUT,SYNCH_MBDAY ; $SETIMR parameters - DAYTIM...
                  1106'CF   00  D2 0E67  2045          MCOML   #0,SYNCH_MBDAY+4        ; ...
              54   10   10   07 F0 0E6C  2046          INSV    #MSG_DOCLEAR,#16,#16,R4 ; ...REQIDT...
              54   08   08   5B F0 0E71  2047          INSV    R11,#8,#8,R4            ; ...
                   54   01EC C6  90 0E76  2048          MOVB    MA_W_UNIT(R6),R4        ; ...
                                0E7B  2049          $SETIMR_S DAYTIM = SYNCH_MBDAY,-    ; A timer will let us know...
                                0E7B  2050                    ASTADR = MB_TIMEOUT_AST,- ; ...if the other processor...
                                0E7B  2051                    REQIDT = R4               ; ...doesn't reply to the following
           00 01E8 C6   5B   E2 0E8E  2052          BBSS    R11,MA_L_DOCLEAR(R6),110$ ; Log that we expect reply ·  our mail
                                0E94  2053 110$:
                                0E94  2054          $QIO_S  CHAN    = SYNCH_MBCHN,-  ; Tell another processor to clear a bit
                                0E94  2055                  FUNC    = #IO$_WRITEVBLK!IO$M_NOW,-
                                0E94  2056                  IOSB    = SYNCH_MBISB,-
                                0E94  2057                  P1      = SYNCH_MBBUF,-
                                0E94  2058                  P2      = #MB_K_END
                         54   DD 0EB7  2059          PUSHL   R4
                1422'CF    01   FB 0EB9  2060          CALLS   #1,@IOMB_SYNCH_CHECK    ; Check the completion status
                                0EBE  2061          $DASSGN_S CHAN = SYNCH_MBCHN    ; We no longer need the MB
```

```
                     0ECA  2063 ;
                     0ECA  2064 ; Again, for the same reasons as above, we must wait.  Likewise, the mailbox
                     0ECA  2065 ; AST routine and the mailbox timeout routine will act on our buffer as they
                     0ECA  2066 ; did before.
                     0ECA  2067 ;
                     0ECA  2068          $HIBER_S                            ; Wait for mailbox AST or time out
            08   B1  0ED1  2069          CMPW    #MSG_ISCLEAR,-              ; Message acknowledged?
    10B3'CF          0ED3  2070                  SYNCH_MBBUF+MB_W_WHY
            13   13  0ED6  2071          BEQL    120$                        ; BR if it was
  00000060 8F   C1  0ED8  2072          ADDL3   #COMEF_EFN,-                ; Timed out. Event already logged,...
 50  10B5'CF          0EDE  2073                  SYNCH_MBBUF+MB_K_MISC,R0    ; ...
                     0EE2  2074          $CLREF_S EFN = R0                   ; ...so just recover
                     0EEB  2075 120$:
                     0EEB  2076          $READEF_S EFN = #COMEF_EFN,-        ; Check to see if the bit we wanted...
                     0EEB  2077                  STATE = SYNCH_MBBUF+MB_K_MISC+4
    10B9'CF   D5  0EFC  2078          TSTL    SYNCH_MBBUF+MB_K_MISC+4     ; ...got cleared
            19   13  0F00  2079          BEQL    130$                        ; BR if it did
00 01C4 C6   5B   E2  0F02  2080          BBSS    R11,MA_L_DOCLEARFAIL(R6),125$ ; Bit wasn't cleared, log it
                     0F08  2081 125$:
  00000060 8F   C1  0F08  2082          ADDL3   #COMEF_EFN,-                ; Event is now logged,...
 50  10B5'CF          0F0E  2083                  SYNCH_MBBUF+MB_K_MISC,R0    ; ...
                     0F12  2084          $CLREF_S EFN = R0                   ; ...so just recover
                     0F1B  2085 130$:
 5B   5A   58   C3  0F1B  2086          SUBL3   R8,R10,R11                  ; Get number of bits we searched
            5B   D6  0F1F  2087          INCL    R11                         ; Correct off-by-one (inclusive search)
        59   5B   C2  0F21  2088          SUBL2   R11,R9                      ; Adjust number of bits left to search
 58   5A   01   C1  0F24  2089          ADDL3   #1,R10,R8                   ; Point to new first bit to search
           FDB2   31  0F28  2090          BRW     40$                         ; Loop thru all procs on this memory
                     0F28  2091 140$:
                     0F28  2092          $DACEFC_S EFN = #COMEF_EFN          ; Disassociate from shared CEF cluster
        28 A7   D4  0F38  2093          CLRL    GS_L_CEF_FLAG(R7)           ; Release the interlock
                     0F3B  2094 150$:
      56   66   C0  0F3B  2095          ADDL2   (R6),R6                     ; Point to node for next memory
56  000001D8'8F   D1  0F3E  2096          CMPL    #UNIT_LIST,R6               ; Back at the start of the queue?
            03   13  0F45  2097          BEQL    160$                        ; BR if we are - we're finished
          FD64   31  0F47  2098          BRW     10$                         ; More to go so loop
                     0F4A  2099 160$:
```

```
                                0F4A  2101 MEM_INTERLOCK:
                                0F4A  2102 ;+
                                0F4A  2103 ;       Start the part of the test which tests use of interlocking instructions
                                0F4A  2104 ;       in the shared memory.  There is an interlock (please forgive the
                                0F4A  2105 ;       overloading of the word "interlock") longword in the interprocessor
                                0F4A  2106 ;       global section for this part of the test.  If we can't get that lock
                                0F4A  2107 ;       for a given memory, try to get it for the next memory or proceed to the
                                0F4A  2108 ;       next part of the test.  If we do get it, send mail to each processor
                                0F4A  2109 ;       connected to the memory instructing the processor to drop everything
                                0F4A  2110 ;       else that it is doing and begin the interlocking test.
                                0F4A  2111 ;-
                                0F4A  2112
                                0F4A  2113 ; NOTE WELL!  Within the following loop, registers R6 through R10 are set and
                                0F4A  2114 ; assumed to remain unchanged.
 56  01D8'CF   000001D8'8F   C1  0F4A  2115         ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; R6 points to node for current memory
                                0F54  2116 10$:
              57   062C C6   D0  0F54  2117         MOVL    MA_Q_COMGSRAD(R6),R7    ; Point to the start of interproc GS
           08 34 A7    00   E6  0F59  2118         BBSSI   #0,GS_L_INTLK_FLAG(R7),20$ ; Try to grab interlock
                 01AC C6   90  0F5E  2119         MOVB    MA_B_MYPORT(R6),-       ; Bit was clear...
                    37 A7      0F62  2120                 GS_L_INTLK_FLAG+3(R7)
                       03   11  0F64  2121         BRB     30$                     ; ...my port has the interlock
                                0F66  2122 20$:
                     017C   31  0F66  2123         BRW     210$                    ; Bit was set, someone else has it
                                0F69  2124 30$:
                    2C A7   7C  0F69  2125         CLRQ    GS_L_PARTICIPATING(R7)  ; Clear participating and finished masks
              00    67    00   2C  0F6C  2126         MOVC5   #0,(R7),#0,-            ; Clear the common ADAWI words...
                       14       0F70  2127                 #<PROCESSOR_MAX+1>*4,-  ; ...and the private ADAWI words
                    38 A7       0F71  2128                 GS_L_ADAWI_COM(R7)
                    60 A7   7C  0F73  2129         CLRQ    GS_Q_BUSY(R7)           ; Set up the busy queue...
                    58 A7   7C  0F76  2130         CLRQ    GS_Q_FREE(R7)           ; ...clear out the free queue...
        5B    00000064 8F   D0  0F79  2131         MOVL    #COMGS_QUEUE_MAX,R11    ; ...and then set it up...
              55    68 A7   DE  0F80  2132         MOVAL   GS_K_Q_ENTRIES(R7),R5   ; ...
                                0F84  2133 40$:
              58 A7    65   5C  0F84  2134         INSQHI  (R5),GS_Q_FREE(R7)      ; Put an entry on the free queue
                 55    10   C0  0F88  2135         ADDL2   #16,R5                  ; Skip flink, blink, port, filler
                    F6 5B   F5  0F8B  2136         SOBGTR  R11,40$
        08    00    67    00   2C  0F8E  2137         MOVC5   #0,(R7),#0,#PROCESSOR_MAX*2,- ; Clear out the counters...
                    4C A7       0F93  2138                 GS_K_Q_COUNT(R7)        ; ...for each processor's entries
                                0F95  2139
                       58   D4  0F95  2140         CLRL    R8                      ; R8 is STRTPOS for FFS operations
                    59 04   D0  0F97  2141         MOVL    #PROCESSOR_MAX,R9       ; R9 is SIZE for FFS operations
                                0F9A  2142 50$:
 5A   24 A7   59    58   EA  0F9A  2143         FFS     R8,R9,GS_L_AVAILABLE(R7),R10 ; Search for a willing processor
                       03   12  0FA0  2144         BNEQ    60$                     ; BR if we found one
                     00AA   31  0FA2  2145         BRW     100$                    ; BR if there are none left
                                0FA5  2146 60$:
```

```
                     0FA5  2148 ;
                     0FA5  2149 ; If, as in the other sections, the current processor were treated just like
                     0FA5  2150 ; any other, it, too, would get mail, drop everything, and proceed to the
                     0FA5  2151 ; code which did the actual testing.  That would cause a problem unless the
                     0FA5  2152 ; current processor were the last one to receive the mail, an unlikely
                     0FA5  2153 ; circumstance.  Make the current processor a special case, then, and don't
                     0FA5  2154 ; send it mail.  Instead, after all the other mail is sent, simulate the
                     0FA5  2155 ; necessary portions of the other processors' code when the mail is received
                     0FA5  2156 ; and go to the code which does the actual testing.
                     0FA5  2157 ;
        5A   01AC C6   91  0FA5  2158          CMPB    MA_B_MYPORT(R6),R10     ; Did I pick my processor?
                 03    12  0FAA  2159          BNEQ    70$
                 0090  31  0FAC  2160          BRW     90$                     ; BR if so - I don't get my own mail
                     0FAF  2161 70$:
             01AC C6   90  0FAF  2162          MOVB    MA_B_MYPORT(R6),-       ; Start filling in message...
             10B0'CF        0FB3  2163                  SYNCH_MBBUF+MB_B_SENDER ; ...sender's port number...
        10B1'CF   5A   90  0FB6  2164          MOVB    R10,SYNCH_MBBUF+MB_B_RECEIVER ; ...receiver's port number...
             01EC C6   90  0FBB  2165          MOVB    MA_W_UNIT(R6),-         ; ...unit no. of shared memory...
             10B2'CF        0FBF  2166                  SYNCH_MBBUF+MB_B_UNIT
                 09    B0  0FC2  2167          MOVW    #MSG_INTERLOCK,-        ; ...reason for message
             10B3'CF        0FC4  2168                  SYNCH_MBBUF+MB_W_WHY
             0258 C6   28  0FC7  2169          MOVC3   MA_Q_COMMBNAM(R6),-     ; Form the logical name...
    10E3'CF  0260 C6        0FCB  2170                  MA_T_COMMBNAM(R6),SYNCH_MBNAM ; ...of other processor's MB
    10DB'CF  0258 C6   D0  0FD1  2171          MOVL    MA_Q_COMMBNAM(R6),SYNCH_MBPTR ; Form descriptor to...
    10DF'CF  10E3'CF   DE  0FD8  2172          MOVAL   SYNCH_MBNAM,SYNCH_MBPTR+4 ; ...the logical name
             50    5A   D0  0FDF  2173          MOVL    R10,R0                  ; Include the port number...
                 024C  30  0FE2  2174          BSBW    HEXUNIT                 ; ...
            FF A3   50   90  0FE5  2175          MOVB    R0,-1(R3)               ; ...in the name
                     0FE9  2176          $CREMBX_S CHAN = SYNCH_MBCHN,-  ; Establish a link to other proc's MB
                     0FE9  2177                    LOGNAM = SYNCH_MBPTR
                     1000  2178 ; Note that no timer is set - we assume any processor which will start will do
                     1000  2179 ; so as soon as possible.
                     1000  2180          $QIO_S  CHAN    = SYNCH_MBCHN,-  ; Tell the other processor to...
                     1000  2181                  FUNC    = #IO$_WRITEVBLK!IO$M_NOW,-
                     1000  2182                  IOSB    = SYNCH_MBISB,-  ; ...test memory interlocking
                     1000  2183                  P1      = SYNCH_MBBUF,-
                     1000  2184                  P2      = #MB_K_END
         7E   09   B0  1023  2185          MOVW    #MSG_INTERLOCK,-(SP)    ; Set up QIOMB_SYNCH_CHECK param
         7E   5A   90  1026  2186          MOVB    R10,-(SP)
    7E   01EC C6   90  1029  2187          MOVB    MA_W_UNIT(R6),-(SP)
    1422'CF   01   FB  102E  2188          CALLS   #1,QIOMB_SYNCH_CHECK    ; Check the completion status
                     1033  2189          $DASSGN_S CHAN = SYNCH_MBCHN    ; We no longer need the MB
                     103F  2190 90$:
    5B   5A   58   C3  103F  2191          SUBL3   R8,R10,R11              ; Get number of bits we searched
              5B   D6  1043  2192          INCL    R11                     ; Correct off-by-one (inclusive search)
              5B   C2  1045  2193          SUBL2   R11,R9                  ; Adjust number of bits left to search
    5R        01   C1  1048  2194          ADDL3   #1,R10,R8               ; Point to new first bit to search
            FF4B   31  104C  2195          BRW     50$                     ; Loop thru all procs on this memory
                     104F  2196 100$:
    1164'CF   56   D0  104F  2197          MOVL    R6,INTERLOCK_SAVED_R6   ; Save vital register...
                     1054  2198                                          ; ...and lock us to this memory
    10F7'CF   00   FB  1054  2199          CALLS   #0,INTERLOCKING_TEST    ; Go thrash memory interlock feature
         1164'CF   D4  1059  2200          CLRL    INTERLOCK_SAVED_R6      ; It's OK to thrash some other memory
```

```
                                105D  2202 ;
                                105D  2203 ; We've finished our portion of the test; it's a fair guess to say that other
                                105D  2204 ; processors are probably close to finishing also.  However, to be on the safe
                                105D  2205 ; side, allow some extra time for them to finish.  We'll know they're finished
                                105D  2206 ; because the mask of processors which have finished will be the same as the
                                105D  2207 ; mask as those which started.
                                105D  2208 ;
                          5B    D4  105D  2209          CLRL    R11                             ; Clear out our counter
                                105F  2210 120$:
                        2C A7   CD  105F  2211          XORL3   GS_L_PARTICIPATING(R7),-        ; Has everyone finished yet?
                52      30 A7       1062  2212                  GS_L_FINISHED(R7),R2
                             24 13  1065  2213          BEQL    130$                            ; BR if they have
                                1067  2214          $SCHDWK_S DAYTIM = ONE_SECOND           ; Wait a second...
                                1078  2215          $HIBER_S                                ; ...if they haven't
              DC 5B    0A    F2  107F  2216          AOBLSS  #INTERLOCK_LIMIT,R11,120$       ; Loop a reasonable number of times
            01D0 C6    52    C8  1083  2217          BISL2   R2,MA_L_INTLKTMO(R6)            ; Timed out, remember who is left
                     0057    31  1088  2218          BRW     200$                            ; Consistency checking is useless
                                108B  2219 130$:
                                108B  2220
                                108B  2221 ;
                                108B  2222 ; To see if the memory interlocking worked correctly, we check the results of
                                108B  2223 ; the two tests performed.  For the ADAWI test, the common longword should
                                108B  2224 ; equal the sum of the private longwords.  For the queue test, the busy queue
                                108B  2225 ; should be empty, all entries should be on the free queue and no processor
                                108B  2226 ; should have any queue entries outstanding.
                                108B  2227 ;
                          5B    D4  108B  2228          CLRL    R11                             ; Initialize sum
                          55    D4  108D  2229          CLRL    R5                              ; Initialize counter and pointer
                                108F  2230 140$:
                    5B    3C A745   C0  108F  2231          ADDL2   GS_K_ADAWI_PVT(R7)[R5],R11 ; Add in this processor's count
                      F7 55    04   F2  1094  2232          AOBLSS  #PROCESSOR_MAX,R5,140$  ; Loop through all count longwords
                      5B    38 A7   D1  1098  2233          CMPL    GS_L_ADAWI_COM(R7),R11  ; Expected sum equal actual sum?
                                06  13  109C  2234          BEQL    150$                    ; BR if it is
                                30 A7   C8  109E  2235          BISL2   GS_L_FINISHED(R7),-     ; Otherwise, remember the group...
                             01D4 C6        10A1  2236                  MA_L_IADAWI(R6)         ; ...of CPUs who didn't add up
                                10A4  2237 150$:
                                10A4  2238
                      50    60 A7   7D  10A4  2239          MOVQ    GS_Q_BUSY(R7),R0        ; Is busy queue empty?  (R0 is trash)
                                32  12  10A8  2240          BNEQ    190$                    ; BR if it isn't
                                55  D4  10AA  2241          CLRL    R5                      ; Initialize counter and pointer
                                10AC  2242 160$:
                             4C A745   B5  10AC  2243          TSTW    GS_K_Q_COUNT(R7)[R5]    ; Any entries for this processor?
                                2A  12  10B0  2244          BNEQ    190$                    ; BR if there are - we're screwed up
                      F6 55    04   F2  10B2  2245          AOBLSS  #PROCESSOR_MAX,R5,160$  ; Loop to check all processors
                                10B6  2246
                          58 A7   D5  10B6  2247          TSTL    GS_Q_FREE(R7)           ; Anything on free queue?
                                21  13  10B9  2248          BEQL    190$                    ; BR if not - all screwed up
                      5B    58 A7   DE  10BB  2249          MOVAL   GS_Q_FREE(R7),R11       ; Point to header of free queue
                            54    5B  D0  10BF  2250          MOVL    R11,R4                  ; Set up a constant to compare with
                55  00000064    8F  D0  10C2  2251          MOVL    #COMGS_QUEUE_MAX,R5     ; Initialize counter
                                10C9  2252 170$:
                          5B    6B  C0  10C9  2253          ADDL2   (R11),R11               ; Point to the next queue entry
                                55  D7  10CC  2254          DECL    R5                      ; Count down entries in queue
                                07  19  10CE  2255          BLSS    180$                    ; BR if we've examined all entries
                          5B    54  D1  10D0  2256          CMPL    R4,R11                  ; Pointing back at the header now?
                                07  13  10D3  2257          BEQL    190$                    ; Shouldn't be!  BR if we are
                                F2  11  10D5  2258          BRB     170$                    ; See where the next entry points us
```

```
                        10D7  2259 180$:
          5B    54      D1  10D7  2260          CMPL    R4,R11                  ; All entries counted, back at header?
                06      13  10DA  2261          BEQL    200$                    ; We're OK - go clean up
                        10DC  2262 190$:
            30  A7      C8  10DC  2263          BISL2   GS_L_FINISHED(R7),-     ; Inconsistent queues or count,...
          01D8  C6          10DF  2264                  MA_L_IQUEUE(R6)         ; ...remember who participated
                        10E2  2265 200$:
            34  A7      D4  10E2  2266          CLRL    GS_L_INTLK_FLAG(R7)     ; Release the interlock
                        10E5  2267 210$:
          56    66      C0  10E5  2268          ADDL2   (R6),R6                 ; Point to node for next memory
    56  000001D8'8F     D1  10E8  2269          CMPL    #UNIT_LIST,R6           ; Back at the start of the queue?
                03      13  10EF  2270          BEQL    220$                    ; BR if we are - we're finished
              FE60      31  10F1  2271          BRW     10$                     ; More to go so loop
                        10F4  2272 220$:
              008C      31  10F4  2273          BRW     NEXT_ITERATION          ; Pass over INTERLOCKING_TEST
```

```
                                10F7  2275 INTERLOCKING_TEST:
                                10F7  2276 ;+
                                10F7  2277 ;       We reach this point by two routes.  The simple one is a call from the
                                10F7  2278 ;       code at MEM_INTERLOCK.  In that case, this processor owns the lock for
                                10F7  2279 ;       a memory and executes the test synchronously.  Other processors have
                                10F7  2280 ;       been sent mailboxes and will join in the testing as soon as they can.
                                10F7  2281 ;       If, on the other hand, we are one of the processors which was off in
                                10F7  2282 ;       some other part of the test and suddenly got a mailbox AST, this code
                                10F7  2283 ;       is being executed asynchronously, BUT NOT AT AST LEVEL!  The mailbox
                                10F7  2284 ;       routine has reenabled ASTs.  In either case, we have the address of
                                10F7  2285 ;       the node for the memory in which the interlocking will be tested in
                                10F7  2286 ;       INETRLOCK_SAVED_R6, which also serves to lock us to thrashing one
                                10F7  2287 ;       memory at a time.
                                10F7  2288 ;-
                                10F7  2289
                        0FFC    10F7  2290         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
 56     1164'CF       D0      10F9  2291         MOVL    INTERLOCK_SAVED_R6,R6   ; Set up vital registers - node...
 57     062C C6       D0      10FE  2292         MOVL    MA_Q_COMGSRAD(R6),R7    ; ...and base of interprocessor GS
 58     01AC C6       9A      1103  2293         MOVZBL  MA_B_MYPORT(R6),R8              ; Set up to say that...
00 2C A7    58       E6      1108  2294         BBSSI   R8,GS_L_PARTICIPATING(R7),10$ ; ...we're participating
                                110D  2295 10$:
                                110D  2296
                                110D  2297 ;
                                110D  2298 ; For the first test of memory interlocking, we will try to force collisions
                                110D  2299 ; using the add-aligned-word-interlocked instruction.  Each processor
                                110D  2300 ; increments in parallel a longword reserved to that processor and a longword
                                110D  2301 ; accessed in common.  The sum of the private longwords should equal the value
                                110D  2302 ; in the common longword.
                                110D  2303 ;
 58     3C A748       DE      110D  2304         MOVAL   GS_K_ADAWI_PVT(R7)[R8],R8 ; Set up pointer to my own counter...
                        59    D4    1112  2305         CLRL    R9                      ; ...and a loop counter
                              1114  2306 20$:
   38 A7       01       58    1114  2307         ADAWI   #1,GS_L_ADAWI_COM(R7)   ; Increment the common word
                        04    1E    1118  2308         BCC     30$             ; BR if no carry
   3A A7       01       58    111A  2309         ADAWI   #1,GS_L_ADAWI_COM+2(R7) ; Increment common overflow word
                              111E  2310 30$:
      68       01       58    111E  2311         ADAWI   #1,(R8)                 ; Increment my private word
                        04    1E    1121  2312         BCC     40$             ; BR if no carry
   02 A8       01       58    1123  2313         ADAWI   #1,2(R8)                ; Increment my private overflow word
                              1127  2314 40$:
E5 59    00030D40    8F       F2    1127  2315         AOBLSS  #ADAWI_LIMIT,R9,20$     ; Loop until counter is exhausted
```

```
                        112F  2317 ;
                        112F  2318 ; For the second test of memory interlocking, processors will take an entry
                        112F  2319 ; from a "free" queue and add it to a "busy" queue.  They will keep count in
                        112F  2320 ; a per processor area of the number of entries they have in the busy queue.
                        112F  2321 ; When the free queue is exhausted, processors will start to remove queue
                        112F  2322 ; entries (anyone's) from the busy queue and replace them in the free queue,
                        112F  2323 ; decrementing the appropriate count.  The sequence repeats a specified number
                        112F  2324 ; of times.  When the dust dies down, each processor's count should be zero,
                        112F  2325 ; the busy queue empty and the free queue full (the same as our initial state).
                        112F  2326 ;
      58  01AC C6   9A  112F  2327         MOVZBL  MA_B_MYPORT(R6),R8      ; Set up a pointer to...
      58  4C A748   3E  1134  2328         MOVAW   GS_K_Q_COUNT(R7)[R8],R8 ; ...my counter
            59      D4  1139  2329         CLRL    R9                      ; Clear loop counter
                        113B  2330 110$:
        5A  58 A7   5E  113B  2331         REMQHI  GS_Q_FREE(R7),R10       ; Pick off a "free" queue entry
              04    1C  113F  2332         BVC     120$                    ; BR if we got one from the queue
              F8    1F  1141  2333         BCS     110$                    ; BR if secondary interlock failed
              11    11  1143  2334         BRB     140$                    ; Queue is empty, start refilling it
                        1145  2335 120$:
08 AA   01AC C6   9A  1145  2336         MOVZBL  MA_B_MYPORT(R6),8(R10)  ; Mark the entry as mine
                        114B  2337 130$:
        60 A7   6A  5C  114B  2338         INSQHI  (R10),GS_Q_BUSY(R7)     ; Put the entry in the "busy" queue
              FA    1F  114F  2339         BCS     130$                    ; BR if secondary interlock failed
           68   01  58  1151  2340         ADAWI   #1,(R8)                 ; Count my entries in the busy queue
              E5    11  1154  2341         BRB     110$                    ; Loop until free queue is exhausted
                        1156  2342 140$:
        5A  60 A7   5F  1156  2343         REMQTI  GS_Q_BUSY(R7),R10       ; Pick off a "busy" queue entry
              04    1C  115A  2344         BVC     150$                    ; BR if we got one from the queue
              F8    1F  115C  2345         BCS     140$                    ; BR if secondary interlock failed
              13    11  115E  2346         BRB     170$                    ; Queue is empty, count an iteration
                        1160  2347 150$:
        5B  08 AA   D0  1160  2348         MOVL    8(R10),R11              ; Who owned this entry?
4C A74B  FFFF 8F    58  1164  2349         ADAWI   #-1,GS_K_Q_COUNT(R7)[R11] ; Decrement the owner's count
                        116B  2350 160$:
        58 A7   6A  5D  116B  2351         INSQTI  (R10),GS_Q_FREE(R7)     ; Replace the entry in the free queue
              FA    1F  116F  2352         BCS     160$                    ; BR if secondary interlock failed
              E3    11  1171  2353         BRB     140$                    ; Loop until busy queue is empty
                        1173  2354 170$:
C0 59   00000GC8 8F   F2  1173  2355         AOBLSS  #QUEUE_LIMIT,R9,110$    ; Loop until counter is exhausted
                        117B  2356
                        117B  2357 ;
                        117B  2358 ; This processor has finished its share of the memory interlock test.  However
                        117B  2359 ; the success or failure of the test can be checked only by the initiating
                        117B  2360 ; processor and then only when all processors which indicated that they were
                        117B  2361 ; participating have indicated that they finished.  So for now, just indicate
                        117B  2362 ; that we're done.
                        117B  2363 ;
         01AC C6    E6  117B  2364         BBSSI   MA_B_MYPORT(R6),-       ; Say that we've finished
      00 30 A7           117F  2365                 GS_L_FINISHED(R7),200$
                        1182  2366 200$:
              04    1182  2367         RET
```

```
                        1183  2369 NEXT_ITERATION:
          01B2'CF   D6  1183  2370         INCL    ITERATION                ; Increment iteration count
03 0002'CF    01  E0  1187  2371         BBS     #TEST_OVERV,FLAG,SUC_EXIT ; BR if test is over
        F6FA    31  118D  2372         BRW     ONEPROC_GS               ; Loop until the test is over
                        1190  2373
                        1190  2374 SUC_EXIT:
   1C47'CF    00  FB  1190  2375         CALLS   #0,ERROR_SUMMARY         ; Print any error statistics
                        1195  2376         $TRNLOG_S LOGNAM = MODE,-
                        1195  2377                   RSLLEN = BUFFER_PTR,-
                        1195  2378                   RSLBUF = FAO_BUF       ; Get the run mode
   0014'CF    20  8A  11AE  2379         BICB2   #LC_BITM,BUFFER          ; Convert to upper case
0014'CF    4C  8F  91  11B3  2380         CMPB    #^A7L/,BUFFER            ; Is this a loop for ever?
        53  12  11B9  2381         BNEQ    10$                      ; BR if not
   0002'CF    02  AA  11BB  2382         BICW2   #TEST_OVERM,FLAG         ; Reset the termination flag
          01B6'CF   D6  11C0  2383         INCL    PASS                     ; Bump the pass count
                        11C4  2384         $FAO_S  CTRSTR = PASS_MSG,-
                        11C4  2385                 OUTLEN = BUFFER_PTR,-
                        11C4  2386                 OUTBUF = FAO_BUF,-
                        11C4  2387                 P1     = PASS,-
                        11C4  2388                 P2     = ITERATION,-
                        11C4  2389                 P3     = #0              ; Make the end of pass message
          000C'CF   DF  11E1  2390         PUSHAL  BUFFER_PTR               ; Push the string desc.
                01  DD  11E5  2391         PUSHL   #1                       ; Push arg count
     00741133 8F  DD  11E7  2392         PUSHL   #UETP$_TEXT!STS$K_INFO   ; Push the signal name
00000000'GF    03  FB  11ED  2393         CALLS   #3,G^LIB$SIGNAL          ; Print the end of pass message
          01B2'CF   D4  11F4  2394         CLRL    ITERATION                ; Reset the iteration count
                        11F8  2395         $SETIMR_S DAYTIM = THREEMIN,-    ; Set a new timer AST for 3 minutes
                        11F8  2396                   ASTADR = TIME_OUT,-
                        11F8  2397                   EFN    = #EFN2
              F67C    31  120B  2398         BRW     ONEPROC_GS               ; Start next pass, avoiding re-init
                        120E  2399 10$:
09 0002'CF    04  E1  120E  2400         BBC     #ONE_SHOTV,FLAG,20$      ; Skip following unless one-shot mode
                        1214  2401         $WAITFR_S EFN = #SPRGS_EFN       ; Allow $UPDSEC to finish - otherwise...
                        121D  2402                                          ; ...we exit, turn off ASTs and never...
                        121D  2403                                          ; ...check data validity
                        121D  2404 20$:
     10000001 8F    D0  121D  2405         MOVL    #SS$_NORMAL!STS$M_INHIB_MSG,- ; Set successful exit status
          018A'CF         1223  2406                 STATUS
                        1226  2407         $EXIT_S STATUS                   ; Exit with the status
```

```
                     1231  2409          .SBTTL  Quickie Subroutines
                     1231  2410 ;++
                     1231  2411 ; FUNCTIONAL DESCRIPTION:
                     1231  2412 ;       ...being a set of short subroutines not worthy of much comment.
                     1231  2413 ;
                     1231  2414 ; TYPICAL CALLING SEQUENCE:
                     1231  2415 ;       MOVx    arg,R0
                     1231  2416 ;       BSBW    subroutine
                     1231  2417 ;       result returned in R0
                     1231  2418 ;
                     1231  2419 ; INPUT PARAMETERS:
                     1231  2420 ;       defined for each routine
                     1231  2421 ;
                     1231  2422 ; OUTPUT PARAMETERS:
                     1231  2423 ;       defined for each routine
                     1231  2424 ;
                     1231  2425 ;--
                     1231  2426
                     1231  2427 ;
                     1231  2428 ; Given the unit number of a multiport memory (one byte), return its
                     1231  2429 ; ASCII equivalent.  Routine is very trusting.
                     1231  2430 ;
                     1231  2431 HEXUNIT:
50   09   91        1231  2432          CMPB    #9,R0                   ; Is unit within decimal digits?
     03   18        1234  2433          BGEQ    10$                     ; BR if it is
50   37   80        1236  2434          ADDB2   #<^A/A/-10>,R0          ; Add a kludge factor if it isn't
                     1239  2435 10$:
50   30   80        1239  2436          ADDB2   #^A/0/,R0               ; Convert hex to ASCII
          05        123C  2437          RSB
                     123D  2438
                     123D  2439
                     123D  2440 ;
                     123D  2441 ; Given the address of a buffer of longwords in R1 and the buffer size in
                     123D  2442 ; longwords in R0, fill the buffer with pseudo-random data.
                     123D  2443 ;
                     123D  2444 RANDOM_FILL:
01AA'CF   01AE'CF    C0    123D  2445          ADDL2   RANDOM2,RANDOM1         ; Create a "random" longword
     81   01AA'CF    D0    1244  2446          MOVL    RANDOM1,(R1)+           ; Stuff it in our buffer...
            F1 50    F5    1249  2447          SOBGTR  R0,RANDOM_FILL          ; ...until the buffer is full
                     05    124C  2448          RSB
```

```
                                    124D  2450 ;
                                    124D  2451 ; Take a bit mask in R0 and from it produce a list on the stack of those bit
                                    124D  2452 ; positions in R0 which have a 1 in them.  The list is suitable for use with
                                    124D  2453 ; the !#() $FAO directive.  R0 returns the number of items in the list and R1
                                    124D  2454 ; is trashed.
                                    124D  2455 ;
                                    124D  2456 MASK_?_ARGLST:
                              02    BA    124D  2457          POPR    #^M<R1>                     ; Save our return address
                              00    DD    124F  2458          PUSHL   #0                          ; Set up counter on stack
                                          1251  2459 10$:
                              50    D5    1251  2460          TSTL    R0                          ; Will FFS find any bits?
                              0F    13    1253  2461          BEQL    20$                         ; BR if it won't
                        7E    6E    01    C1    1255  2462          ADDL3   #1,(SP),-(SP)               ; Count one it will find into new counter
                  04 AE   50    20    00    EA    1259  2463          FFS     #0,#32,R0,4(SP)             ; Find a lit bit, save its position
                     ED 50    04 AE    E4    125F  2464          BBSC    4(SP),R0,10$                ; Ensure we find it only once. Look for next
                                          1264  2465 20$:
                              50    6E    D0    1264  2466          MOVL    (SP),R0                     ; Return subroutine value
                              61    17    1267  2467          JMP     (R1)                        ; RSB, a la kludge
                                          1269  2468
                                          1269  2469
                                    1269  2470 ;
                                    1269  2471 ; Given the address of an $FAO control string in R0, do a "typical" call to the
                                    1269  2472 ; $FAO system service - a call which includes only the shared memory name and
                                    1269  2473 ; its unit number.  R1 is trashed.  R0 has result of $FAO.  Assumes that we are
                                    1269  2474 ; "typical" in that R6 contains the address of the unit block of the current
                                    1269  2475 ; memory.
                                    1269  2476 ;
                                    1269  2477 DO_FAO:
51    01F1 C6    DE                 1269  2478          MOVAL   MA_Q_MEMNAM(R6),R1          ; Set up $FAO parameter
                                    126E  2479          $FAO_S  CTRSTR = (R0),-
                                    126E  2480                  OUTLEN = BUFFER_PTR,-
                                    126E  2481                  OUTBUF = FAO_BUF,-
                                    126E  2482                  P1     = R1,-
                                    126E  2483                  P2     = MA_W_UNIT(R6)
                              05    1285  2484          RSB
```

```
                                1286  2486 ;
                                1286  2487 ; With the same inputs as the DO_FAO subroutine, do the $FAO and follow it
                                1286  2488 ; by a call to LIB$SIGNAL which treats the result of the $FAO as a warning
                                1286  2489 ; or info message.  R0 and R1 are trashed.
                                1286  2490 ;
                                1286  2491 DO_FAO_SIGNAL:
                          00    DD  1286  2492         PUSHL   #STS$K_WARNING
                                1288  2493         .ENABLE LSB
                          02    11  1288  2494         BRB     10$
                                128A  2495 DO_FAO_INFO:
                          03    DD  128A  2496         PUSHL   #STS$K_INFO
                                128C  2497 10$:
                                128C  2498         .DISABLE LSB
                          DB    10  128C  2499         BSBB    DO_FAO                  ; Get the $FAO over with
     51   8E    00741130 8F    C9  128E  2500         BISL3   #UETP$_TEXT,(SP)+,R1    ; Form the correct message code
                   000C'CF    DF  1296  2501         PUSHAL  BUFFER_PTR
                          01   DD  129A  2502         PUSHL   #1
                          51   DD  129C  2503         PUSHL   R1
                00000000'GF    03  FB  129E  2504         CALLS   #3,G^LIB$SIGNAL
                                05  12A5  2505         RSB
                                12A6  2506
                                12A6  2507
                                12A6  2508 ;
                                12A6  2509 ; Similar to DO_FAO_SIGNAL, but instead of doing the call to LIB$SIGNAL, die
                                12A6  2510 ; by way of ERROR_EXIT, which does its own call to LIB$SIGNAL.  No return.
                                12A6  2511 ;
                                12A6  2512 DO_FAO_EXIT:
                          C1    10  12A6  2513         BSBB    DO_FAO                  ; Get the $FAO over with
                   000C'CF    DF  12A8  2514         PUSHAL  BUFFER_PTR
                          01    DD  12AC  2515         PUSHL   #1
                   00741132 8F    DD  12AE  2516         PUSHL   #UETP$_TEXT!STS$K_ERROR
                          03    DD  12B4  2517         PUSHL   #3
                        0DE7    31  12B6  2518         BRW     ERROR_EXIT
```

```
                        12B9  2520          .SBTTL  Shared Memory Name Routine
                        12B9  2521 ;++
                        12B9  2522 ; FUNCTIONAL DESCRIPTION:
                        12B9  2523 ;       Given the unit number of a shared memory, extract from the SHD and ADP
                        12B9  2524 ;       the corresponding memory name and name length and the port to which our
                        12B9  2525 ;       CPU is connected.
                        12B9  2526 ;
                        12B9  2527 ; CALLING SEQUENCE:
                        12B9  2528 ;       $CMKRNL_S ROUTIN = SHM_INFO,-
                        12B9  2529 ;                 ARGLST = argument-list-as-described-below
                        12B9  2530 ;
                        12B9  2531 ; INPUT PARAMETERS:
                        12B9  2532 ;       @04(AP) - Unit number of multiport memory
                        12B9  2533 ;
                        12B9  2534 ; IMPLICIT INPUTS:
                        12B9  2535 ;       NONE
                        12B9  2536 ;
                        12B9  2537 ; OUTPUT PARAMETERS:
                        12B9  2538 ;       @08(AP) - Address of byte for shared memory port
                        12B9  2539 ;       @12(AP) - Address for .ASCII string of shared memory name
                        12B9  2540 ;       @16(AP) - Address of word for length of shared memory name
                        12B9  2541 ;
                        12B9  2542 ; IMPLICIT OUTPUTS:
                        12B9  2543 ;       NONE
                        12B9  2544 ;
                        12B9  2545 ; COMPLETION CODES:
                        12B9  2546 ;       SS$_NORMAL
                        12B9  2547 ;       UETP$_DENOSU if no arglist supplied or can't find specified unit
                        12B9  2548 ;
                        12B9  2549 ; SIDE EFFECTS:
                        12B9  2550 ;       NONE
                        12B9  2551 ;
                        12B9  2552 ;--
                        12B9  2553
                        12B9  2554 SHM_INFO:
                   0FFC 12B9  2555          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        12BB  2556
                     5C D5  12BB  2557          TSTL    AP                      ; Were we supplied an argument list?
                     48 13  12BD  2558          BEQL    30$                     ; BR if not
56   00000000'GF        D0  12BF  2559          MOVL    G^EXE$GL_SHBLIST,R6     ; Point to SHB list
                     3F 13  12C6  2560          BEQL    30$                     ; BR if none - can't find ours
                        12C8  2561 10$:
                  57 1C A6  D0  12C8  2562          MOVL    SHB$L_ADP(R6),R7        ; Get ADP pointer
                     39 13  12CC  2563          BEQL    30$                     ; BR if there is none - we've goofed
                  14 A6 91  12CE  2564          CMPB    SHB$B_NEXUS(R6),-       ; Does ADP match our SHB?
                  0C A7     12D1  2565                  ADP$W_TR(R7)
                     32 12  12D3  2566          BNEQ    30$                     ; BR if not - someone goofed
               57    67 D0  12D5  2567          MOVL    ADP$L_CSR(R7),R7        ; Get CSR of adapter
                     2D 13  12D8  2568          BEQL    30$                     ; BR if something is screwey
                  57 1C A7  D0  12DA  2569          MOVL    MPM$L_MR(R7),R7         ; (I/O space reference restriction)
                     0E EF  12DE  2570          EXTZV   #MPM$V_MR_UNIT,-        ; Get the unit number of our memory
            57    57 02     12E0  2571                  #MPM$S_MR_UNIT,R7,R7
                  57 04 BC  B1  12E3  2572          CMPW    @04(AP),R7              ; Got the right memory unit?
                     19 12  12E7  2573          BNEQ    20$                     ; BR if not
                  57 04 A6  D0  12E9  2574          MOVL    SHB$L_DATAPAGE(R6),R7   ; Got correct memory. Get its SHD
            10 BC 20 A7  9B  12ED  2575          MOVZBW  SHD$T_NAME(R7),@16(AP)  ; Get the length of the memory name...
                  10 BC 28  12F2  2576          MOVC3   @16(AP),-               ; ...and the name itself
```

```
                OC BC    21 A7      12F5  2577                    SHD$T_NAME+1(R7),@12(AP)
                08 BC    15 A6  90  12F9  2578            MOVB    SHB$B_PORT(R6),@08(AP)  ; Return the port connected to our CPU
                      50    01  D0  12FE  2579            MOVL    #SS$_NORMAL,R0          ; We've succeeded
                                04  1301  2580            RET
                                    1302  2581 20$:
                      56    66  D0  1302  2582            MOVL    SHB$L_LINK(R6),R6       ; Check the next SHB
                            C1  12  1305  2583            BNEQ    10$                     ; BR if there is one...
                                    1307  2584                                            ; ...else fall into error return
                                    1307  2585 30$:
          50    00748332 8F     D0  1307  2586            MOVL    #<UETP$_DENOSU&^C7!STS$K_ERROR>,R0 ; Something is awry
                                04  130E  2587            RET
```

```
                     130F  2589          .SBTTL  $UPDSEC AST Routine for Single Processor Section
                     130F  2590 ;++
                     130F  2591 ; FUNCTIONAL DESCRIPTION:
                     130F  2592 ;       This routine gets called at AST level whenever the $UPDSEC finishes
                     130F  2593 ;       for a single processor global section.  It takes care of verifying the
                     130F  2594 ;       data and cleaning up the section.
                     130F  2595 ;
                     130F  2596 ; CALLING SEQUENCE:
                     130F  2597 ;       Called as an AST in user mode
                     130F  2598 ;
                     130F  2599 ; INPUT PARAMETERS:
                     130F  2600 ;       ASTPRM is the address of the node for the memory with the section.
                     130F  2601 ;
                     130F  2602 ; IMPLICIT INPUTS:
                     130F  2603 ;       NONE
                     130F  2604 ;
                     130F  2605 ; OUTPUT PARAMETERS:
                     130F  2606 ;       NONE
                     130F  2607 ;
                     130F  2608 ; IMPLICIT OUTPUTS:
                     130F  2609 ;       Section is verified, virtual addresses deleted, section deleted and
                     130F  2610 ;       channel deassigned.
                     130F  2611 ;
                     130F  2612 ; COMPLETION CODES:
                     130F  2613 ;       NONE
                     130F  2614 ;
                     130F  2615 ; SIDE EFFECTS:
                     130F  2616 ;       NONE
                     130F  2617 ;
                     130F  2618 ;--
                     130F  2619
                     130F  2620 SPRGS_AST:
                0FFC 130F  2621          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     1311  2622
6D   1F17'CF    DE   1311  2623          MOVAL   SSERROR,(FP)                ; Set up AST level exception handler
  56     04 AC  D0   1316  2624          MOVL    4(AP),R6                    ; Fetch pointer to node
  54 0644 C6    E8   131A  2625          BLBS    MA_Q_SPRGSISB(R6),20$       ; BR if the $UPDSEC went well
        57      D4   131F  2626          CLRL    R7                          ; It bombed. Clear err msg arg counter
7E   0644 C6    3C   1321  2627          MOVZWL  MA_Q_SPRGSISB(R6),-(SP)     ; Save the main reason for the error
  0F 0646 C6    E9   1326  2628          BLBC    MA_Q_SPRGSISB+2(R6),10$     ; BR if hardware was not a factor
     04A8'CF    DF   132B  2629          PUSHAL  UPDSEC_HWE                  ; Hardware write error. Let user know
        01      DD   132F  2630          PUSHL   #1
 00741132 8F    DD   1331  2631          PUSHL   #UETP$_TEXT!STS$K_ERROR
     57   03    D0   1337  2632          MOVL    #3,R7                       ; Save count of args for HWE
                     133A  2633 10$:
        02      8A   133A  2634          BICB2   #UETUNT$M_TESTABLE,-        ; The MPM is not testable...
     0B A6              133C  2635                  UETUNT$B_FLAGS(R6)          ; ...(as far as one-shot thinks)
55   01F1 C6    DE   133E  2636          MOVAL   MA_Q_MEMNAM(R6),R5          ; Form $FAO parameter
                     1343  2637          $FAO_S  CTRSTR = UPDSEC_FAILED,-    ; Form msg to say where error occurred
                     1343  2638                  OUTLEN = BUFFER_PTR,-
                     1343  2639                  OUTBUF = FAO_BUF,-
                     1343  2640                  P1     = R5,-
                     1343  2641                  P2     = MA_W_UNIT(R6),-
                     1343  2642                  P3     = MA_Q_SPRGSISB+4(R6)
     000C'CF    DF   1360  2643          PUSHAL  BUFFER_PTR                  ; NOTE: Although we use BUFFER and...
        01      DD   1364  2644          PUSHL   #1                          ; ...BUFFER_PTR here and synchronously...
 00741132 8F    DD   1366  2645          PUSHL   #UETP$_TEXT!STS$K_ERROR     ; ...we should be safe because we exit
```

```
                7E    57    04    C1  136C  2646          ADDL3   #4,R7,-(SP)             ; Get total of args
                          0D2D    31  1370  2647          BRW     ERROR_EXIT
                                      1373  2648 20$:
                                      1373  2649          $DELTVA_S INADR = MA_Q_SPRGSRAD(R6) ; We don't need section any more...
                                      1382  2650          $DGBLSC_S GSDNAM = MA_Q_SPRGSNAM(R6) ; ...
                                      1391  2651          $DASSGN_S CHAN = MA_K_SPRFAB+FAB$L_STV(R6) ;
                                      139D  2652          $OPEN   FAB = MA_K_CHKFAB(R6),- ; Reopen the file for the section
                                      139D  2653                  ERR = RMS_ERROR
                            72 50     E9  13AC  2654          BLBC    R0,40$                  ; Exit quickly if RMS file error
                                      13AF  2655          $CONNECT RAB = MA_K_CHKRAB(R6),-
                                      13AF  2656                   ERR = RMS_ERROR
                                      13BE  2657          $READ   RAB = MA_K_CHKRAB(R6),- ; Read the entire file
                                      13BE  2658                   ERR = RMS_ERROR
                57  264C C6        DE  13CD  2659          MOVAL   MA_K_CHKBUF(R6),R7      ; Save address of check buffer
                    2000 8F         29  13D2  2660          CMPC3   #WRITE_SIZE,-           ; Did the file get updated OK?
                67  064C C6            13D6  2661                  MA_K_SPRBUF(R6),(R7)
                              36    13  13DA  2662          BEQL    30$                     ; BR if it did
                          7E  63    9A  13DC  2663          MOVZBL  (R3),-(SP)              ; Bad update.  Save the bad byte...
                          7E  61    9A  13DF  2664          MOVZBL  (R1),-(SP)              ; ...the corresponding good byte...
                7E    53    57    C3  13E2  2665          SUBL3   R7,R3,-(SP)             ; ...where in the buffer it was...
                          01F1 C6   DF  13E6  2666          PUSHAL  MA_Q_MEMNAM(R6)         ; ...the memory where error was seen...
                                04  DD  13EA  2667          PUSHL   #4                      ; ...the argument count...
                         0074801A 8F  DD  13EC  2668          PUSHL   #UETP$_DATADEVERR!STS$K_ERROR ; ...and the error type
                              0457'CF  DF  13F2  2669          PUSHAL  BAD_UPDATE              ; Tell what the error was from
                                01  DD  13F6  2670          PUSHL   #1
                         00741132 8F  DD  13F8  2671          PUSHL   #UETP$_TEXT!STS$K_ERROR
                                09  DD  13FE  2672          PUSHL   #9                      ; Count the args we pushed
                                      1400  2673          $CLOSE  FAB = MA_K_CHKFAB(R6)   ; Allow us to delete file, later
                                      140B  2674                                          ; (ERR= is useless, we're already dead)
                                02  8A  140B  2675          BICB2   #UETUNT$M_TESTABLE,-    ; The MPM is not testable...
                              0B A6         140D  2676                  UETUNT$B_FLAGS(R6)  ; ...(as far as one-shot thinks)
                               0C8E   31    140F  2677          BRW     ERROR_EXIT
                                      1412  2678 30$:
                                      1412  2679          $CLOSE  FAB = MA_K_CHKFAB(R6),- ; We're finished with the file for now
                                      1412  2680                  ERR = RMS_ERROR
                                      1421  2681 40$:
                                04    1421  2682          RET                             ; In fact, we're quite finished
```

```
                     1422  2684          .SBTTL  Mailbox $QIO Completion Status Routines
                     1422  2685 ;++
                     1422  2686 ; FUNCTIONAL DESCRIPTION:
                     1422  2687 ;       Check completion status when we send an interprocessor mailbox.  Print
                     1422  2688 ;       a message if there was an error.  Note that we have two entry points -
                     1422  2689 ;       one for synchronous sends and one for AST sends.
                     1422  2690 ;
                     1422  2691 ; CALLING SEQUENCE:
                     1422  2692 ;       CALLS   #1,QIOMB_%%%%%_CHECK
                     1422  2693 ;
                     1422  2694 ; INPUT PARAMETERS:
                     1422  2695 ;       06(AP) - Message type (word).
                     1422  2696 ;       05(AP) - Receiver's port number on shared memory (byte).
                     1422  2697 ;       04(AP) - Shared memory unit number (byte).
                     1422  2698 ;
                     1422  2699 ; IMPLICIT INPUTS:
                     1422  2700 ;       The IOSB from either synchronous or AST sending.
                     1422  2701 ;
                     1422  2702 ; OUTPUT PARAMETERS:
                     1422  2703 ;       NONE
                     1422  2704 ;
                     1422  2705 ; IMPLICIT OUTPUTS:
                     1422  2706 ;       NONE
                     1422  2707 ;
                     1422  2708 ; COMPLETION CODES:
                     1422  2709 ;       NONE
                     1422  2710 ;
                     1422  2711 ; SIDE EFFECTS:
                     1422  2712 ;       Message if there was a problem sending.
                     1422  2713 ;
                     1422  2714 ;--
                     1422  2715
                     1422  2716 QIOMB_SYNCH_CHECK:
                OFFC 1422  2717         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
     52  10D3'CF    3C  1424  2718         MOVZWL  SYNCH_MBISB,R2          ; Save synchronous unique data
           07       11  1429  2719         BRB     QIOMB_CHECK
                     142B  2720 QIOMB_AST_CHECK:
                OFFC 142B  2721         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
     52  112D'CF    3C  142D  2722         MOVZWL  AST_MBISB,R2            ; Save AST unique data
                     1432  2723
                     1432  2724 QIOMB_CHECK:
           4D 52     E8  143?  2725         BLBS    R2,10$                  ; Exit quickly if there was no error
              52     DD  1435  2726         PUSHL   R2                      ; Form a message...
        7E    06 AC    3C  1437  2727         MOVZWL  06(AP),-(SP)            ; ...with the error code...
        7E    05 AC    9A  143B  2728         MOVZBL  05(AP),-(SP)            ; ...and the info from ASTPRM...
        7E    04 AC    9A  143F  2729         MOVZBL  04(AP),-(SP)            ; ...on the stack...
           55    5E    D0  1443  2730         MOVL    SP,R5                   ; ...so that we don't interfere...
           FF30 CE    DF  1446  2731         PUSHAL  -TEXT_BUFFER-8(SP)      ; ...with any other messages...
     000000C8 8F      DD  144A  2732         PUSHL   #TEXT_BUFFER            ; ...that may be being formed...
           54    5E    D0  1450  2733         MOVL    SP,R4                   ; ...
  5E 000000C8 8F      C2  1453  2734         SUBL2   #TEXT_BUFFER,SP         ; ...
                     145A  2735         $FAOL_S CTRSTR = BAD_MB_QIO,-   ; ...
                     145A  2736                 OUTLEN = (R4),-
                     145A  2737                 OUTBUF = (R4),-
                     145A  2738                 PRMLST = (R5)
              54     DD  146B  2739         PUSHL   R4
              01     DD  146D  2740         PUSHL   #1
```

```
                                                B 5
                     00   EF  146F  2741          EXTZV   #STS$V_SEVERITY,-
          7E    52   03       1471  2742                  #STS$S_SEVERITY,R2,-(SP;
      6E  00741130   8F   C8  1474  2743          BISL2   #UETP$_TEXT,(SP)
  00000000'GF        03   FB  147B  2744          CALLS   #3,G^LIB$SIGNAL
                              1482  2745 10$:
                          04  1482  2746          RET
```

```
                                        1483  2748          .SBTTL  Mailbox Received AST Routine
                                        1483  2749 ;++
                                        1483  2750 ; FUNCTIONAL DESCRIPTION:
                                        1483  2751 ;       Interprocessor mailbox messages are received here.  Based on the
                                        1483  2752 ;       MB_W_WHY field, dispatch to a section which will process the message.
                                        1483  2753 ;
                                        1483  2754 ; CALLING SEQUENCE:
                                        1483  2755 ;       Called via AST upon receipt of a message to this processor's mailbox.
                                        1483  2756 ;
                                        1483  2757 ; INPUT PARAMETERS:
                                        1483  2758 ;       04(AP) = address of the node for the shared memory through which the
                                        1483  2759 ;                message was sent
                                        1483  2760 ;
                                        1483  2761 ; IMPLICIT INPUTS:
                                        1483  2762 ;       AST_MBISB - IOSB when mail is received
                                        1483  2763 ;       AST_MBBUF - Buffer into which mail is written
                                        1483  2764 ;
                                        1483  2765 ; OUTPUT PARAMETERS:
                                        1483  2766 ;       NONE
                                        1483  2767 ;
                                        1483  2768 ; IMPLICIT OUTPUTS:
                                        1483  2769 ;       NONE
                                        1483  2770 ;
                                        1483  2771 ; COMPLETION CODES:
                                        1483  2772 ;       NONE
                                        1483  2773 ;
                                        1483  2774 ; SIDE EFFECTS:
                                        1483  2775 ;       NONE
                                        1483  2776 ;
                                        1483  2777 ;--
                                        1483  2778
                                        1483  2779 RCVMB_AST:
                                  0FFC  1483  2780          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                        1485  2781
6D    1F17'CF    DE                   1485  2782          MOVAL   SSERROR,(FP)            ; Set up AST level exception handler
  56    04 AC    D0                   148A  2783          MOVL    04(AP),R6               ; Get the pointer to node for this MB
7E    110D'CF    B0                   148E  2784          MOVW    AST_MBBUF+MB_W_WHY,-(SP)  ; Set up QIOMB_AST_CHECK parameter...
7E    110B'CF    90                   1493  2785          MOVB    AST_MBBUF+MB_B_RECEIVER,-(SP) ; ...
7E    01EC C6    90                   1498  2786          MOVB    MA_Q_UNIT(R6),-(SP)     ; ...
  8A AF    01    FB                   149D  2787          CALLS   #1,QIOMB_AST_CHECK      ; Check $QIO completion status
      110B'CF    91                   14A1  2788          CMPB    AST_MBBUF+MB_B_RECEIVER,- ; Do a minimal consistency check...
      01AC C6                         14A5  2789                  MA_B_MYPORT(R6)         ; ...to see if message is for me
            43    12                  14A8  2790          BNEQ    40$                     ; BR if it wasn't
      110C'CF    91                   14AA  2791          CMPB    AST_MBBUF+MB_B_UNIT,-   ; More consistency checking
      01EC C6                         14AE  2792                  MA_Q_UNIT(R6)
            3A    12                  14B1  2793          BNEQ    40$                     ; BR if wrong unit
            0A    B1                  14B3  2794          CMPW    #MSG_ME2ME,-            ; I can legitimately send myself...
      110D'CF                         14B5  2795                  AST_MBBUF+MB_W_WHY      ; ...mail anytime...
            13    13                  14B8  2796          BEQL    20$                     ; ...so BR if I, in fact, did...
      062C C6    D5                   14BA  2797          TSTL    MA_Q_COMGSRAD(R6)       ; ...but other mail needs interproc GS
            0D    12                  14BE  2798          BNEQ    20$                     ; BR if interproc GS was set up
      110A'CF    9C                   14C0  2799          ROTL    AST_MBBUF+MB_B_SENDER,- ; It wasn't. Log that I...
      50    01                        14C4  2800                  #1,R0                   ; ...probably got mail...
01CC C6    50    C8                   14C6  2801          BISL2   R0,MA_L_OLDSTUFF(R6)    ; ...which was intended for...
                                        14CB  2802                                      ; ...an image which died suddenly
            4B    11                  14CB  2803          BRB     EXIT_FROM_RCV           ; Aside from that, ignore the mailbox
```

```
                                14CD  2805 20$:
            01   110D'CF   AF   14CD  2806          CASEW   AST_MBBUF+MB_W_WHY,#1,- ; Dispatch based on message type
                       0C       14D2  2807                  #MAX_MSG_TYPE
                                14D3  2808          .MACRO  X       MBCODE,DUMMY   ; A macro will define dispatch address
                                14D3  2809                  .WORD   RCV_'MBCODE-30$
                                14D3  2810          .ENDM   X
                                14D3  2811 30$:
                                14D3  2812          .SHOW   MEB
                                14D3  2813          MAILBOX_MSG_TYPES              ; Generate dispatch table
                         006D'  14D3                        .WORD   RCV_HELLO-30$
                         0152'  14D5                        .WORD   RCV_ICU_UCME-30$
                         0214'  14D7                        .WORD   RCV_BYEFORNOW-30$
                         0247'  14D9                        .WORD   RCV_BADCODE-30$
                         0287'  14DB                        .WORD   RCV_DOSET-30$
                         0357'  14DD                        .WORD   RCV_ISSET-30$
                         039D'  14DF                        .WORD   RCV_DOCLEAR-30$
                         046D'  14E1                        .WORD   RCV_ISCLEAR-30$
                         04B3'  14E3                        .WORD   RCV_INTERLOCK-30$
                         04F9'  14E5                        .WORD   RCV_ME2ME-30$
                         001A'  14E7                        .WORD   RCV_ME2MEDONE-30$
                         0528'  14E9                        .WORD   RCV_ILOSTU-30$
                         0542'  14EB                        .WORD   RCV_BYEBYE-30$
                                14ED  2814          .NOSHOW MEB
                                14ED  2815
                                14ED  2816 40$:
                                14ED  2817 RCV_ME2MEDONE:                          ; Should never get mail with this code
            7E   000C'CF   3C   14ED  2818          MOVZWL  BUFFER_PTR,-(SP)       ; Save stuff...
                 57   5E   D0   14F2  2819          MOVL    SP,R7                  ; ...that DO_FAO_SIGNAL...
                 5E   6E   C2   14F5  2820          SUBL2   (SP),SP                ; ...will overwrite at AST level...
          6E   0014'CF   67   28   14F8  2821          MOVC3   (R7),BUFFER,(SP)       ; ...which could be in progress
                 50   05E2'CF   DE   14FE  2822          MOVAL   BAD_MSG_TYPE,R0
                         FD80   30   1503  2823          BSBW    DO_FAO_SIGNAL          ; Warn that we have junk in our mailbox
          0014'CF   6E   67   28   1506  2824          MOVC3   (R7),(SP),BUFFER       ; Restore possible old message
                 5E   57   D0   150C  2825          MOVL    R7,SP                  ; That done, restore the stack...
                 0080 8F   BA   150F  2826          POPR    #^M<R7>                ; ...and...
               000C'CF   57   B0   1513  2827          MOVW    R7,BUFFER_PTR          ; ...other stuff which was clobbered
                                1518  2828          ;BRW    EXIT_FROM_RCV          ; Fall into common routine for exit
                                1518  2829
                                1518  2830 ;
                                1518  2831 ; All the routines dispatched via the above CASEW will exit through this little
                                1518  2832 ; routine.
                                1518  2833 ;
                                1518  2834 EXIT_FROM_RCV:
                                1518  2835          $QIO_S  CHAN    = MA_W_SPRMBCHN(R6),- ; Read my mailbox...
                                1518  2836                  FUNC    = #IO$_READVBLK,-     ; ...whenever someone writes to it
                                1518  2837                  IOSB    = AST_MBISB,-
                                1518  2838                  ASTADR  = RCVMB_AST,-
                                1518  2839                  ASTPRM  = R6,-
                                1518  2840                  P1      = AST_MBBUF,-
                                1518  2841                  P2      = #MB_K_END
                         04     153F  2842          RET
```

```
                                1540  2844 RCV_HELLO:
                                1540  2845 ;
                                1540  2846 ; Some processor is (re-)starting the test and wants to say hi.  It's sent
                                1540  2847 ; the date-time quadword that appears at the end of my private area in the
                                1540  2848 ; interprocessor global section.  If that turns out to be incorrect, let the
                                1540  2849 ; processor know by way of a return mailbox.  If the date-time is correct,
                                1540  2850 ; reply with a different message so saying and expect the processor to
                                1540  2851 ; acknowledge my message.  There is some code which will be common to either
                                1540  2852 ; path; get that out of the way first.  The code paths rejoin later.
                                1540  2853 ;
          110A'CF    90         1540  2854         MOVB    AST_MBBUF+MB_B_SENDER,- ; The proc which sent now will receive
          110B'CF                1544  2855                 AST_MBBUF+MB_B_RECEIVER
          01AC C6    90         1547  2856         MOVB    MA_B_MYPORT(R6),-       ; I am the sender
          110A'CF                154B  2857                 AST_MBBUF+MB_B_SENDER
          0258 C6    28         154E  2858         MOVC3   MA_Q_COMMBNAM(R6),-     ; Form the logical name...
113D'CF   0260 C6                1552  2859                 MA_T_COMMBNAM(R6),AST_MBNAM ; ...of the other processor's MB
1135'CF   0258 C6    D0         1558  2860         MOVL    MA_Q_COMMBNAM(R6),AST_MBPTR ; Form descriptor of...
1139'CF   113D'CF     DE         155F  2861         MOVAL   AST_MBNAM,AST_MBPTR+4   ; ...the logical name
     50   110B'CF    D0         1566  2862         MOVL    AST_MBBUF+MB_B_RECEIVER,R0 ; Include the port number...
             FCC3     30         156B  2863         BSBW    HEXONIT                 ; ...
       FF A3   50     90         156E  2864         MOVB    R0,-1(R3)               ; ...in the mailbox name
                                1572  2865         $CREMBX_S CHAN = AST_MBCHN,-    ; Establish a link to other proc's MB
                                1572  2866                   LOGNAM = AST_MBPTR
          110B'CF    F0         1589  2867         INSV    AST_MBBUF+MB_B_RECEIVER,- ; Set up common parts of REQIDT...
     57   08   08               158D  2868                 #8,#8,R7
     57   01EC C6    90         1590  2869         MOVB    MA_W_UNIT(R6),R7        ; ...and/or status check, below
                                1595  2870         ; End of common code
                                1595  2871
     01A4 C6   08    29         1595  2872         CMPC3   #8,MA_Q_DAYTIME(R6),-   ; Did I get passed my date-time?
          110F'CF               159A  2873                 AST_MBBUF+MB_K_MISC
                17    13         159D  2874         BEQL    10$                     ; BR if I did
          110B'CF    9C         159F  2875         ROTL    AST_MBBUF+MB_B_RECEIVER,- ; I got passed trash, so...
          50   01               15A3  2876                 #1,R0
     01BC C6   50    C8         15A5  2877         BISL2   R0,MA_L_SBADCODE(R6)    ; ...log the error
                04    B0         15AA  2878         MOVW    #MSG_BADCODE,-          ; Set up the reason for the mailbox
          110D'CF               15AC  2879                 AST_MBBUF+MB_W_WHY
     57   10   10    04    F0   15AF  2880         INSV    #MSG_BADCODE,#16,#16,R7 ; Finish info for status check, below
                36    11         15B4  2881         BRB     20$                     ; Rejoin common code
                                15B6  2882 10$:
                02    B0         15B6  2883         MOVW    #MSG_ICU_UCME,-         ; The reason for the mailbox
          110D'CF               15B8  2884                 AST_MBBUF+MB_W_WHY
115C'CF    FA0A1F00 8F    D0    15BB  2885         MOVL    #ICO_UCME_TIMEOUT,AST_MBDAY ; $SETIMR parameters - DAYTIM...
     1160'CF   00    D2         15C4  2886         MCOML   #0,AST_MBDAY+4          ; ...
     57   10   10    02    F0   15C9  2887         INSV    #MSG_ICU_UCME,#16,#16,R7 ; ...and REQIDT (also ststus check)
                                15CE  2888         $SETIMR_S DAYTIM = AST_MBDAY,-  ; We'll need a reply for this one
                                15CE  2889                   ASTADR = MB_TIMEOUT_AST,-
                                15CE  2890                   REQIDT = R7
          110B'CF    9C         15E1  2891         ROTL    AST_MBBUF+MB_B_RECEIVER,- ; We're going to reply...
          50   01               15E5  2892                 #1,R0
     01E0 C6   50    C8         15E7  2893         BISL2   R0,MA_L_ICU_UCME(R6)    ; ...so log that we expect a return
                                15EC  2894         ;BRB    20$                     ; Rejoin common code
                                15EC  2895 20$:
                                15EC  2896         $QIO_S  CHAN    = AST_MBCHN,-   ; Reply to the hello
                                15EC  2897                 FUNC    = #IO$_WRITEVBLK!IO$M_NOW,-
                                15EC  2898                 IOSB    = AST_MBISB,-
                                15EC  2899                 P1      = AST_MBBUF,-
                                15EC  2900                 P2      = #MB_K_END
```

```
                57   DD  160F  2901          PUSHL   R7                          ; Check completion status...
      FE15 CF   01   FB  1611  2902          CALLS   #1,QIOMB_AST_CHECK          ; ...of the $QIO
                     1616  2903          $DASSGN_S CHAN = AST_MBCHN           ; Free our hold on the mailbox
              FEF3   31  1622  2904          BRW     EXIT_FROM_RCV               ; Exit via common routine
```

```
                                1625  2906 RCV_ICU_UCME:
                                1625  2907 ;
                                1625  2908 ; A processor to which we've sent a hello message liked what we sent and is
                                1625  2909 ; trying to tell us so.  If we did indeed send the message, let the other
                                1625  2910 ; processor know that we received its mail.
                                1625  2911 ;
      50    110A'CF    9A     1625  2912         MOVZBL  AST_MBBUF+MB_B_SENDER,R0 ; Find out who sent us a message
   09 01DC C6    50    E4     162A  2913         BBSC    R0,MA_L_HELLO(R6),10$   ; BR if we expected it...
      50    01    50    9C     1630  2914         ROTL    R0,#1,R0                ; ...else...
      01C8 C6    50    C8     1634  2915         BISL2   R0,MA_L_NOCAUSE(R6)     ; ...log the error
                                1639  2916 10$:
   57   10    10    01    F0     1639  2917         INSV    #MSG_HELLO,#16,#16,R7   ; We've a reply to our hello...
57 08    08    110A'CF    F0     163E  2918         INSV    AST_MBBUF+MB_B_SENDER,#8,#8,R7 ; ...
      57    01EC C6    90     1645  2919         MOVB    MA_Q_UNIT(R6),R7        ; ...
                                164A  2920         $CANTIM_S REQIDT = R7           ; ...so we need not worry time out
                                1655  2921
            110A'CF    90     1655  2922         MOVB    AST_MBBUF+MB_B_SENDER,- ; The proc which sent now will receive
            110B'CF           1659  2923                 AST_MBBUF+MB_B_RECEIVER
            01AC C6    90     165C  2924         MOVB    MA_B_MYPORT(R6),-       ; I am the sender
            110A'CF           1660  2925                 AST_MBBUF+MB_B_SENDER
            0258 C6    28     1663  2926         MOVC3   MA_Q_COMMBNAM(R6),-     ; Form the logical name...
   113D'CF  0260 C6           1667  2927                 MA_T_COMMBNAM(R6),AST_MBNAM ; ...of the other processor's MB
   1135'CF  0258 C6    D0     166D  2928         MOVL    MA_Q_COMMBNAM(R6),AST_MBPTR ; Form descriptor of...
   1139'CF  113D'CF    DE     1674  2929         MOVAL   AST_MBNAM,AST_MBPTR+4   ; ...the logical name
      50    110B'CF    D0     167B  2930         MOVL    AST_MBBUF+MB_B_RECEIVER,R0 ; Include the port number...
               FBAE    30     1680  2931         BSBW    HEXONIT                 ; ...
         FF A3    50    90     1683  2932         MOVB    R0,-1(R3)               ; ...in the mailbox name
                                1687  2933         $CREMBX_S CHAN = AST_MBCHN,-    ; Establish a link to other proc's MB
                                1687  2934                 LOGNAM = AST_MBPTR
            03    B0     169E  2935         MOVW    #MSG_BYEFORNOW,-        ; The reason for the mailbox
      110D'CF           16A0  2936                 AST_MBBUF+MB_W_WHY
                                16A3  2937         $QIO_S  CHAN = AST_MBCHN,-      ; Reply to the hello
                                16A3  2938                 FUNC = #IO$_WRITEVBLK!IO$M_NOW,-
                                16A3  2939                 IOSB = AST_MBISB,-
                                16A3  2940                 P1   = AST_MBBUF,-
                                16A3  2941                 P2   = #MB_K_END
      7E    03    B0     16C6  2942         MOVW    #MSG_BYEFORNOW,-(SP)    ; Set up QIOMB_AST_CHECK parameter...
7E    110B'CF    90     16C9  2943         MOVB    AST_MBBUF+MB_B_RECEIVER,-(SP) ; ...
7E    01EC C6    90     16CE  2944         MOVB    MA_Q_UNIT(R6),-(SP)     ; ...
FD53  CF    01    FB     16D3  2945         CALLS   #1,QIOMB_AST_CHECK      ; Check $QIO completion status
                          16D8  2946         $DASSGN_S CHAN = AST_MBCHN      ; Free our hold on the mailbox
         FE31    31     16E4  2947         BRW     EXIT_FROM_RCV           ; Exit via common routine
```

```
                                16E7  2949 RCV_BYEFORNOW:
                                16E7  2950 ;
                                16E7  2951 ; We've exchanged messages with a processor to say that we can see each other
                                16E7  2952 ; and trade data via the interprocessor global section.  Clean up.
                                16E7  2953 ;
            50   110A'CF    9A  16E7  2954         MOVZBL  AST_MBBUF+MB_B_SENDER,R0 ; Find out who sent us a message
         09 01E0 C6    50    E4  16EC  2955         BBSC    R0,MA_L_ICU_UCME(R6),10$; BR if we expected it. .
            50   01    50    9C  16F2  2956         ROTL    R0,#1,R0                ; ...else...
            01C8 C6    50    C8  16F6  2957         BISL2   R0,MA_L_NOCAUSE(R6)     ; ...log the error
                                16FB  2958 10$:
      57    10   10    02    F0  16FB  2959         INSV    #MSG_ICU_UCME,#16,#16,R7 ; We've gotten a reply to our reply...
57    08    08   110A'CF    F0  1700  2960         INSV    AST_MBBUF+MB_B_SENDER,#8,#8,R7 ; ...
            57   01EC C6    90  1707  2961         MOVB    MA_Q_UNIT(R6),R7        ; ...
                                170C  2962         $CANTIM_S REQIDT = R7           ; ...so we need not worry time out
                                1717  2963
                   FDFE    31  1717  2964         BRW     EXIT_FROM_RCV           ; Exit via common routine
```

```
                                171A  2966 RCV_BADCODE:
                                171A  2967 ;
                                171A  2968 ; Some processor thinks we sent it a bum messsage.  Log the error.
                                171A  2969 ;
      57   10   10    01   F0   171A  2970          INSV    #MSG_HELLO,#16,#16,R7   ; We've a reply to our hello...
   57 08   08   110A'CF   F0   171F  2971          INSV    AST_MBBUF+MB_B_SENDER,#8,#8,R7 ; ...
           57   01EC C6   90   1726  2972          MOVB    MA_Q_UNIT(R6),R7        ; ...
                              1728  2973          $CANTIM_S REQIDT = R7           ; ...so we need not worry time out
                              1736  2974
           50   110A'CF   9A   1736  2975          MOVZBL  AST_MBBUF+MB_B_SENDER,R0 ; Find out who sent us a message
      0B 01DC C6    50   E4   173B  2976          BBSC    R0,MA_L_HELLO(R6),10$   ; BR if we expected it...
           50   01    50   9C   1741  2977          ROTL    R0,#1,R0                ; ...else...
           01C8 C6    50   C8   1745  2978          BISL2   R0,MA_L_NOCAUSE(R6)     ; ...log the error
                       0B   11   174A  2979          BRB     20$                     ; No need to log BADCODE if not our fault
                              174C  2980 10$:
                110A'CF   9C   174C  2981          ROTL    AST_MBBUF+MB_B_SENDER,- ; We sent trash...
                50   01        1750  2982                  #1,R0
           01B8 C6    50   C8   1752  2983          BISL2   R0,MA_L_RBADCODE(R6)    ; ...so log the error
                              1757  2984 20$:
                    FDBE   31   1757  2985          BRW     EXIT_FROM_RCV           ; Exit via common routine
```

```
                                175A  2987 RCV_DOSET:
                                175A  2988 ;
                                175A  2989 ; Some processor wants us to set a bit in the shared common event flag cluster.
                                175A  2990 ; Set the bit and reply.
                                175A  2991 ;
                                175A  2992         $ASCEFC_S EFN = #COMEF_EFN,-     ; Associate with interprocessor cluster
                                175A  2993                   NAME = MA_Q_COMEFNAM(R6)
      00000060 8F    C1         176F  2994         ADDL3   #COMEF_EFN,-            ; Figure which bit to set in it
     57   110F'CF               1775  2995                 AST_MBBUF+MB_K_MISC,R7
                                1779  2996         $SETEF_S EFN = R7               ; Set the requested bit
          110A'CF    91         1782  2997         CMPB    AST_MBBUF+MB_B_SENDER,- ; Did I ask myself to set the bit?
          01AC C6               1786  2998                 MA_B_MYPORT(R6)
                OD    13        1789  2999         BEQL    10$                     ; BR if so - I'll expect cluster on return
                                178B  3000         $DACEFC_S EFN = #COMEF_EFN      ; Release the cluster
                                1798  3001 10$:
                                1798  3002
          110A'CF    90         1798  3003         MOVB    AST_MBBUF+MB_B_SENDER,- ; The proc which sent now will receive
          110B'CF               179C  3004                 AST_MBBUF+MB_B_RECEIVER
          01AC C6    90         179F  3005         MOVB    MA_B_MYPORT(R6),-       ; I am the sender
          110A'CF               17A3  3006                 AST_MBBUF+MB_B_SENDER
     110D'CF   06    B0         17A6  3007         MOVW    #MSG_ISSET,AST_MBBUF+MB_W_WHY ; Say that we've set the bit
          0258 C6    28         17AB  3008         MOVC3   MA_Q_COMMBNAM(R6),-     ; Form the logical name...
113D'CF   0260 C6               17AF  3009                 MA_T_COMMBNAM(R6),AST_MBNAM ; ...of the other processor's MB
1135'CF   0258 C6    D0         17B5  3010         MOVL    MA_Q_COMMBNAM(R6),AST_MBPTR ; Form descriptor of...
1139'CF   113D'CF    DE         17BC  3011         MOVAL   AST_MBNAM,AST_MBPTR+4   ; ...the logical name
     50   110B'CF    D0         17C3  3012         MOVL    AST_MBBUF+MB_B_RECEIVER,R0 ; Include the port number...
             FA66    30         17C8  3013         BSBW    HEXONIT                 ; ...
       FF A3   50    90         17CB  3014         MOVB    R0,-1(R3)               ; ...in the mailbox name
                               17CF  3015         $CREMBX_S CHAN = AST_MBCHN,-    ; Establish a link to other proc's MB
                               17CF  3016                   LOGNAM = AST_MBPTR
                               17E6  3017         $QIO_S  CHAN = AST_MBCHN,-      ; Let the other proc know...
                               17E6  3018                 FUNC = #IO$_WRITEVBLK!IO$M_NOW,-
                               17E6  3019                 IOSB = AST_MBISB,-      ; ...that we've set the bit
                               17E6  3020                 P1   = AST_MBBUF,-
                               17E6  3021                 P2   = #MB_K_END
      7E   06    B0         1809  3022         MOVW    #MSG_ISSET,-(SP)        ; Set up QIOMB_AST_CHECK parameter...
7E    110B'CF    90         180C  3023         MOVB    AST_MBBUF+MB_B_RECEIVER,-(SP) ; ...
7E    01EC C6    90         1811  3024         MOVB    MA_Q_UNIT(R6),-(SP)     ; ...
FC10 CF    01    FB         1816  3025         CALLS   #1,QIOMB_AST_CHECK      ; Check $QIO completion status
                            181B  3026         $DASSGN_S CHAN = AST_MBCHN      ; Free our hold on the mailbox
          FCEE    31         1827  3027         BRW     EXIT_FROM_RCV           ; Exit via common routine
```

```
                                182A  3029 RCV_ISSET:
                                182A  3030 ;
                                182A  3031 ; A processor has let us know that a bit which we wanted to be set has been
                                182A  3032 ; set.  We really don't need to do anything about that here, but the
                                182A  3033 ; synchronous code testing the shared common event flag cluster needs to know
                                182A  3034 ; and is hibernating while waiting to be told.  Let the synchronous code know
                                182A  3035 ; that the bit has been set by modifying the synchronous mailbox buffer to the
                                182A  3036 ; extent of changing the reason for the mailbox.  Awaken the synchronous code.
                                182A  3037 ;
      50    110A'CF    9A      182A  3038          MOVZBL  AST_MBBUF+MB_B_SENDER,R0 ; Find out who sent us a message
   0C 01E4  C6   50    E4      182F  3039          BBSC    R0,MA_L_DOSET(R6),10$    ; BR if we expected it...
      50    01   50    9C      1835  3040          ROTL    R0,#1,R0                 ; ...else...
      01C8  C6   50    C8      1839  3041          BISL2   R0,MA_L_NOCAUSE(R6)      ; ...log the error
                FCD7    31      183E  3042          BRW     EXIT_FROM_RCV            ; Doing the rest could screw us up!
                                1841  3043 10$:
   57   10    10   05    F0      1841  3044          INSV    #MSG_DOSET,#16,#16,R7    ; We've gotten a reply to our...
57 08   08    110A'CF    F0      1846  3045          INSV    AST_MBBUF+MB_B_SENDER,#8,#8,R7 ; ...request for setting an EF
      57    01EC  C6    90      184D  3046          MOVB    MA_W_UNIT(R6),R7         ; ...
                                1852  3047          $CANTIM_S REQIDT = R7            ; ...so we need not worry time out
                                185D  3048
                      06    B0      185D  3049          MOVW    #MSG_ISSET,-             ; Change the reason for the mailbox...
                10B3'CF             185F  3050                  SYNCH_MBBUF+MB_W_WHY     ; ...in the synchronous buffer
                                1862  3051          $WAKE_S                          ; Get the synchronous code moving again
                   FCA8    31      1867  3052          BRW     EXIT_FROM_RCV            ; Exit via common routine
```

```
                                1870  3054 RCV_DOCLEAR:
                                1870  3055 ;
                                1870  3056 ; Some processor has requested that we clear a bit in the common event flag
                                1870  3057 ; cluster.  Clear the bit and reply.
                                1870  3058 ;
                                1870  3059         $ASCEFC_S EFN = #COMEF_EFN,-     ; Associate with interprocessor cluster
                                1870  3060                   NAME = MA_Q_COMEFNAM(R6)
          00000060 8F    C1     1885  3061         ADDL3   #COMEF_EFN,-             ; Figure which bit to clear in it
       57    110F'CF             188B  3062                 AST_MBBUF+MB_K_MISC,R7
                                188F  3063         $CLREF_S EFN = R7                ; Clear the requested bit
            110A'CF    91       1898  3064         CMPB    AST_MBBUF+MB_B_SENDER,- ; Did I ask myself to clear the bit?
            01AC C6             189C  3065                 MA_B_MYPORT(R6)
                  0D    13      189F  3066         BEQL    10$                      ; BR if so - I'll expect cluster on return
                                18A1  3067         $DACEFC_S EFN = #COMEF_EFN       ; Release the cluster
                                18AE  3068 10$:
                                18AE  3069
            110A'CF    90       18AE  3070         MOVB    AST_MBBUF+MB_B_SENDER,- ; The proc which sent now will receive
            110B'CF             18B2  3071                 AST_MBBUF+MB_B_RECEIVER
            01AC C6    90       18B5  3072         MOVB    MA_B_MYPORT(R6),-        ; I am the sender
            110A'CF             18B9  3073                 AST_MBBUF+MB_B_SENDER
                  08    B0      18BC  3074         MOVW    #MSG_ISCLEAR,-           ; Say that the bit is cleared
            110D'CF             18BE  3075                 AST_MBBUF+MB_W_WHY
            0258 C6    28       18C1  3076         MOVC3   MA_Q_COMMBNAM(R6),-          ; Form the logical name...
   113D'CF  0260 C6             18C5  3077                 MA_T_COMMBNAM(R6),AST_MBNAM ; ...of the other processor's MB
1135'CF     0258 C6    D0       18CB  3078         MOVL    MA_Q_COMMBNAM(R6),AST_MBPTR ; Form descriptor of...
1139'CF     113D'CF    DE       18D2  3079         MOVAL   AST_MBNAM,AST_MBPTR+4       ; ...the logical name
      50    110B'CF    D0       18D9  3080         MOVL    AST_MBBUF+MB_B_RECEIVER,R0 ; Include the port number...
              F950     30       18DE  3081         BSBW    HEXONIT                    ; ...
         FF A3   50    90       18E1  3082         MOVB    R0,-1(R3)                  ; ...in the mailbox name
                                18E5  3083         $CREMBX_S CHAN = AST_MBCHN,-     ; Establish a link to other proc's MB
                                18E5  3084                   LOGNAM = AST_MBPTR
                                18FC  3085         $QIO_S  CHAN = AST_MBCHN,-       ; Let the other proc know...
                                18FC  3086                 FUNC = #IO$_WRITEVBLK!IO$M_NOW,-
                                18FC  3087                 IOSB = AST_MBISB,-       ; ...that the bit is cleared
                                18FC  3088                 P1   = AST_MBBUF,-
                                18FC  3089                 P2   = #MB_K_END
         7E    08    B0       191F  3090         MOVW    #MSG_ISCLEAR,-(SP)       ; Set up QIOMB_AST_CHECK parameter...
   7E    110B'CF    90       1922  3091         MOVB    AST_MBBUF+MB_B_RECEIVER,-(SP) ; ...
   7E    01EC C6    90       1927  3092         MOVB    MA_Q_UNIT(R6),-(SP)          ; ...
   FAFA  CF    01    FB       192C  3093         CALLS   #1,QIOMB_AST_CHECK       ; Check $QIO completion status
                              1931  3094         $DASSGN_S CHAN = AST_MBCHN       ; Free our hold on the mailbox
                 FED8 31       193D  3095         BRW     EXIT_FROM_RCV            ; Exit via common routine
```

```
                                1940  3097 RCV_ISCLEAR:
                                1940  3098 ;
                                1940  3099 ; A processor has let us know that a bit which we wanted to be cleared has been
                                1940  3100 ; cleared.  We really don't need to do anything about that here, but the
                                1940  3101 ; synchronous code testing the shared common event flag cluster needs to know
                                1940  3102 ; and is hibernating while waiting to be told.  Let the synchronous code know
                                1940  3103 ; that the bit has been cleared by modifying the synchronous mailbox buffer to
                                1940  3104 ; the extent of changing the reason for the mailbox.  Awaken the synchronous
                                1940  3105 ; code.
                                1940  3106 ;
      50    110A'CF    9A     1940  3107         MOVZBL  AST_MBBUF+MB_B_SENDER,R0 ; Find out who sent us a message
   0C 01E8 C6    50    E4     1945  3108         BBSC    R0,MA_L_DOCLEAR(R6),10$ ; BR if we expected it...
      50    01    50    9C     194B  3109         ROTL    R0,#1,R0                ; ...else...
      01C8 C6    50    C8     194F  3110         BISL2   R0,MA_L_NOCAUSE(R6)     ; ...log the error
            FBC1    31        1954  3111         BRW     EXIT_FROM_RCV           ; Doing the rest could screw us up!
                              1957  3112 10$:
   57   10    10    07    F0     1957  3113         INSV    #MSG_DOCLEAR,#16,#16,R7 ; We've gotten a reply to our...
57 08   08    110A'CF    F0     195C  3114         INSV    AST_MBBUF+MB_B_SENDER,#8,#8,R7 ; ...request for clearing an EF
      57    01EC C6    90     1963  3115         MOVB    MA_B_UNIT(R6),R7        ; ...
                              1968  3116         $CANTIM_S REQIDT = R7           ; ...so we need not worry time out
                              1973  3117
            08    B0        1973  3118         MOVW    #MSG_ISCLEAR,-          ; Change the reason for the mailbox...
         10B3'CF              1975  3119                 SYNCH_MBBUF+MB_W_WHY    ; ...in the synchronous buffer
                              1978  3120         $WAKE_S                         ; Get the synchronous code moving again
            FB92    31        1983  3121         BRW     EXIT_FROM_RCV           ; Exit via common routine
```

```
                                                N 5
                     1986  3123 RCV_INTERLOCK:
                     1986  3124 ;
                     1986  3125 ; Some processor wants to test memory interlocking instructions.  A valid test
                     1986  3126 ; depends on all processors cooperating in real time, in order that there be a
                     1986  3127 ; chance that there will be a conflict in accessing a memory location.  When
                     1986  3128 ; we receive an interlock request mailbox, we assume that the requesting
                     1986  3129 ; processor has already begun the test.  In order to synchronize, we have to
                     1986  3130 ; begin the test on this processor as soon as possible.  This means
                     1986  3131 ; interrupting whatever else is going on synchronously and starting this part
                     1986  3132 ; of the test.  AST delivery of the mailbox does the first part.  However,
                     1986  3133 ; the interlocking test runs for several real seconds and we'd have troubles
                     1986  3134 ; with staying at AST level that long because it effectively disables ASTs
                     1986  3135 ; which might come from processors connected to other shared memories.  To
                     1986  3136 ; allow interruption of synchronous testing, but not stay at AST level, we
                     1986  3137 ; cheat somewhat:  we use an undocumented system service to dismiss the AST
                     1986  3138 ; and then call the test routine.  However, one of those interruptions
                     1986  3139 ; could be from a processor on another memory wanting to start this test on
                     1986  3140 ; that memory!  That would be pointless; the first test would be stopped dead
                     1986  3141 ; while the second ran and then the first test could run with no competition.
                     1986  3142 ; To prevent that sort of thing and to pass and save vital information, R6
                     1986  3143 ; (the pointer into UNIT_LIST we receive via ASTPRM) is saved in a location
                     1986  3144 ; which is accessed by the PC and not by indexing through UNIT_LIST or in a
                     1986  3145 ; interprocessor global section.  This location is used as a lock.
                     1986  3146 ;
1164'CF    D5        1986  3147          TSTL    INTERLOCK_SAVED_R6      ; Are we being called recursively?
     03    13        198A  3148          BEQL    10$                     ; BR if not
   FB89    31        198C  3149          BRW     EXIT_FROM_RCV           ; Don't play ball if we are
                     198F  3150 10$:
                     198F  3151          $QIO_S  CHAN    = MA_W_SPRMBCHN(R6),- ; Reset to allow mailbox reads...
                     198F  3152                  FUNC    = #IO$_READVBLK,-     ; ...during the interlock test
                     198F  3153                  IOSB    = AST_MBISB,-
                     198F  3154                  ASTADR  = RCVMB_AST,-
                     198F  3155                  ASTPRM  = R6,-
                     198F  3156                  P1      = AST_MBBUF,-
                     198F  3157                  P2      = #MB_K_END
   1164'CF    56    D0    19B6  3158          MOVL    R6,INTERLOCK_SAVED_R6   ; Prevent recursion and save info...
00000000'GF    00    FB    19BB  3159          CALLS   #0,G^SYS$CLRAST         ; ...because the AST is dismissed
   F730 CF    00    FB    19C2  3160          CALLS   #0,INTERLOCKING_TEST    ; Perform the interlock test
        1164'CF    D4    19C7  3161          CLRL    INTERLOCK_SAVED_R6      ; Allow us to thrash any memory again
                   04    19CB  3162          RET                             ; Rejoin mainline code
```

```
                                19CC  3164 RCV_ME2ME:
                                19CC  3165 ;
                                19CC  3166 ; We've tested that it's possible to send mail to our own processor.  To let
                                19CC  3167 ; the synchronous code know that we've been successful, change the reason for
                                19CC  3168 ; the mailbox (MB_W_WHY) in the synchronous mailbox buffer and wake up the
                                19CC  3169 ; synchronous code, which has been waiting for us to do exactly that.
                                19CC  3170 ;
      57   10   10   0A   F0   19CC  3171          INSV    #MSG_ME2ME,#16,#16,R7    ; We've gotten our own mailbox..
57    08   08   01AC C6   F0   19D1  3172          INSV    MA_B_MYPORT(R6),#8,#8,R7 ; ...
           57   01EC C6   90   19D8  3173          MOVB    MA_W_UNIT(R6),R7         ; ...
                                19DD  3174          $CANTIM_S REQIDT = R7            ; ...so we need not worry time out
                                19E8  3175
                          0B   B0   19E8  3176          MOVW    #MSG_ME2MEDONE,-         ; Change the reason for the mailbox...
                    10B3'CF         19EA  3177                  SYNCH_MBBUF+MB_W_WHY     ; ...in the synchronous buffer
                                19ED  3178          $WAKE_S                          ; Get the synchronous code moving again
                       FB1D   31   19F8  3179          BRW     EXIT_FROM_RCV            ; Exit via common routine
```

```
                     19FB  3181 RCV_ILOSTU:
                     19FB  3182 ;
                     19FB  3183 ; For some reason, a processor with which we were communicating thinks we
                     19FB  3184 ; went away.  Our entry in the interprocessor global section bit mask of
                     19FB  3185 ; available processors has been cleared!  Log the error and reset the bit.
                     19FB  3186 ;
      110A'CF   9C   19FB  3187          ROTL    AST_MBBUF+MB_B_SENDER,- ; We were too slow...
      50    01       19FF  3188                  #1,R0
01B4  C6    50   C8  1A01  3189          BISL2   R0,MA_L_RILOSTU(R6)     ; ...so log the error
57    062C  C6   D0  1A06  3190          MOVL    MA_Q_COMGSRAD(R6),R7    ; Point to start of interprocessor GS
      01AC  C6   E6  1A0B  3191          BBSSI   MA_B_MYPORT(R6),-       ; We may have just been too slow...
     00 24  A7        1A0F  3192                  GS_L_AVAILABLE(R7),10$  ; ...tell others we're really here
                     1A12  3193 10$:
        FB03   31    1A12  3194          BRW     EXIT_FROM_RCV           ; Exit via common routine
```

```
                                1A15  3196 RCV_BYEBYE:
                                1A15  3197 ;
                                1A15  3198 ; Some processor is terminating execution.  We'd like to get consistent so that
                                1A15  3199 ; there is minimal impact due to its going away.  In particular, we no longer
                                1A15  3200 ; expect any mailbox messages from that processor.
                                1A15  3201 ;
                110A'CF   9C    1A15  3202         ROTL    AST_MBBUF+MB_B_SENDER,- ; Figure out who's going awa,
                50   01         1A19  3203                 #1,R0
          01DC  C6   50   CA    1A1B  3204         BICL2   R0,MA_L_HELLO(R6)       ; We no longer expect any replies...
          01E0  C6   50   CA    1A20  3205         BICL2   R0,MA_L_ICU_UCME(R6)    ; ...
          01E4  C6   50   CA    1A25  3206         BICL2   R0,MA_L_DOSET(R6)       ; ...
          01E8  C6   50   CA    1A2A  3207         BICL2   R0,MA_L_DOCLEAR(R6)     ; ...
    57    08    08   110A'CF   F0    1A2F  3208         INSV    AST_MBBUF+MB_B_SENDER,#8,#8,R7 ; Insert stuff that...
          57    01EC C6   90    1A36  3209         MOVB    MA_B_UNIT(R6),R7        ; ...will be the same for all $CANTIMs
                                1A3B  3210
          57    10   10   01    F0    1A3B  3211         INSV    #MSG_HELLO,#16,#16,R7   ; Cancel critical timer requests...
                                1A40  3212         $CANTIM_S REQIDT = R7           ; ...for the departing processor
                                1A4B  3213
          57    10   10   02    F0    1A4B  3214         INSV    #MSG_ICU_UCME,#16,#16,R7
                                1A50  3215         $CANTIM_S REQIDT = R7
                                1A5B  3216
                                1A5B  3217         $GETJPI_S ITMLST = GETJPI_ITMLST ; Get the current timer queue count
                1168'CF   DD    1A70  3218         PUSHL   TQCNT                   ; Save it for later comparison
          57    10   10   05    F0    1A74  3219         INSV    #MSG_DOSET,#16,#16,R7
                                1A79  3220         $CANTIM_S REQIDT = R7
                                1A84  3221         $GETJPI_S ITMLST = GETJPI_ITMLST ; Get the new timer queue count
          8E    1168'CF   D1    1A99  3222         CMPL    TQCNT,(SP)+             ; Has it changed?
                      0B  13    1A9E  3223         BEQL    10$                     ; BR if not - no request outstanding
                                1AA0  3224         $WAKE_S                         ; We have a $HIBER pending
                                1AAB  3225 10$:
                                1AAB  3226
                                1AAB  3227         $GETJPI_S ITMLST = GETJPI_ITMLST ; Get the current timer queue count
                1168'CF   DD    1AC0  3228         PUSHL   TQCNT                   ; Save it for later comparison
    57    10    10   07    F0    1AC4  3229         INSV    #MSG_DOCLEAR,#16,#16,R7
                                1AC9  3230         $CANTIM_S REQIDT = R7
                                1AD4  3231         $GETJPI_S ITMLST = GETJPI_ITMLST ; Get the new timer queue count
          8E    1168'CF   D1    1AE9  3232         CMPL    TQCNT,(SP)+             ; Has it changed?
                      0B  13    1AEE  3233         BEQL    20$                     ; BR if not - no request outstanding
                                1AF0  3234         $WAKE_S                         ; We have a $HIBER pending
                                1AFB  3235 20$:
                                1AFB  3236
                   FA1A   31    1AFB  3237         BRW     EXIT_FROM_RCV           ; Exit via common routine
```

```
                        1AFE  3239          .SBTTL  Mailbox Timer Expiration Routine
                        1AFE  3240 ;++
                        1AFE  3241 ; FUNCTIONAL DESCRIPTION:
                        1AFE  3242 ;       This routine runs when the timer goes off to indicate that another
                        1AFE  3243 ;       processor hasn't replied to a mailbox.
                        1AFE  3244 ;
                        1AFE  3245 ; CALLING SEQUENCE:
                        1AFE  3246 ;       Called via AST at $SETIMR expiration.
                        1AFE  3247 ;
                        1AFE  3248 ; INPUT PARAMETERS:
                        1AFE  3249 ;       04(AP) = .BYTE  shared memory unit number (MA_W_UNIT)
                        1AFE  3250 ;                .BYTE  other processor's port on that memory (MB_B_RECEIVER)
                        1AFE  3251 ;                .WORD  reason for sending mail (MB_W_WHY)
                        1AFE  3252 ;
                        1AFE  3253 ; IMPLICIT INPUTS:
                        1AFE  3254 ;       NONE
                        1AFE  3255 ;
                        1AFE  3256 ; OUTPUT PARAMETERS:
                        1AFE  3257 ;       NONE
                        1AFE  3258 ;
                        1AFE  3259 ; IMPLICIT OUTPUTS:
                        1AFE  3260 ;       NONE
                        1AFE  3261 ;
                        1AFE  3262 ; COMPLETION CODES:
                        1AFE  3263 ;       NONE
                        1AFE  3264 ;
                        1AFE  3265 ; SIDE EFFECTS:
                        1AFE  3266 ;       NONE
                        1AFE  3267 ;
                        1AFE  3268 ;--
                        1AFE  3269
                        1AFE  3270 MB_TIMEOUT_AST:
                  0FFC  1AFE  3271          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        1B00  3272
          6D  1F17'CF   DE  1B00  3273          MOVAL   SSERROR,(FP)                ; Set up AST level exception handler
56  01D8'CF  00001D8'8F  C1  1B05  3274          ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; Must search for correct node
                        1B0F  3275 10$:
        01EC C6   04 AC   91  1B0F  3276          CMPB    04(AP),MA_W_UNIT(R6)      ; Match correct memory?
                     0F   13  1B15  3277          BEQL    20$                       ; BR if we have
                  56   66   C0  1B17  3278          ADDL2   (R6),R6                   ; Point to the next one if we haven't
        56  00001D8'8F      D1  1B1A  3279          CMPL    #UNIT_LIST,R6             ; Searched all possibilities yet?
                     EC   12  1B21  3280          BNEQ    10$                       ; BR if not - we're OK
                   0024   31  1B23  3281          BRW     40$                       ; BR because of bad REQIDT
                        1B26  3282 20$:
        50   01   05 AC   9C  1B26  3283          ROTL    05(AP),#1,R0              ; Supply a mask for those timeouts...
                        1B2B  3284                                                ; ...which need to clear a bit...
                        1B2B  3285                                                ; ...in an event expected longword
```

```
                0C   01    06 AC    AF   1B2B  3287          CASEW   06(AP),#1,#MAX_MSG_TYPE ; Dispatch based on message type
                                     1B30  3288          .MACRO  X       MBCODE,DUMMY    ; A macro will define dispatch address
                                     1B30  3289                  .WORD   TMO_'MBCODE-30$
                                     1B30  3290          .ENDM   X
                                     1B30  3291 30$:
                                     1B30  3292          .SHOW   MEB
                                     1B30  3293          MAILBOX_MSG_TYPES               ; Generate dispatch table
                           0107'     1B30                        .WORD   TMO_HELLO-30$
                           010F'     1B32                        .WORD   TMO_ICU_UCME-30$
                           001A'     1B34                        .WORD   TMO_BYEFORNOW-30$
                           001A'     1B36                        .WORD   TMO_BADCODE-30$
                           00ED'     1B38                        .WORD   TMO_DOSET-30$
                           001A'     1B3A                        .WOR    TMO_ISSET-30$
                           00F4'     1B3C                        .WORD   TMO_DOCLEAR-30$
                           001A'     1B3E                        .WORD   TMO_ISCLEAR-30$
                           001A'     1B40                        .WORD   TMO_INTERLOCK-30$
                           00F9'     1B42                        .WORD   TMO_ME2ME-30$
                           001A'     1B44                        .WORD   TMO_ME2MEDONE-30$
                           001A'     1B46                        .WORD   TMO_ILOSTU-30$
                           001A'     1B48                        .WORD   TMO_BYEBYE-30$
                                     1B4A  3294          .NOSHOW MEB
                                     1B4A  3295 40$:
                                     1B4A  3296          .MACRO  X       MBCODE,SECONDS  ; A macro to define unused labels
                                     1B4A  3297                  .IF B SECONDS
                                     1B4A  3298                      TMO_'MBCODE = 40$
                                     1B4A  3299                  .ENDC ; B SECONDS
                                     1B4A  3300          .ENDM   X
                                     1B4A  3301          MAILBOX_MSG_TYPES               ; Generate labels
                        06D2'CF      1B4A  3302          PUSHAL  BAD_REQIDT              ; If we reach here...
                             01      1B4E  3303          PUSHL   #1                      ; ...we got a phoney AST
                       00741130 8F   1B50  3304          PUSHL   #UETP$_TEXT!STS$K_WARNING
                  00000000'GF   03   1B56  3305          CALLS   #3,G^LIB$SIGNAL
                                     1B5D  3306          :BRW    EXIT_FROM_TMO           ; Fall into common routine for exit
                                     1B5D  3307
                                     1B5D  3308 ;
                                     1B5D  3309 ; All the routines dispatched via the above CASEW will exit through this little
                                     1B5D  3310 ; routine.
                                     1B5D  3311 ;
                                     1B5D  3312 EXIT_FROM_TMO:
                     06 AC    0A     1B5D  3313          CMPW    #MSG_ME2ME,06(AP)       ; Was this a single processor message?
                              14     1B61  3314          BEQL    10$                     ; BR if so - GS may not be there
                     06 AC    0B     1B63  3315          CMPW    #MSG_ME2MEDONE,06(AP)   ; Catch pathological case
                              0E     1B67  3316          BEQL    10$                     ; BR if so - again GS may not be there
                   57    062C C6     1B69  3317          MOVL    MA_Q_COMGSRAD(R6),R7    ; Point to start of global section
                     58    05 AC     1B6E  3318          MOVZBL  05(AP),R8
                00 24 A7    58 E7     1B72  3319          BBCCI   R8,GS_L_AVAILABLE(R7),10$ ; Assume receiver processor went away
                                     1B77  3320 10$:
                         01AC C6     1B77  3321          MOVB    MA_B_MYPORT(R6),-       ; Send a MB so saying - sender...
                         110A'CF     1B7B  3322                  AST_MBBUF+MB_B_SENDER
                           05 AC     1B7E  3323          MOVB    05(AP),-                ; ...receiver...
                         110B'CF     1B81  3324                  AST_MBBUF+MB_B_RECEIVER ; ...(the one we think is gone)...
                         01EC C6     1B84  3325          MOVB    MA_Q_UNIT(R6),-         ; ...the shared memory involved...
                         110C'CF     1B88  3326                  AST_MBBUF+MB_B_UNIT
                              0C     1B8B  3327          MOVW    #MSG_ILOSTU,-           ; ...the reason for the mailbox...
                         110D'CF     1B8D  3328                  AST_MBBUF+MB_W_WHY
                           06 AC     1B90  3329          MOVW    06(AP),-                ; ...and what we originally sent
                         110F'CF     1B93  3330                  AST_MBBUF+MB_K_MISC
```

```
                0258 C6    28  1B96  3331          MOVC3   MA_Q_COMMBNAM(R6),-          ; Form the logical name...
113D'CF         0260 C6        1B9A  3332                  MA_T_COMMBNAM(R6),AST_MBNAM ; ...of the other processor's MB
1135'CF         0258 C6    D0  1BA0  3333          MOVL    MA_Q_COMMBNAM(R6),AST_MBPTR ; Form descriptor of...
1139'CF     113D'CF        DE  1BA7  3334          MOVAL   AST_MBNAM,AST_MBPTR+4       ; ...the logical name
          50    05 AC      9A  1BAE  3335          MOVZBL  05(AP),R0                   ; Include the port number...
              F67C         30  1BB2  3336          BSBW    HEXUNIT                     ; ...
        FF A3    50        90  1BB5  3337          MOVB    R0,-1(R3)                   ; ...in the mailbox name
                           1BB9  3338          $CREMBX_S CHAN = AST_MBCHN,-         ; Establish a link to other proc's MB
                           1BB9  3339                    LOGNAM = AST_MBPTR
                           1BD0  3340          $QIO_S  CHAN = AST_MBCHN,-           ; Let the other proc know...
                           1BD0  3341                  FUNC = #IO$_WRITEVBLK!IO$M_NOW,-
                           1BD0  3342                  IOSB = AST_MBISB,-           ; ...that its bit is cleared
                           1BD0  3343                  P1   = AST_MBBUF,-
                           1BD0  3344                  P2   = #MB_K_END
          7E    0C     B0  1BF3  3345          MOVW    #MSG_ILOSTD,-(SP)           ; Set up QIOMB_AST_CHECK parameter...
        7E    110B'CF    90  1BF6  3346          MOVB    AST_MBBUF+MB_B_RECEIVER,-(SP) ; ...
        7E    01EC C6    90  1BFB  3347          MOVB    MA_Q_UNIT(R6),-(SP)         ; ...
        F826  CF    01   FB  1C00  3348          CALLS   #1,QIOMB_AST_CHECK          ; Check $QIO completion status
                           1C05  3349          $DASSGN_S CHAN = AST_MBCHN           ; Free our hold on the mailbox
              110B'CF      9C  1C11  3350          ROTL    AST_MBBUF+MB_B_RECEIVER,- ; Processor didn't respond...
              50    01         1C15  3351                  #1,R0
        01B0 C6     50     C8  1C17  3352          BISL2   R0,MA_L_SILOSTU(R6)        ; ...so log the error
                           04  1C1C  3353          RET
```

```
                                                ;
                                                ; Save some code on this page by taking advantage of the simplicity of these
                                                ; routines.  DOSET, DOCLEAR and ME2ME need to awaken us from hibernation.
                                                ; The others need only say that certain events are no longer expected;
                                                ; everything else that needs to be done is handled by the common exit routine.
                                                ;


                                                TMO_DOSET:
                                                ;
                                                ; Some processor didn't set a bit in the shared common event flag.
                                                ;
01E4 C6   50    CA                              BICL2   R0,MA_L_DOSET(R6)       ; Don't expect flag to be set
          05    11                              BRB     COMMON_WAKE_TMO


                                                TMO_DOCLEAR:
                                                ;
                                                ; Some processor didn't clear a bit in the shared common event flag.
                                                ;
01E8 C6   50    CA                              BICL2   R0,MA_L_DOCLEAR(R6)     ; Don't expect flag to be cleared
                                                ;BRB    COMMON_WAKE_TMO


                                                TMO_ME2ME:
                                                ;
                                                ; I couldn't send mail to myself.
                                                ;


                                                COMMON_WAKE_TMO:
                                                        $WAKE_S                 ; Get the test going again
          FF26  31                              BRW     EXIT_FROM_TMO


                                                TMO_HELLO:
                                                ;
                                                ; Some processor didn't respond to our friendly hello.
                                                ;
01DC C6    50   CA                              BICL2   R0,MA_L_HELLO(R6)       ; No longer expect other processor
          FF1E  31                              BRW     EXIT_FROM_TMO


                                                TMO_ICU_UCME:
                                                ;
                                                ; Some processor didn't let us know that we could see it, too.
                                                ;
01E0 C6    50   CA                              BICL2   R0,MA_L_ICU_UCME(R6)    ; No longer expect other processor
          FF16  31                              BRW     EXIT_FROM_TMO
```

```
                                                1C47  3404          .SBTTL  Print Error Summary Routine
                                                1C47  3405 ;++
                                                1C47  3406 ; FUNCTIONAL DESCRIPTION:
                                                1C47  3407 ;       This routine is called at the end of a thre-minute test iteration or
                                                1C47  3408 ;       one shot pass to print whatever information we may have stored
                                                1C47  3409 ;       regarding errors during the test.
                                                1C47  3410 ;
                                                1C47  3411 ; CALLING SEQUENCE:
                                                1C47  3412 ;       CALLS   #0,ERROR_SUMMARY
                                                1C47  3413 ;
                                                1C47  3414 ; INPUT PARAMETERS:
                                                1C47  3415 ;       NONE
                                                1C47  3416 ;
                                                1C47  3417 ; IMPLICIT INPUTS:
                                                1C47  3418 ;       UNIT_LIST points to the list of nodes, one for each memory.  Within
                                                1C47  3419 ;       each node, there are several mask words, each of which corresponds to
                                                1C47  3420 ;       a particular condition.  The bits in each mask correspond to the ports
                                                1C47  3421 ;       of the shared memory attached to processors.  A bit is lit if the
                                                1C47  3422 ;       condition applies to that processor.
                                                1C47  3423 ;
                                                1C47  3424 ; OUTPUT PARAMETERS:
                                                1C47  3425 ;       NONE
                                                1C47  3426 ;
                                                1C47  3427 ; IMPLICIT OUTPUTS:
                                                1C47  3428 ;       NONE
                                                1C47  3429 ;
                                                1C47  3430 ; COMPLETION CODES:
                                                1C47  3431 ;       NONE
                                                1C47  3432 ;
                                                1C47  3433 ; SIDE EFFECTS:
                                                1C47  3434 ;       Messages to SYS$OUTPUT.  Masks described above are cleared out.
                                                1C47  3435 ;
                                                1C47  3436 ;--
                                                1C47  3437
                                                1C47  3438 ERROR_SUMMARY:
                                      0FFC      1C47  3439          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                                1C49  3440
 56   01D8'CF    000001D8'8F    C1      1C49  3441          ADDL3   #UNIT_LIST.UNIT_LIST,R6 ; R6 points to node for current memory
                                                1C53  3442 10$:
                          2C    00      3B      1C53  3443          SKPC    #0,#MA_K_ERRLEN,-       ; See if we have...
                        01B0 C6                 1C56  3444                  MA_L_ERRBLK(R6)         ; ...anything to report for this memory
                             03         12      1C59  3445          BNEQ    20$                     ; BR if we found a problem
                           007B         31      1C5B  3446          BRW     100$                    ; BR if it's clean
                                                1C5E  3447 20$:
                50   07A0'CF            DE      1C5E  3448          MOVAL   SUMMARY_HEADER,R0       ; We've some error, print a header
                           F624         30      1C63  3449          BSBW    DO_FAO_INFO
```

```
                50   01B4 C6   D0  1C66  3451          MOVL    MA_L_RILOSTU(R6),R0      ; Did someone think we went away?
                       08      13  1C6B  3452          BEQL    30$                      ; BR if not
                57   07D5'CF   DE  1C6D  3453          MOVAL   RILOSTU_MSG,R7           ; Yes, print appropriate message
                     00AA      30  1C72  3454          BSBW    200$
                                   1C75  3455 30$:
                50   01B0 C6   D0  1C75  3456          MOVL    MA_L_SILOSTU(R6),R0      ; Did we think someone else went away?
                       08      13  1C7A  3457          BEQL    40$                      ; BR if not
                57   0836'CF   DE  1C7C  3458          MOVAL   SILOSTU_MSG,R7           ; Yes, print appropriate message
                     009B      30  1C81  3459          BSBW    200$
                                   1C84  3460 40$:
                50   01B8 C6   D0  1C84  3461          MOVL    MA_L_RBADCODE(R6),R0     ; Did we send someone a trashy HELLO?
                       08      13  1C89  3462          BEQL    50$                      ; BR if not
                57   088D'CF   DE  1C8B  3463          MOVAL   RBADCODE_MSG,R7          ; Yes, print appropriate message
                     008C      30  1C90  3464          BSBW    200$
                                   1C93  3465 50$:
                50   01BC C6   D0  1C93  3466          MOVL    MA_L_SBADCODE(R6),R0     ; Did someone send us a trashy HELLO?
                       07      13  1C98  3467          BEQL    60$                      ; BR if not
                57   0905'CF   DE  1C9A  3468          MOVAL   SBADCODE_MSG,R7          ; Yes, print appropriate message
                       7E      10  1C9F  3469          BSBB    200$
                                   1CA1  3470 60$:
                50   01C0 C6   D0  1CA1  3471          MOVL    MA_L_DOSETFAIL(R6),R0    ; Did someone fail to set a bit?
                       07      13  1CA6  3472          BEQL    70$                      ; BR if not
                57   0978'CF   DE  1CA8  3473          MOVAL   DOSET_MSG,R7             ; Yes, print appropriate message
                       70      10  1CAD  3474          BSBB    200$
                                   1CAF  3475 70$:
                50   01C4 C6   D0  1CAF  3476          MOVL    MA_L_DOCLEARFAIL(R6),R0  ; Did someone fail to clear a bit?
                       07      13  1CB4  3477          BEQL    80$                      ; BR if not
                57   09C3'CF   DE  1CB6  3478          MOVAL   DOCLEAR_MSG,R7           ; Yes, print appropriate message
                       62      10  1CBB  3479          BSBB    200$
                                   1CBD  3480 80$:
                50   01C8 C6   D0  1CBD  3481          MOVL    MA_L_NOCAUSE(R6),R0      ; Did we get an unexpected message?
                       07      13  1CC2  3482          BEQL    90$                      ; BR if not
                57   0A10'CF   DE  1CC4  3483          MOVAL   NOCAUSE_MSG,R7           ; Yes, print appropriate message
                       54      10  1CC9  3484          BSBB    200$
                                   1CCB  3485 90$:
                50   01CC C6   D0  1CCB  3486          MOVL    MA_L_OLDSTUFF(R6),R0     ; Did we get msg before we were ready?
                       07      13  1CD0  3487          BEQL    100$                     ; BR if not
                57   0A72'CF   DE  1CD2  3488          MOVAL   OLDSTUFF_MSG,R7          ; Yes, print appropriate message
                       46      10  1CD7  3489          BSBB    200$
                                   1CD9  3490 100$:
                50   01D0 C6   D0  1CD9  3491          MOVL    MA_L_INTLKTMO(R6),R0     ; Did everyone finish interlock test?
                       07      13  1CDE  3492          BEQL    110$                     ; BR if so
                57   0AF1'CF   DE  1CE0  3493          MOVAL   INTLKTMO_MSG,R7          ; No, print appropriate message
                       38      10  1CE5  3494          BSBB    200$
                                   1CE7  3495 110$:
                50   01D4 C6   D0  1CE7  3496          MOVL    MA_L_IADAWI(R6),R0       ; Did ADAWI counts add up?
                       07      13  1CEC  3497          BEQL    120$                     ; BR if so
                57   0B55'CF   DE  1CEE  3498          MOVAL   IADAWI_MSG,R7            ; No, print appropriate message
                       2A      10  1CF3  3499          BSBB    200$
                                   1CF5  3500 120$:
                50   01D8 C6   D0  1CF5  3501          MOVL    MA_L_IQUEUE(R6),R0       ; Were all interlock queues consistent?
                       07      13  1CFA  3502          BEQL    130$                     ; BR if so
                57   0BBE'CF   DE  1CFC  3503          MOVAL   IQUEUE_MSG,R7            ; No, print appropriate message
                       1C      10  1D01  3504          BSBB    200$
                                   1D03  3505 130$:
2C   00   00000000'EF   00   2C   1D03  3506          MOVC5   #0,0,#0,#MA_K_ERRLEN,-   ; Clear out this run's errors
                     01B0 C6        1D0C  3507                  MA_L_ERRBLK(R6)
```

```
                56    66   C0   1D0F   3508          ADDL2   (R6),R6                 ; Point to node for next memory
          56  000001D8'8F   D1   1D12   3509          CMPL    #UNIT_LIST,R6           ; Back at the start of the queue?
                      03   13   1D19   3510          BEQL    150$                    ; BR if we are - we're finished
                    FF35   31   1D1B   3511          BRW     10$                     ; More to go so loop
                           1D1E   3512 150$:
                      04   1D1E   3513          RET                                 ; Error summary finished printing
```

```
                        1D1F  3515 200$:
          F52B    30    1D1F  3516          BSBW    MASK_2_ARGLST                   ; Convert R0 mask to $FAO arg list
     58    5E    D0    1D22  3517          MOVL    SP,R8                           ; Save pointer to arg list
                        1D25  3518          $FAOL_S CTRSTR = (R7),-                 ; Format message with errant procs
                        1D25  3519                  OUTLEN = BUFFER_PTR,-
                        1D25  3520                  OUTBUF = FAO_BUF,-
                        1D25  3521                  PRMLST = (R8)
          000C'CF   DF  1D38  3522          PUSHAL  BUFFER_PTR
              01    DD  1D3C  3523          PUSHL   #1
       00741133 8F    DD  1D3E  3524          PUSHL   #UETP$_TEXT!STS$K_INFO
 00000000'GF    03    FB  1D44  3525          CALLS   #3,G^LIB$SIGNAL
58    04    68    04  7A  1D4B  3526          EMUL    #4,(R8),#4,R8                   ; Remove MASK_2_ARGLST results...
          5E    58    C0  1D50  3527          ADDL2   R8,SP                           ; ...from the stack
                    05  1D53  3528          RSB
```

```
                     1D54  3530          .SBTTL  Test Cleanup Routine
                     1D54  3531 ;++
                     1D54  3532 ; FUNCTIONAL DESCRIPTION:
                     1D54  3533 ;       This routine is called by the exit handler to ensure that all objects
                     1D54  3534 ;       which use system resources (e.g., global sections in shared memory)
                     1D54  3535 ;       are all properly disposed of and that we are consistent with respect to
                     1D54  3536 ;       other processors.
                     1D54  3537 ;
                     1D54  3538 ; CALLING SEQUENCE:
                     1D54  3539 ;       CALLS   #0,CLEANUP
                     1D54  3540 ;
                     1D54  3541 ; INPUT PARAMETERS:
                     1D54  3542 ;       NONE
                     1D54  3543 ;
                     1D54  3544 ; IMPLICIT INPUTS:
                     1D54  3545 ;       UNIT_LIST points to the list of nodes, one for each memory.  Within
                     1D54  3546 ;       node, there is a pointer to the interprocessor global section for that
                     1D54  3547 ;       memory as well as miscellaneous status info.
                     1D54  3548 ;
                     1D54  3549 ; OUTPUT PARAMETERS:
                     1D54  3550 ;       NONE
                     1D54  3551 ;
                     1D54  3552 ; IMPLICIT OUTPUTS:
                     1D54  3553 ;       NONE
                     1D54  3554 ;
                     1D54  3555 ; COMPLETION CODES:
                     1D54  3556 ;       NONE
                     1D54  3557 ;
                     1D54  3558 ; SIDE EFFECTS:
                     1D54  3559 ;       Remove as many traces as possible of our existence.
                     1D54  3560 ;
                     1D54  3561 ;--
                     1D54  3562
                     1D54  3563 CLEANUP:
0FFC                 1D54  3564          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     1D56  3565
                     1D56  3566 ;+
                     1D56  3567 ;       The following objects (using system resources) are created during the
                     1D56  3568 ;       execution of the shared memory test:  one global section per memory
                     1D56  3569 ;       accessed by all processors connected to that memory, one global section
                     1D56  3570 ;       per memory accessed only by the current processor, the files associated
                     1D56  3571 ;       with those global sections, one mailbox per memory associated with the
                     1D56  3572 ;       current processor, other mailboxes in each memory associated with
                     1D56  3573 ;       processors connected to that memory, one common event flag cluster
                     1D56  3574 ;       per memory accessed by all processors connected to that memory, and one
                     1D56  3575 ;       common event flag cluster per memory accessed only by the current
                     1D56  3576 ;       processor.  Of those items, only the global sections and the associated
                     1D56  3577 ;       files are not automatically deleted or marked for deletion during image
                     1D56  3578 ;       rundown; the others have a "temporary" existence.
                     1D56  3579 ;
                     1D56  3580 ;       The interprocessor global section may not have actually been created by
                     1D56  3581 ;       this processor.  In that case, we cannot delete it, but only delete its
                     1D56  3582 ;       address space.  The file that was created in our attempt to create the
                     1D56  3583 ;       section still must be deleted, however.
                     1D56  3584 ;
                     1D56  3585 ;       Even though we may be able to try to delete the interprocess global
                     1D56  3586 ;       section, it may only be marked for deletion because other processors
```

```
                                1D56  3587 ;       are still using it.  Because there are data structures within the
                                1D56  3588 ;       section and expectations on the part of other processors as to how we
                                1D56  3589 ;       will access those structures, we will attempt to keep things consistent
                                1D56  3590 ;       within the section and between processors regardless of whether or not
                                1D56  3591 ;       we can delete the section.
                                1D56  3592 ;-
                                1D56  3593
      56   01D8'CF    DE        1D56  3594          MOVAL   UNIT_LIST,R6             ; R6 will point to node for each memory
                                1D5B  3595 10$:
           56   66    C0        1D5B  3596          ADDL2   (R6),R6                 ; Point to next node on list
56   000001D8'8F    D1        1D5E  3597          CMPL    #UNIT_LIST,R6           ; Check now for back at start of queue...
             03    12        1D65  3598          BNEQ    20$                     ; ...and exit if empty, because we...
           01A4    31        1D67  3599          BRW     200$                    ; ...can be called before queue is set
                                1D6A  3600 20$:
                                1D6A  3601 ;
                                1D6A  3602 ; Before we can get rid of the interprocessor global section, we must do all
                                1D6A  3603 ; the things which will keep us consistent with any processors which may still
                                1D6A  3604 ; be using it.  This involves removing us from any masks in the global section
                                1D6A  3605 ; which say we are present and releasing any locks we may own.  Note that we
                                1D6A  3606 ; are ignoring any entries we may have on the free queue and the busy queue
                                1D6A  3607 ; and the count of such, and that we also ignore anything we may have done with
                                1D6A  3608 ; the ADAWI count in the interprocessor global section.  To try to fix these
                                1D6A  3609 ; would probably cause more trouble than leaving them, since there is an
                                1D6A  3610 ; excellent chance that they are consistent anyway.  When everything needed for
                                1D6A  3611 ; consistency is done, we have to tell any other processors that we're going
                                1D6A  3612 ; away.  We needn't bother cancelling timers for any other messages we may have
                                1D6A  3613 ; to those processors; that is one of the functions of image rundown.
                                1D6A  3614 ;
      57   062C C6    D0        1D6A  3615          MOVL    MA_Q_COMGSRAD(R6),R7    ; Point to start of interprocessor GS
             03    12        1D6F  3616          BNEQ    30$
           0126    31        1D71  3617          BRW     140$                    ; Skip all this is we haven't mapped it
                                1D74  3618 30$:
      58   01AC C6    9A        1D74  3619          MOVZBL  MA_B_MYPORT(R6),R8      ; Set up for bit clear instructions
   00 24 A7    58    E7        1D79  3620          BBCCI   R8,GS_L_AVAILABLE(R7),40$ ; We're no longer playing ball
                                1D7E  3621 40$:
   00 2C A7    58    E7        1D7E  3622          BBCCI   R8,GS_L_PARTICIPATING(R7),50$ ; We're not interlocking...
                                1D83  3623 50$:
   00 30 A7    58    E7        1D83  3624          BBCCI   R8,GS_L_FINISHED(R7),60$      ; ...
                                1D88  3625 60$:
         01AC C6    91        1D88  3626          CMPB    MA_B_MYPORT(R6),-       ; Do I own the lock?
            2B A7              1D8C  3627                  GS_L_CEF_FLAG+3(R7)
               03    12        1D8E  3628          BNEQ    70$                     ; BR if not - it will get cleared
            28 A7    D4        1D90  3629          CLRL    GS_L_CEF_FLAG(R7)       ; Unlock interprocessor CEF test
                                1D93  3630 70$:
         01AC C6    91        1D93  3631          CMPB    MA_B_MYPORT(R6),-       ; Do I own the lock?
            37 A7              1D97  3632                  GS_L_INTLK_FLAG+3(R7)
               03    12        1D99  3633          BNEQ    80$                     ; BR if not - it will get cleared
            34 A7    D4        1D9B  3634          CLRL    GS_L_INTLK_FLAG(R7)     ; Unlock memory interlocking test
                                1D9E  3635 80$:
                                1D9E  3636
               58    D4        1D9E  3637          CLRL    R8                      ; R8 is STRTPOS for FFS operations
         59    04    D0        1DA0  3638          MOVL    #PROCESSOR_MAX,R9       ; R9 is SIZE for FFS operations
                                1DA3  3639 90$:
5A   24 A7    59    58    EA  1DA3  3640          FFS     R8,R9,GS_L_AVAILABLE(R7),R10 ; Search for a willing processor
               03    12        1DA9  3641          BNEQ    100$                    ; BR if we found one
            0090    31        1DAB  3642          BRW     110$                    ; BR if there are none left
                                1DAE  3643 100$:
```

```
                01AC C6    90  1DAE  3644          MOVB    MA_B_MYPORT(R6),-            ; Start filling in message...
                110A'CF            1DB2  3645                  AST_MBBUF+MB_B_SENDER        ; ...sender's port number...
          110B'CF    5A    90  1DB5  3646          MOVB    R10,AST_MBBUF+MB_B_RECEIVER ; ...receiver's port number...
                01EC C6    90  1DBA  3647          MOVB    MA_W_UNIT(R6),-              ; ...unit of the memory we share...
                110C'CF            1DBE  3648                  AST_MBBUF+MB_B_UNIT
                      0D   B0  1DC1  3649          MOVW    #MSG_BYEBYE,-                ; ...reason for the message
                110D'CF            1DC3  3650                  AST_MBBUF+MB_W_WHY
                0258 C6    28  1DC6  3651          MOVC3   MA_Q_COMMBNAM(R6),-          ; Form the logical name...
      113D'CF   0260 C6            1DCA  3652                  MA_T_COMMBNAM(R6),AST_MBNAM ; ...of other processor's MB
1135'CF 1139'CF 0258 C6    D0  1DD0  3653          MOVL    MA_Q_COMMBNAM(R6),AST_MBPTR ; Form descriptor to...
      113D'CF   50   5A    DE  1DD7  3654          MOVAL   AST_MBNAM,AST_MBPTR+4        ; ...the logical name
                50   5A    D0  1DDE  3655          MOVL    R10,R0                       ; Include the port number...
                   F44D    30  1DE1  3656          BSBW    HEXUNIT                      ; ...
              FF A3    50    90  1DE4  3657          MOVB    R0,-1(R3)                    ; ...in the name
                                 1DE8  3658          $CREMBX_S CHAN = AST_MBCHN,-         ; Establish a link to other proc's MB
                                 1DE8  3659                  LOGNAM = AST_MBPTR
                                 1DFF  3660          $QIO_S  CHAN = AST_MBCHN,-           ; Say bye to the other processor
                                 1DFF  3661                  FUNC = #IO$_WRITEVBLK!IO$M_NOW,-
                                 1DFF  3662                  IOSB = AST_MBISB,-           ; NOTE: We ignore errors!
                                 1DFF  3663                  P1   = AST_MBBUF,-
                                 1DFF  3664                  P2   = #MB_K_END
                                 1E22  3665          $DASSGN_S CHAN = SYNCH_MBCHN         ; We no longer need the MB
          5B    5A    58    C3  1E2E  3666          SUBL3   R8,R10,R11                   ; Get number of bits we searched
                      5B    D6  1E32  3667          INCL    R11                          ; Correct off-by-one (inclusive search)
                5A    5B    C2  1E34  3668          SUBL2   R11,R9                       ; Adjust number of bits left to search
          58    5A    01    C1  1E37  3669          ADDL3   #1,R10,R8                    ; Point to new first bit to search
                   FF65    31  1E3B  3670          BRW     90$                          ; Loop thru all procs on this memory
                                 1E3E  3671 110$:
                                 1E3E  3672 ;
                                 1E3E  3673 ; Only update or delete the interprocessor global section if we were the
                                 1E3E  3674 ; processor which created it.  Delete the file regardless.
                                 1E3E  3675 ;
            58    20 A7    90  1E3E  3676          MOVB    GS_L_OWNER(R7),R8            ; Save a copy of the GS creator's port
      01AC C6     58    91  1E42  3677          CMPB    R8,MA_B_MYPORT(R6)           ; Did we create the section?
                20    12  1E47  3678          BNEQ    120$                         ; BR if not - can't update it
                                 1E49  3679          $UPDSEC_S INADR = MA_Q_COMGSRAD(R6),- ; Updating the section now...
                                 1E49  3680                  EFN   = #COMGS_EFN           ; ...means we avoid implicit update
                                 1E60  3681          $WAITFR_S EFN = #COMGS_EFN   ; Let the $UPDSEC finish in peace
                                 1E69  3682 120$:
                                 1E69  3683          $DELTVA_S INADR = MA_Q_COMGSRAD(R6) ; Remove GS virtual address space
      01AC C6     58    91  1E78  3684          CMPB    R8,MA_B_MYPORT(R6)           ; Did we create the section?
                0F    12  1E7D  3685          BNEQ    130$                         ; BR if not - can't delete it
                                 1E7F  3686          $DGBLSC_S GSDNAM = MA_Q_COMGSNAM(R6) ; Delete the section...
                                 1E8E  3687 130$:
                                 1E8E  3688          $DASSGN_S CHAN = MA_K_COMFAB+FAB$L_STV(R6) ; ...and unhook RMS
                                 1E9A  3689 140$:
          0112 C6    B5  1E9A  3690          TSTW    MA_K_COMFAB+FAB$W_IFI(R6) ; Is the file still open?
                0B    13  1E9E  3691          BEQL    150$                         ; BR if not
                                 1EA0  3692          $CLOSE  FAB = MA_K_COMFAB(R6)   ; Close it if it is
                                 1EAB  3693 150$:
                                 1EAB  3694          $ERASE  FAB = MA_K_COMFAB(R6)   ; Get rid of the associated file
                                 1EB6  3695                                       ; (We can swallow RMS-W-IFI errors)
                                 1EB6  3696 ;
                                 1EB6  3697 ; We can just blindly get rid of the single processor global section because no
                                 1EB6  3698 ; one else knows of it.  We must, however, wait to catch the situation where we
                                 1EB6  3699 ; reach this point but an $UPDSEC is still in progress.  See the note where
                                 1EB6  3700 ; SPRGS_EFN is defined.  If the section is already removed, there should be
```

```
                     1EB6  3701 ; no problem.
                     1EB6  3702 ;
063C C6   D5         1EB6  3703          TSTL    MA_Q_SPRGSRAD(R6)               ; Did we ever map the section?
      33   13        1EBA  3704          BEQL    160$                            ; BR if not
                     1EBC  3705          $WAITFR_S EFN = #SPRGS_EFN              ; Prevent prcb if $UPDSEC in progress
                     1EC5  3706          $DELTVA_S INADR = MA_Q_SPRGSRAD(R6) ; Remove virtual address space...
                     1ED4  3707          $DGBLSC_S GSDNAM = MA_Q_SPRGSNAM(R6) ; ...delete the section...
                     1EE3  3708          $DASSGN_S CHAN = MA_K_SPRFAB+FAB$L_STV(R6) ; ...and unhook RMS
                     1EEF  3709 160$:
03FE C6   B5         1EEF  3710          TSTW    MA_K_SPRFAB+FAB$W_IFI(R6) ; Is the file still open?
      0B   13        1EF3  3711          BEQL    170$                            ; BR if not
                     1EF5  3712          $CLOSE  FAB = MA_K_SPRFAB(R6)           ; Close it if it is
                     1F00  3713 170$:
                     1F00  3714          $ERASE  FAB = MA_K_SPRFAB(R6)           ; Get rid of the associated file
                     1F0B  3715                                                  ; (We can swallow RMS-W-IFI errors)
                     1F0B  3716
   FE4D   31         1F0B  3717          BRW     10$                             ; Loop for the next memory
                     1F0E  3718 200$:
          04         1F0E  3719          RET
```

```
                     1F0F  3721          .SBTTL  Test Timer Expiration Routine
                     1F0F  3722 ;++
                     1F0F  3723 ; FUNCTIONAL DESCRIPTION:
                     1F0F  3724 ;       This routine runs when the timer to mark the end of the test goes off.
                     1F0F  3725 ;
                     1F0F  3726 ; CALLING SEQUENCE:
                     1F0F  3727 ;       Called via AST at $SETIMR expiration.
                     1F0F  3728 ;
                     1F0F  3729 ; INPUT PARAMETERS:
                     1F0F  3730 ;       NONE
                     1F0F  3731 ;
                     1F0F  3732 ; IMPLICIT INPUTS:
                     1F0F  3733 ;       NONE
                     1F0F  3734 ;
                     1F0F  3735 ; OUTPUT PARAMETERS:
                     1F0F  3736 ;       NONE
                     1F0F  3737 ;
                     1F0F  3738 ; IMPLICIT OUTPUTS:
                     1F0F  3739 ;       NONE
                     1F0F  3740 ;
                     1F0F  3741 ; COMPLETION CODES:
                     1F0F  3742 ;       NONE
                     1F0F  3743 ;
                     1F0F  3744 ; SIDE EFFECTS:
                     1F0F  3745 ;       Sets a flag to indicate timer expiration.
                     1F0F  3746 ;
                     1F0F  3747 ;--
                     1F0F  3748
                     1F0F  3749 TIME_OUT:
                OFFC 1F0F  3750          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     1F11  3751
0002'CF    02    A8 1F11  3752          BISW2   #TEST_OVERM,FLAG        ; Indicate end of pass or of test
                   04 1F16  3753          RET
```

```
1F17  3755          .SBTTL  System Service Exception Handler
1F17  3756 ;++
1F17  3757 ; FUNCTIONAL DESCRIPTION:
1F17  3758 ;       This routine is executed if a software or hardware exception occurs or
1F17  3759 ;       if a LIB$SIGNAL system service is used to output a message.
1F17  3760 ;
1F17  3761 ; CALLING SEQUENCE:
1F17  3762 ;       Entered via an exception from the system
1F17  3763 ;
1F17  3764 ; INPUT PARAMETERS:
1F17  3765 ;       ERROR_COUNT    = previous cumulative error count
1F17  3766 ;
1F17  3767 ;          AP ---->     +---------------+
1F17  3768 ;                       |       2       |
1F17  3769 ;                       +---------------+
1F17  3770 ;                       |  SIGNL ARY PNT|
1F17  3771 ;                       +---------------+
1F17  3772 ;                       |  MECH  ARY PNT|  ---------
1F17  3773 ;                       +---------------+         ^
1F17  3774 ;                       |       4       |         |
1F17  3775 ;                       +---------------+         |
1F17  3776 ;                       |  ESTABLISH FP |         |
1F17  3777 ;                       +---------------+         |
1F17  3778 ;                       |     DEPTH     |  Mechanism Array
1F17  3779 ;                       +---------------+         |
1F17  3780 ;                       |       R0      |         |
1F17  3781 ;                       +---------------+         v
1F17  3782 ;                       |       R1      |  ---------
1F17  3783 ;                       +---------------+         ^
1F17  3784 ;                       |       N       |         |
1F17  3785 ;                       +---------------+         |
1F17  3786 ;                       |  CONDITION NAME|        |
1F17  3787 ;                       +---------------+         |
1F17  3788 ;                       | N-3 ADDITIONAL|    Signal Array
1F17  3789 ;                       | LONG WORD ARGS|         |
1F17  3790 ;                       +---------------+         |
1F17  3791 ;                       |       PC      |         |
1F17  3792 ;                       +---------------+         |
1F17  3793 ;                       |       PSL     |         v
1F17  3794 ;                       +---------------+  ---------
1F17  3795 ; IMPLICIT INPUTS:
1F17  3796 ;       NONE
1F17  3797 ;
1F17  3798 ; OUTPUT PARAMETERS:
1F17  3799 ;       NONE
1F17  3800 ;
1F17  3801 ; IMPLICIT OUTPUTS:
1F17  3802 ;       NONE
1F17  3803 ;
1F17  3804 ; COMPLETION CODES:
1F17  3805 ;       SS$_NORMAL if it's a UETP condition or RMS error.
1F17  3806 ;       Error status from exception, otherwise.
1F17  3807 ;
1F17  3808 ; SIDE EFFECTS:
1F17  3809 ;       May branch to ERROR_EXIT.
1F17  3810 ;       May print a message.
1F17  3810 ;--
1F17  3811
```

```
                                1F17  3812 SSERROR:
                           0FFC 1F17  3813          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                1F19  3814
                                1F19  3815          $SETAST_S ENBFLG = #0           ; Disable AST delivery
                             01 DD   1F22  3816          PUSHL   #1                  ; Assume ASTs were enabled
                          50 09 D1   1F24  3817          CMPL    S^#SS$_WASSET,R0    ; Were ASTs enabled?
                             02 13   1F27  3818          BEQL    10$                 ; BR if they were
                             6E D4   1F29  3819          CLRL    (SP)                ; Set ASTs to remain disabled
                                1F2B  3820 10$:
                                1F2B  3821          $SETSFM_S ENBFLG = #0           ; Disable SS failure mode
                             01 DD   1F34  3822          PUSHL   #1                  ; Assume SS failure mode was enabled
                          50 09 D1   1F36  3823          CMPL    S^#SS$_WASSET,R0    ; Was SS failure mode enabled?
                             02 13   1F39  3824          BEQL    20$                 ; BR if it was
                             6E D4   1F3B  3825          CLRL    (SP)                ; Set SS failure mode to remain off
                                1F3D  3826 20$:
                       56 04 AC DO   1F3D  3827          MOVL    CHF$L_SIGARGLST(AP),R6  ; Get the signal array pointer
                       59 04 A6 7D   1F41  3828          MOVQ    CHF$L_SIG_NAME(R6),R9   ; Get NAME in R9 and ARG1 in R10
                             10 ED   1F45  3829          CMPZV   #STS$V_FAC_NO,-         ; Is this a message from LIB$SIGNAL?
                                0C   1F47  3830                  #STS$S_FAC_NO,-
                00000074 8F     59   1F48  3831                  R9,#UETP$_FACILITY
                             14 12   1F4E  3832          BNEQ    30$                 ; BR if this is not a UETP exception
                          66 02 C2   1F50  3833          SUBL2   #2,CHF$L_SIG_ARGS(R6)   ; Drop the PC and PSL
                                1F53  3834          $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6) ; Print the message
                             21 11   1F62  3835          BRB     40$                 ; Restore ASTs and SS fail mode
                                1F64  3836 30$:
                59 0000045C 8F   D1   1F64  3837          CMPL    #SS$_SSFAIL,R9      ; RMS failures are SysSvc failures
                             32 12   1F6B  3838          BNEQ    50$                 ; BR if this can't be an RMS failure
                             10 ED   1F6D  3839          CMPZV   #STS$V_FAC_NO,-     ; Is it an RMS failure?
                                0C   1F6F  3840                  #STS$S_FAC_NO,-
                          01    5A   1F70  3841                  R10,#RMS$_FACILITY
                             2B 12   1F72  3842          BNEQ    50$                 ; BR if not
                5A    F0000000 8F   CA   1F74  3843          BICL2   #^XF0000000,R10     ; Strip control bits from status code
                       08 A6    04   39   1F7B  3844          MATCHC  #4,CHF$L_SIG_ARG1(R6),- ; Is it an RMS failure for which...
                             14        1F7F  3845                  #NRAT_LENGTH,-
                          004D'CF      1F80  3846                  NO_RMS_AST_TABLE    ; ...no AST can be delivered?
                             1A 13   1F83  3847          BEQL    50$                 ; BR if so - must give error here
                                1F85  3848 40$:
                             01 BA   1F85  3849          POPR    #^M<R0>             ; Restore SS failure mode...
                                1F87  3850          $SETSFM_S ENBFLG = R0           ; ...
                             01 BA   1F90  3851          POPR    #^M<R0>             ; Restore AST enable...
                                1F92  3852          $SETAST_S ENBFLG = R0           ; ...
                          50 01 D0   1F9B  3853          MOVL    S^#SS$_NORMAL,R0    ; Supply a standard status for exit
                             04 1F9E  3854          RET                             ; Resume processing (or goto RMS_ERROR)
                                1F9F  3855 50$:
                    018A'CF  59   D0   1F9F  3856          MOVL    R9,STATUS           ; Save the status
                             58   D4   1FA4  3857          CLRL    R8                  ; Assume for now it's not SS failure
                59 0000045C  8F   D1   1FA6  3858          CMPL    #SS$_SSFAIL,R9      ; But is it a System Service failure?
                             38 12   1FAD  3859          BNEQ    70$                 ; BR if not - no special case message
                                1FAF  3860          $GETMSG_S MSGID = R10,-         ; Get SS failure code associated text
                                1FAF  3861                    MSGLEN = BUFFER_PTR,-
                                1FAF  3862                    BUFADR = FAO_BUF,-
                                1FAF  3863                    FLAGS  = #14,-
                                1FAF  3864                    OUTADR = MSG_BLOCK
                          01BB'CF    95   1FC6  3865          TSTB    MSG_BLOCK+1     ; Get FAO arg count for SS failure code
                             16 13   1FCA  3866          BEQL    60$                 ; Don't use $GETMSG if no $FAO args...
                          000C'CF    DF   1FCC  3867          PUSHAL  BUFFER_PTR          ; ...else build up...
                             01 DD   1FD0  3868          PUSHL   #1                  ; ...a message describing...
```

```
00741130 8F    DD   1FD2  3869          PUSHL   #UETP$_TEXT                     ; ...why the System Service failed
      00 5A    F0   1FD8  3870          INSV    R10,#STS$V_SEVERITY,-           ; Give the message...
      6E 03         1FDB  3871                  #STS$S_SEVERITY,(SP)            ; ...the correct severity code
      58 03    D0   1FDD  3872          MOVL    #3,R8                           ; Count the number of args we pushed
         05    11   1FE0  3873          BRB     70$
                    1FE2  3874 60$:
         5A    DD   1FE2  3875          PUSHL   R10                             ; Save SS failure code
      58 01    D0   1FE4  3876          MOVL    #1,R8                           ; Count the number of args we pushed
                    1FE7  3877 70$:
   57 66 04    C5   1FE7  3878          MULL3   #4,CHF$L_SIG_ARGS(R6),R7 ; Convert longwords to bytes
      5E 57    C2   1FEB  3879          SUBL2   R7,SP                           ; Save the current signal array...
6E 04 A6 57    28   1FEE  3880          MOVC3   R7,CHF$L_SIG_NAME(R6),(SP) ; ...on the stack
   7E 66 58    C1   1FF3  3881          ADDL3   R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
         00A6  31   1FF7  3882          BRW     ERROR_EXIT
```

```
                        1FFA  3884          .SBTTL  RMS Error Handler
                        1FFA  3885 ;++
                        1FFA  3886 ; FUNCTIONAL DESCRIPTION:
                        1FFA  3887 ;       This routine handles error returns from RMS calls.
                        1FFA  3888 ;
                        1FFA  3889 ; CALLING SEQUENCE:
                        1FFA  3890 ;       Called by RMS when a file processing error is found.
                        1FFA  3891 ;
                        1FFA  3892 ; INPUT PARAMETERS:
                        1FFA  3893 ;       Address of the FAB or RAB associated with the RMS call.
                        1FFA  3894 ;
                        1FFA  3895 ; IMPLICIT INPUTS:
                        1FFA  3896 ;       NONE
                        1FFA  3897 ;
                        1FFA  3898 ; OUTPUT PARAMETERS:
                        1FFA  3899 ;       NONE
                        1FFA  3900 ;
                        1FFA  3901 ; IMPLICIT OUTPUTS:
                        1FFA  3902 ;       Error message
                        1FFA  3903 ;
                        1FFA  3904 ; COMPLETION CODES:
                        1FFA  3905 ;       NONE
                        1FFA  3906 ;
                        1FFA  3907 ; SIDE EFFECTS:
                        1FFA  3908 ;       Program may exit, depending on severity of the error.
                        1FFA  3909 ;
                        1FFA  3910 ;--
                        1FFA  3911
                        1FFA  3912 RMS_ERROR:
                  0FFC  1FFA  3913          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        1FFC  3914
          56   04 AC   D0  1FFC  3915          MOVL    4(AP),R6                ; See whether we're dealing with...
             66   03   91  2000  3916          CMPB    #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
                  16   12  2003  3917          BNEQ    10$                     ; BR if it's a RAB
       57   01D4'CF   DE  2005  3918          MOVAL   FILE,R7                 ; FAB-specific code:  text string...
             58   56   D0  200A  3919          MOVL    R6,R8                   ; ...address of FAB...
                0C A6   DD  200D  3920          PUSHL   FAB$L_STV(R6)           ; ...STV field for error...
                08 A6   DD  2010  3921          PUSHL   FAB$L_STS(R6)           ; ...STS field for error...
018A'CF         08 A6   D0  2013  3922          MOVL    FAB$L_STS(R6),STATUS    ; ...and save the error code
                  15   11  2019  3923          BRB     COMMON                  ; FAB and RAB share other code
                        201B  3924 10$:
       57   01E0'CF   DE  201B  3925          MOVAL   RECORD,R7               ; RAB-specific code:  text string...
             58   3C A6   D0  2020  3926          MOVL    RAB$L_FAB(R6),R8        ; ...address of associated FAB...
                0C A6   DD  2024  3927          PUSHL   RAB$L_STV(R6)           ; ...STV field for error...
                08 A6   DD  2027  3928          PUSHL   RAB$L_STS(R6)           ; ...STS field for error...
018A'CF         08 A6   D0  202A  3929          MOVL    RAB$L_STS(R6),STATUS    ; ...and save the error code
                        2030  3930 COMMON:
          5A   34 A8   9A  2030  3931          MOVZBL  FAB$B_FNS(R8),R10       ; Get the file name size
                        2034  3932          $FAO_S  CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
                        2034  3933                  OUTLEN = BUFFER_PTR,-
                        2034  3934                  OUTBUF = FAO_BUF,-
                        2034  3935                  P1     = R7,-
                        2034  3936                  P2     = R10,-
                        2034  3937                  P3     = FAB$L_FNA(R8)
             000C'CF    DF  204E  3938          PUSHAL  BUFFER_PTR              ; ...and arguments for ERROR_EXIT...
                   01   DD  2052  3939          PUSHL   #1                      ; ...
          00741130   8F   DD  2054  3940          PUSHL   #UETP$_TEXT             ; ...
```

```
                00   EF  205A  3941          EXTZV   #STS$V_SEVERITY,-
                03       205C  3942                  #STS$S_SEVERITY,-
          59    018A'CF  205D  3943                  STATUS,R9                   ; ...get the severity code...
                6E 59    88    2061  3944    BISB2   R9,(SP)                     ; ...and add it into the signal name
                   05    DD    2064  3945    PUSHL   #5                          ; Current arg count
                   0037  31    2066  3946    BRW     ERROR_EXIT
```

```
                                2069  3948          .SBTTL  CTRL/C Handler
                                2069  3949 ;++
                                2069  3950 ; FUNCTIONAL DESCRIPTION:
                                2069  3951 ;       This routine handles CTRL/C AST's
                                2069  3952 ;
                                2069  3953 ; CALLING SEQUENCE:
                                2069  3954 ;       Called via AST
                                2069  3955 ;
                                2069  3956 ; INPUT PARAMETERS:
                                2069  3957 ;       NONE
                                2069  3958 ;
                                2069  3959 ; IMPLICIT INPUTS:
                                2069  3960 ;       NONE
                                2069  3961 ;
                                2069  3962 ; OUTPUT PARAMETERS:
                                2069  3963 ;       NONE
                                2069  3964 ;
                                2069  3965 ; IMPLICIT OUTPUTS:
                                2069  3966 ;       NONE
                                2069  3967 ;
                                2069  3968 ; COMPLETION CODES:
                                2069  3969 ;       NONE
                                2069  3970 ;
                                2069  3971 ; SIDE EFFECTS:
                                2069  3972 ;       NONE
                                2069  3973 ;
                                2069  3974 ;--
                                2069  3975
                                2069  3976 CCASTHAND:
                        0FFC    2069  3977          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                2068  3978
                0082'CF   DF    2068  3979          PUSHAL  CNTRLCMSG                   ; Set message pointer
                          01    206B  3980          PUSHL   #1                          ; Set arg count
                00741130  8F    206D  3981          PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
                          00    2071  3982          PUSHL   #0                          ; Indicate an abnormal termination
                00E4'CF   DF    2077  3983          PUSHAL  PROCESS_NAME                ; ...
                          02    2079  3984          PUSHL   #2                          ; ...
                007410E0  8F    207D  3985          PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
      00000000'GF       07    FB    207F  3986          CALLS   #7,G^LIB$SIGNAL             ; Output the message
                              D0    2085  3987          MOVL    #<STS$M_INHIB_MSG!-         ; Set the exit status
                                    208C  3988                  SS$_CONTROLC-
                                    208D  3989                  STS$K_SUCCESS+STS$K_WARNING>,-
018A'CF   10000650  8F              208D  3990                  STATUS
                                    2095  3991          $EXIT_S STATUS                      ; Terminate program cleanly
```

```
                        20A0  3993          .SBTTL  Error Exit
                        20A0  3994 ;++
                        20A0  3995 ; FUNCTIONAL DESCRIPTION:
                        20A0  3996 ;       This routine prints an error message and exits.
                        20A0  3997 ;
                        20A0  3998 ; CALLING SEQUENCE:
                        20A0  3999 ;       MOVx  error status value,STATUS
                        20A0  4000 ;       PUSHx error specific information on the stack
                        20A0  4001 ;       PUSHL current argument count
                        20A0  4002 ;       BRW   ERROR_EXIT
                        20A0  4003 ;
                        20A0  4004 ; INPUT PARAMETERS:
                        20A0  4005 ;       Arguments to LIB$SIGNAL, as above
                        20A0  4006 ;
                        20A0  4007 ; IMPLICIT INPUTS:
                        20A0  4008 ;       NONE
                        20A0  4009 ;
                        20A0  4010 ; OUTPUT PARAMETERS:
                        20A0  4011 ;       Message to SYS$OUTPUT and SYS$ERROR
                        20A0  4012 ;
                        20A0  4013 ; IMPLICIT OUTPUTS:
                        20A0  4014 ;       Program exit
                        20A0  4015 ;
                        20A0  4016 ; COMPLETION CODES:
                        20A0  4017 ;       NONE
                        20A0  4018 ;
                        20A0  4019 ; SIDE EFFECTS:
                        20A0  4020 ;       NONE
                        20A0  4021 ;
                        20A0  4022 ;--
                        20A0  4023
                        20A0  4024 ERROR_EXIT:
                        20A0  4025
                        20A0  4026          $SETAST_S ENBFLG = #0              ; ASTs can play havoc with messages
    15 0002'CF    03    E0  20A9  4027          BBS     #BEGIN_MSGV,FLAG,10$      ; BR if "begin" msg already printed
                  7E    D4  20AF  4028          CLRL    -(SP)                     ; Set the time stamp flag
            000F'CF     DF  20B1  4029          PUSHAL  TEST_NAME                 ; Set the test name
                  02    DD  20B5  4030          PUSHL   #2                        ; Push the argument count
        00741039  8F    DD  20B7  4031          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
 00000000'GF    04    FB  20BD  4032          CALLS   #4,G^LIB$SIGNAL           ; Print the startup message
                        20C4  4033 10$:
01CE'CF     08    8E    C1  20C4  4034          ADDL3   (SP)+,#8,ARG_COUNT        ; Get total # args, pop partial count
            0186'CF     D6  20CA  4035          INCL    ERROR_COUNT               ; Keep running error count
                  00    DD  20CE  4036          PUSHL   #0                        ; Push the time parameter
            00E4'CF     DF  20D0  4037          PUSHAL  PROCESS_NAME              ; Push test name...
        000F0002  8F    DD  20D4  4038          PUSHL   #^XF0002                  ; ...arg count...
        007410E2  8F    DD  20DA  4039          PUSHL   #UETP$_ABENDD!STS$K_ERROR ; ...and signal name
            0186'CF     DD  20E0  4040          PUSHL   ERROR_COUNT               ; Finish off arg list...
            00E4'CF     DF  20E4  4041          PUSHAL  PROCESS_NAME              ; ...
        00010002  8F    DD  20E8  4042          PUSHL   #^X10002                  ; ...
        00748022  8F    DD  20EE  4043          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
00000000'GF    01CE'CF  FB  20F4  4044          CALLS   ARG_COUNT,G^LIB$SIGNAL    ; Truly bitch
```

```
                FB45 CF    00    FB  20FD  4046          CALLS   #0,ERROR_SUMMARY          ; Print recoverable error statistics
          018A'CF              D5  2102  4047          TSTL    STATUS                   ; Did we exit with an error code?
                09             12  2106  4048          BNEQ    20$                      ; BR if we did
           007410E2 8F         D0  2108  4049          MOVL    #UETP$_ABENDD!STS$K_ERROR,- ; Supply a generic one otherwise
          018A'CF                  210E  4050                  STATUS
                               2111  4051 20$:
      018A'CF    10000000 8F    C8  2111  4052          BISL    #STS$M_INHIB_MSG,STATUS ; Don't print messages twice!
                                   211A  4053          $EXIT_S STATUS                   ; Exit in error
```

```
                                2125  4055          .SBTTL  Exit Handler
                                2125  4056 ;++
                                2125  4057 ; FUNCTIONAL DESCRIPTION:
                                2125  4058 ;       This routine handles cleanup at exit.  If the MODE logical name is
                                2125  4059 ;       equated to "ONE", the routine will update the test flag in the
                                2125  4060 ;       UETINIDEV.DAT file depending on the UETUNT$M_TESTABLE flag state in the
                                2125  4061 ;       UETUNT$B_FLAGS field of the unit block for each unit for the device
                                2125  4062 ;       under test.
                                2125  4063 ;
                                2125  4064 ; CALLING SEQUENCE:
                                2125  4065 ;       Invoked automatically by $EXIT System Service.
                                2125  4066 ;
                                2125  4067 ; INPUT PARAMETERS:
                                2125  4068 ;       STATUS  contains the exit status.
                                2125  4069 ;       FLAG    has synchronizing bits.
                                2125  4070 ;       DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV.
                                2125  4071 ;
                                2125  4072 ; IMPLICIT INPUTS:
                                2125  4073 ;       UNIT_LIST points to the head of a doubly linked circular list of unit
                                2125  4074 ;                 blocks for the device under test.
                                2125  4075 ;
                                2125  4076 ; OUTPUT PARAMETERS:
                                2125  4077 ;       NONE
                                2125  4078 ;
                                2125  4079 ; IMPLICIT OUTPUTS:
                                2125  4080 ;       Various files are de-accessed, the process name is reset, and any
                                2125  4081 ;       necessary synchronization with UETPDEV01 is carried out.
                                2125  4082 ;       If the MODE logical name is equated to "ONE", the routine will update
                                2125  4083 ;       the test flag in the UETINIDEV.DAT file depending on the
                                2125  4084 ;       UETUNT$M_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
                                2125  4085 ;       block for each unit for the device under test.
                                2125  4086 ;
                                2125  4087 ; COMPLETION CODES:
                                2125  4088 ;       NONE
                                2125  4089 ;
                                2125  4090 ; SIDE EFFECTS:
                                2125  4091 ;       NONE
                                2125  4092 ;
                                2125  4093 ;--
                                2125  4094
                                2125  4095 EXIT_HANDLER:
                          0FFC  2125  4096          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                2127  4097
                                2127  4098          $SETSFM_S ENBFLG = #0                   ; Turn off System Service failure mode
                                2130  4099          $SETAST_S ENBFLG = #0                   ; We're finished - no more ASTs
                03 0002'CF    04  E0  2139  4100          BBS     #ONE_SHOTV,FLAG,10$     ; If one-shot, update testability...
                          00B8  31  213F  4101          BRW     END_UPDATE              ; ...else don't update UETINIDEV.DAT
                                2142  4102 10$:
                03 0002'CF    02  E0  2142  4103          BBS     #SAFE_TO_UPDV,FLAG,20$  ; Only update if it's safe
                          00AF  31  2148  4104          BRW     END_UPDATE              ; Else forget it
                                214B  4105 20$:
                   5A   1250'CF   DE  214B  4106          MOVAL   INI_RAB,R10             ; Set the RAB address
                     1E AA    02  90  2150  4107          MOVB    #RAB$C_RFA,RAB$B_RAC(R10) ; Set RFA mode
             10 AA   1294'CF    06  28  2154  4108          MOVC3   #6,DDB_RFA,RAB$W_RFA(R10) ; Set RFA to DDB line
                                215B  4109          $GET    RAB = (R10)             ; Go back to the DDB record
                        75 50   E9  2164  4110          BLBC    R0,UPDATE_FAILED        ; If failure then forget it
                     1E AA    00  90  2167  4111          MOVB    #RAB$C_SEQ,RAB$B_RAC(R10) ; Set back to sequential mode
```

```
                                2177  4114 UNIT_LOOP:
                          01    E1  2177  4115          BBC     #UETUNT$V_TESTABLE,-    ; BR if this unit is not testable
                    02 0B AB          2179  4116                  UETUNT$B_FLAGS(R11),10$
                          59    D6  217C  4117          INCL    R9                      ; Count testable units
                                217E  4118 10$:
                       5B 6B    C0  217E  4119          ADDL2   (R11),R11               ; Next unit block
                000001D8'8F 5B    D1  2181  4120          CMPL    R11,#UNIT_LIST          ; Are we full circle in the list?
                          ED    12  2188  4121          BNEQ    UNIT_LOOP               ; BR if not
                          59    D5  218A  4122          TSTL    R9                      ; Any testable units?
                          12    12  218C  4123          BNEQ    20$                     ; BR if yes...
                 0018'CF  4E 8F 90  218E  4124          MOVB    #^A/N/,BUFFER+4         ; ...else disable the DDB record...
                                2194  4125          $UPDATE RAB = (R10)             ; ...here
                       3C 50    E9  219D  4126          BLBC    R0,UPDATE_FAILED        ; If error then forget it
                                21A0  4127 20$:
                       5B 6B    C0  21A0  4128          ADDL2   (R11),R11               ; Next unit block
                000001D8'8F 5B    D1  21A3  4129          CMPL    R11,#UNIT_LIST          ; Are we full circle in the list?
                          4E    13  21AA  4130          BEQL    END_UPDATE              ; BR if yes
                                21AC  4131          $GET    RAB = (R10)             ; Get a record
                       24 50    E9  21B5  4132          BLBC    R0,UPDATE_FAILED        ; If error then forget it
                    0014'CF 20 8A 21B8  4133          BICB2   #LC_BITM,BUFFER         ; Convert to uppercase
                 0014'CF  55 8F 91  21BD  4134          CMPB    #^A/U/,BUFFER           ; Is it a UCB record?
                          35    12  21C3  4135          BNEQ    END_UPDATE              ; BR if not
                          01    E0  21C5  4136          BBS     #UETUNT$V_TESTABLE,-    ; BR if this unit is testable...
                    D6 0B AB          21C7  4137                  UETUNT$B_FLAGS(R11),20$
                 0018'CF  4E 8F 90  21CA  4138          MOVB    #^A/N/,BUFFER+4         ; ...else disable the UCB record...
                                21D0  4139          $UPDATE RAB = (R10)             ; ...here
                       C4 50    E8  21D9  4140          BLBS    R0,20$                  ; Look at the next record if no error
                                21DC  4141 UPDATE_FAILED:
                       0C AA    DD  21DC  4142          PUSHL   RAB$L_STV(R10)          ; Do a simple message...
                          50    DD  21DF  4143          PUSHL   R0                      ; ...to tell of the failure
                    0197'CF     DF  21E1  4144          PUSHAL  INIDEV_UPDERR
                          01    DD  21E5  4145          PUSHL   #1
                          00    EF  21E7  4146          EXTZV   #STS$V_SEVERITY,-       ; Copy the severity from RMS status...
                 7E    50     03        21E9  4147                  #STS$S_SEVERITY,R0,-(SP)
            6E    00741130 8F CB  21EC  4148          BISL2   #UETP$_TEXT,(SP)        ; ...to our message
            00000000'GF    05 FB  21F3  4149          CALLS   #5,G^LIB$SIGNAL
                                21FA  4150 END_UPDATE:
                FB55 CF    00 FB  21FA  4151          CALLS   #0,CLEANUP              ; Get rid of test specific leftovers
                          00    DD  21FF  4152          PUSHL   #0                      ; Set the time flag
                    000F'CF     DF  2201  4153          PUSHAL  TEST_NAME               ; Push the test name
                          02    DD  2205  4154          PUSHL   #2                      ; Push arg count
                          00    EF  2207  4155          EXTZV   #STS$V_SEVERITY,-       ; Push the proper exit severity...
                          03        2209  4156                  #STS$S_SEVERITY,-
                 7E  018A'CF        220A  4157                  STATUS,-(SP)
            6E    00741080 8F CB  220E  4158          BISL2   #UETP$_ENDEDD,(SP)      ; ...and use it in our message code
                          04    DD  2215  4159          PUSHL   #4
                       51 5E    D0  2217  4160          MOVL    SP,R1
                                221A  4161          $PUTMSG_S MSGVEC = (R1)         ; Output the message
                                2229  4162          $SETPRN_S PRCNAM = ACNT_NAME    ; Reset the process name
                          04    2234  4163          RET                             ; That's all folks!
                                2235  4164
                                2235  4165          .END    UETMA7800
```

```
$$.TAB                  = 00001464 R      06        CONT_DESC                       000001CC R      05
$$.TABEND               = 000014A8 R      06        CS2                             00000273 R      05
$$.TMP                  = 00000800                  DDB_RFA                         00001294 R      06
$$.TMP1                 = 00000001                  DEAD_CTRLNAME                   000000C3 R      05
$$.TMP2                 = 0000006A                  DEVSV_TRM                     = 00000002
$$.TMPX                 = 00000016 R      07        DEVDEP_SIZE                   = 000004A8
$$.TMPX1                = 0000000D                  DEVDSC                          000000DC R      06
$$T1                    = 00000000                  DEVNAM_LEN                      000C01A8 R      06
$$T2                    = 00000006                  DEV_NAME                        000000FB R      06
ACNT_NAME                 00000000 R      05        DIB                             0000010A R      06
ADAWI_LIMIT             = 00030D40                  DIBSK_LENGTH                  = 00000074
ADPSL_CSR               = 00000000                  DIBBUF                          00000112 R      06
ADPSW_TR                = 0000000C                  DOCLEAR_MSG                     000009C3 R      05
ALL_SET                   0000060C R      08        DOCLEAR_TIMEOUT               = FA0A1F00
ALREADY_HERE              000002F3 R      05        DOSET_MSG                       00000978 R      05
ARG_COUNT                 000001CE R      06        DOSET_TIMEOUT                 = FA0A1F00
AST_MBBUF                 0000110A R      06        DOTDAT                          000002DF R      05
AST_MBCHN                 00001129 R      06        DO_FAO                          00001269 R      08
AST_MBDAY                 0000115C R      06        DO_FAO_EXIT                     000012A6 R      08
AST_MBISB                 0000112D R      06        DO_FAO_INFO                     0000128A R      08
AST_MBNAM                 0000113D R      06        DO_FAO_SIGNAL                   00001286 R      08
AST_MBPTR                 00001135 R      06        DVI$_DEVNAM                   = 00000020
BAD_CEF                   00000567 R      05        EFN2                          = 00000004
BAD_ECO                   0000071D R      05        END_UPDATE                      000021FA R      08
BAD_MB_QIO                0000064C R      05        ERROR_COUNT                     00000186 R      06
BAD_MOVC3                 00000402 R      05        ERROR_EXIT                      000020A0 R      08
BAD_MSG_TYPE              000005E2 R      05        ERROR_SUMMARY                   00001C47 R      08
BAD_REQIDT                000006D2 R      05        EXESGL_SHBLIST                  ********    X   08
BAD_UPDATE                00000457 R      05        EXIT_DESC                       000001BE R      06
BEGIN_MSGM              = 00000008                  EXIT_FROM_RCV                   00001518 R      08
BEGIN_MSGV              = 00000003                  EXIT_FROM_TMO                   00001B5D R      08
BUFFER                    00000014 R      06        EXIT_HANDLER                    00002125 R      08
BUFFER_PTR                0000000C R      06        FAB$B_BID                     = 00000000
CCASTHAND                 00002069 R      08        FAB$B_FNS                     = 00000034
CHF$L_SIGARGLST         = 00000004                  FAB$C_BID                     = 00000003
CHF$L_SIG_ARG1          = 00000008                  FAB$C_BLN                     = 00000050
CHF$L_SIG_ARGS          = 00000000                  FAB$C_FIX                     = 00000001
CHF$L_SIG_NAME          = 00000004                  FAB$C_SEQ                     = 00000000
CHKGSFAB                  00001414 R      06        FAB$C_VAR                     = 00000002
CHKGSRAB                  00001464 R      06        FAB$K_BLN                     = 00000050
CLEANUP                   00001D54 R      08        FAB$L_ALQ                     = 00000010
CNTRLCMSG                 00000082 R      05        FAB$L_DEV                     = 00000040
COMEF_EFN               = 00000060                  FAB$L_FNA                     = 0000002C
COMGSFAB                  000012EC R      06        FAB$L_FOP                     = 00000004
COMGSRAB                  0000133C R      06        FAB$L_STS                     = 00000008
COMGS_BUF                 00000208 R      06        FAB$L_STV                     = 0000000C
COMGS_EFN               = 0000003E                  FAB$M_UFO                     = 00020000
COMGS_NAME_LEN          = 00000010                  FAB$V_BIO                     = 00000005
COMGS_PVT_AREA          = 00000200                  FAB$V_CHAN_MODE               = 00000002
COMGS_PVT_CODE          = 000001F8                  FAB$V_CR                      = 00000001
COMGS_QUEUE_MAX         = 00000064                  FAB$V_FILE_MODE               = 00000004
COMGS_SIZE              = 00000008                  FAB$V_GET                     = 00000001
COMGS_VERSION           = 00003039                  FAB$V_LNM_MODE                = 00000000
COMMON                    00002030 R      08        FAB$V_PUT                     = 00000000
COMMON_WAKE_TMO           00001C29 R      08        FAB$V_SUP                     = 00000002
CONSIS_CHECK            = 30390004                  FAB$V_UFO                     = 00000011
CONTROLLER                00000031 R      05        FAB$V_UPD                     = 00000003
```

```
FAB$V_UPI                       = 00000006
FAB$W_GBC                       = 00000048
FAB$W_IFI                       = 00000002
FAO_BOF                           00000004 R      06
FILE                              000001D4 R      05
FIND_IT                           00000215 R      08
FLAG                              00000002 R      06
FOUND_IT                          000002FC R      08
GETJPI_ITMLST                     00000258 R      05
GETSYI_ITMLST                     00000230 R      05
GS_K_ADAWI_PVT                    0000003C
GS_K_END                          00000EA8
GS_K_PVT_AREAS                    000006A8
GS_K_Q_COUNT                      0000004C
GS_K_Q_ENTRIES                    00000068
GS_L_ADAWI_COM                    00000038
GS_L_AVAILABLE                    00000024
GS_L_CEF_FLAG                     00000028
GS_L_ECOLEVEL                     0000001C
GS_L_FINISHED                     00000030
GS_L_INTLK_FLAG                   00000034
GS_L_OWNER                        00000020
GS_L_PARTICIPATING                0000002C
GS_L_VERSION                      00000010
GS_Q_BUSY                         00000060
GS_Q_FREE                         00000058
GS_Q_SYSGQVERSION                 00000014
GS_T_SECT_NAME                    00000000
HELLO_TIMEOUT                   = FA0A1F00
HEXUNIT                           00001231 R      08
IADAWI_MSG                        00000B55 R      05
ICU_UCME_TIMEOUT                = FA0A1F00
ILLEGAL_REC                       00000130 R      05
INADDRESS                         00000196 R      06
INCONSISTENCY                     0000036C R      05
INIDEV_UPDERR                     00000197 R      05
INI_FAB                           00001200 R      06
INI_RAB                           00001250 R      06
INPUT_ITMLST                      00000072 R      05
INTERLOCKING_TEST                 000010F7 R      08
INTERLOCK_LIMIT                 = 0000000A
INTERLOCK_SAVED_R6                00001164 R      06
INTERPROC_EF                      00000CA4 R      08
INTERPROC_GS                      00000B94 R      08
INTLKTMO_MSG                      00000AF1 R      05
IO$M_CTRLCAST                     ********    X   08
IO$M_NOW                          ********    X   08
IO$_READVBLK                      ********    X   08
IO$_SETMODE                       ********    X   08
IO$_WRITEVBLK                     ********    X   08
IQUEUE_MSG                        00000BBE R      05
ITERATION                         000001B2 R      06
JPI$_TQCNT                      = 00000315
LC_BITM                         = 00000020
LIB$SIGNAL                        ********    X   08
MASK_2_ARGLST                     0000124D R      08
MAX_DEV_DESIG                   = 0000000A
MAX_MSG_TYPE                    = 0000000C
MAX_PROC_NAME                   = 0000000F
MAX_UNIT_DESIG                  = 00000005
MA_B_MYPORT                       000001AC
MA_K_CHKBUF                       0000264C
MA_K_CHKFAB                       00000590
MA_K_CHKRAB                       000005E0
MA_K_COMFAB                     = 00000110
MA_K_COMRAB                     = 00000160
MA_K_ERRLEN                     = 0000002C
MA_K_SPRBUF                       0000064C
MA_K_SPRFAB                       000003FC
MA_K_SPRRAB                       0000044C
MA_L_DOCLEAR                      000001E8
MA_L_DOCLEARFAIL                  000001C4
MA_L_DOSET                        000001E4
MA_L_DOSETFAIL                    000001C0
MA_L_ERRBLK                       000001B0
MA_L_HELLO                        000001DC
MA_L_IADAWI                       000001D4
MA_L_ICU_UCME                     000001E0
MA_L_INTLKTMO                     000001D0
MA_L_IQUEUE                       000001D8
MA_L_NOCAUSE                      000001C8
MA_L_OLDSTUFF                     000001CC
MA_L_RBADCODE                     000001B8
MA_L_RILOSTU                      000001B4
MA_L_SBADCODE                     000001BC
MA_L_SILOSTU                      000001B0
MA_L_SPREFSTATE                   000002CD
MA_Q_COMEFNAM                     0000022F
MA_Q_COMGSIAD                     00000624
MA_Q_COMGSNAM                     00000208
MA_Q_COMGSRAD                     0000062C
MA_Q_COMMBNAM                     00000258
MA_Q_DAYTIME                      000001A4
MA_Q_MEMNAM                       000001F1
MA_Q_SPREFNAM                     000002A6
MA_Q_SPRGSIAD                     00000634
MA_Q_SPRGSISB                     00000644
MA_Q_SPRGSNAM                     0000027F
MA_Q_SPRGSRAD                     0000063C
MA_Q_SPRMBNAM                     000002D5
MA_T_CHKFAB                       00000490
MA_T_COMEFNAM                     00000237
MA_T_COMFAB                     = 00000014
MA_T_COMGSNAM                     00000210
MA_T_COMMBNAM                     00000260
MA_T_MEMNAM                       000001F9
MA_T_SPREFNAM                     000002AE
MA_T_SPRFAB                       000002FC
MA_T_SPRGSNAM                     00000287
MA_T_SPRMBNAM                     000002DD
MA_T_UNIT                         000001F0
MA_W_COMMBCHN                     00000256
MA_W_SPRMBCHN                     000002D1
MA_W_UNIT                         000001EC
```

```
MB_B_RECEIVER                   00000001
MB_B_SENDER                     00000000
MB_B_UNIT                       00000002
MB_K_END                        0000001F
MB_K_MISC                       00000005
MB_TIMEOUT_AST                  00001AFE R      08
MB_W_WHY                        00000003
ME2ME_FAILED                    00000531 R      05
ME2ME_TIMEOUT                 = FECED300
MEM_INTERLOCK                   00000F4A R      08
MODE                            00000041 R      05
MPM$L_MR                      = 0000001C
MPM$S_MR_UNIT                 = 00000002
MPM$V_MR_UNIT                 = 0000000E
MPM_LITERAL                     00000268 R      05
MSG_BADCODE                   = 00000004
MSG_BLOCK                       000001BA R      06
MSG_BYEBYE                    = 0000000D
MSG_BYEFORNOW                 = 00000003
MSG_DOCLEAR                   = 00000007
MSG_DOSET                     = 00000005
MSG_HELLO                     = 00000001
MSG_ICU_UCME                  = 00000002
MSG_ILOSTU                    = 0000000C
MSG_INTERLOCK                 = 00000009
MSG_ISCLEAR                   = 00000008
MSG_ISSET                     = 00000006
MSG_ME2ME                     = 0000000A
MSG_ME2MEDONE                 = 0000000B
NAM$C_MAXRSS                  = 000000FF
NAME_LEN                      = 0000000F
NEEDED_PRIVS                    00000228 R      05
NEW_NODE                        000001FE R      06
NEXT_ITERATION                  00001183 R      08
NOCAUSE_MSG                     00000A10 R      05
NOUNIT_SELECTED                 0000010A R      05
NO_CTRLNAME                     000000A3 R      05
NO_RMS_AST_TABLE                0000004D R      05
NRAT_LENGTH                   = 00000014
OLDSTUFF_MSG                    00000A72 R      05
OLD_ONEPROC                     000003B2 R      05
ONEPROC_EF                      00000A4E R      08
ONEPROC_GS                      0000088A R      08
ONEPROC_MB                      00000987 R      08
ONE_SECOND                      000002EB R      05
ONE_SHOTM                     = 00000010
ONE_SHOTV                     = 00000004
OTHER_CHARS                   = 00000002
OTS$CVT_TI_L                    ********   X    08
OUTADDRESS                      0000019E R      06
PAGES                         = 00000024
PASS                            000001B6 R      06
PASS_MSG                        00000164 R      05
PMTSIZ                        = 00000019
PROCESSOR_MAX                 = 00000004
PROCESS_NAME                    000000E4 R      06
PROCESS_NAME_FREE             = 0000000B
PROC_CONT_NAME                  00000094 R      08
PROMPT                          0000020F R      05
PRV$V_PRMGBL                  = 00000018
PRV$V_SHMEM                   = 0000001B
PRV$V_SYSGBL                  = 00000019
QIOMB_AST_CHECK                 0000142B R      08
QIOMB_CHECK                     00001432 R      08
QIOMB_SYNCH_CHECK               00001422 R      08
QUAD_STATUS                     0000018E R      06
QUEUE_LIMIT                   = 000000C8
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$K_BLN                     = 00000044
RAB$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$L_UBF                     = 00000024
RAB$V_BIO                     = 0000000B
RAB$V_PMT                     = 0000001E
RAB$W_RFA                     = 00000010
RAB$W_RSZ                     = 00000022
RANDOM1                         000001AA R      06
RANDOM2                         000001AE R      06
RANDOM_FILL                     0000123D R      08
RBADCODE_MSG                    0000088D R      05
RCVMB_AST                       00001483 R      08
RCV_BADCODE                     0000171A R      08
RCV_BYEBYE                      00001A15 R      08
RCV_BYEFORNOW                   000016E7 R      08
RCV_DOCLEAR                     00001870 R      08
RCV_DOSET                       0000175A R      08
RCV_HELLO                       00001540 R      08
RCV_ICU_UCME                    00001625 R      08
RCV_ILOSTU                      000019FB R      08
RCV_INTERLOCK                   00001986 R      08
RCV_ISCLEAR                     00001940 R      08
RCV_ISSET                       0000182A R      08
RCV_ME2ME                       000019CC R      08
RCV_ME2MEDONE                   000014ED R      08
RECORD                          000001E0 R      05
REC_SIZE                      = 00000028
RESTART                         0000079A R      08
RILOSTU_MSG                     000007D5 R      05
RMS$_BLN                        ********     X  05
RMS$_BUSY                       ********     X  05
RMS$_CDA                        ********     X  05
RMS$_FAB                        ********     X  05
RMS$_FACILITY                 = 00000001
RMS$_RAB                        ********     X  05
```

```
RMS_ERROR                       00001FFA R      08
RMS_ERR_STRING                  000001EE R      05
SAFE_TO_UPDM                =   00000004
SAFE_TO_UPDV                =   00000002
SBADCODE_MSG                    00000905 R      05
SCSNODE                         000001F4 R      06
SCSNODE_LENGTH                  000001FA R      06
SEC$M_EXPREG                    ********   X    08
SEC$M_GBL                       ********   X    08
SEC$M_PERM                      ********   X    08
SEC$M_WRT                       ********   X    08
SHB$B_NEXUS                 =   00000014
SHB$B_PORT                  =   00000015
SHB$L_ADP                   =   0000001C
SHB$L_DATAPAGE              =   00000004
SHB$L_LINK                  =   00000000
SHD$T_NAME                  =   00000020
SHM_INFO                        000012B9 R      08
SHM_INFO_ARGS                   000001E0 R      06
SHR$_ABENDD                 =   000010E0
SHR$_BEGIND                 =   00001038
SHR$_ENDEDD                 =   00001080
SHR$_OPENIN                 =   00001098
SHR$_TEXT                   =   00001130
SILOSTU_MSG                     00000836 R      05
SPREF_EFN                   =   00000040
SPRGSFAB                        00001380 R      06
SPRGSRAB                        000013D0 R      06
SPRGS_AST                       0000130F R      08
SPRGS_EFN                   =   0000003F
SPRGS_SIZE                  =   00000010
SS$_BADPARAM                =   00000014
SS$_CONTROLC                =   00000651
SS$_CREATED                 =   00000619
SS$_NORMAL                  =   00000001
SS$_NOSUCHSEC               =   00000978
SS$_SSFAIL                  =   0000045C
SS$_WASSET                  =   00000009
SSERROR                         00001F17 R      08
SS_SYNCH_EFN                =   00000003
STATUS                          0000018A R      06
STR$UPCASE                      ********   X    08
STS$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                        00001190 R      08
SUMMARY_HEADER                  000007A0 R      05
SUPDEV_GBLSEC                   00000020 R      05
SUP_FAB                         0000129C R      06
SYI$_SCSNODE                =   00001067
SYI$_SID                    =   00001001
SYI$_VERSION                =   00001000
SYNCH_MBBUF                     000010B0 R      06
SYNCH_MBCHN                     000010CF R      06
SYNCH_MBDAY                     00001102 R      06
SYNCH_MBISB                     000010D3 R      06
SYNCH_MBNAM                     000010E3 R      06
SYNCH_MBPTR                     000010DB R      06
SYS$ASCEFC                      ********  GX    08
SYS$ASSIGN                      ********  GX    08
SYS$CANTIM                      ********  GX    08
SYS$CLOSE                       ********  GX    08
SYS$CLRAST                      ********   X    08
SYS$CLREF                       ********  GX    08
SYS$CMKRNL                      ********  GX    08
SYS$CONNECT                     ********  GX    08
SYS$CREATE                      ********  GX    08
SYS$CREMBX                      ********  GX    08
SYS$CRMPSC                      ********  GX    08
SYS$DACEFC                      ********  GX    08
SYS$DASSGN                      ********  GX    08
SYS$DCLEXH                      ********  GX    08
SYS$DELTVA                      ********  GX    08
SYS$DGBLSC                      ********  GX    08
SYS$ERASE                       ********  GX    08
SYS$EXIT                        ********  GX    08
SYS$EXPREG                      ********  GX    08
SYS$FAO                         ********   X    08
SYS$FAOL                        ********  GX    08
SYS$GET                         ********  GX    08
SYS$GETDVI                      ********  GX    08
SYS$GETJPI                      ********  GX    08
SYS$GETMSG                      ********  GX    08
SYS$GETSYIW                     ********  GX    08
SYS$GETTIM                      ********  GX    08
SYS$HIBER                       ********  GX    08
SYS$INPUT                       00000061 R      05
SYS$MGBLSC                      ********  GX    08
SYS$OPEN                        ********  GX    08
SYS$PUTMSG                      ********  GX    08
SYS$QIO                         ********  GX    08
SYS$QIOW                        ********  GX    08
SYS$READ                        ********  GX    08
SYS$READEF                      ********  GX    08
SYS$SCHDWK                      ********  GX    08
SYS$SETAST                      ********  GX    08
SYS$SETEF                       ********  GX    08
SYS$SETIMR                      ********  GX    08
SYS$SETPRN                      ********  GX    08
SYS$SETPRV                      ********  GX    08
SYS$SETSFM                      ********  GX    08
SYS$TEST                        000002CE R      05
SYS$TRNLOG                      ********  GX    08
SYS$UPDATE                      ********  GX    08
SYS$UPDSEC                      ********  GX    08
SYS$WAITFR                      ********  GX    08
SYS$WAKE                        ********  GX    08
SYS$WRITE                       ********  GX    08
```

```
SYSIN_FAB                       0000116C R      06        UPDSEC_HWE                      000004A8 R      05
SYSIN_RAB                       000011BC R      06        WRITE_SIZE                    = 00002000
TEST_NAME                       0000000F R      05        WRONG_VERSION                   0000075E R      05
TEST_OVERM                    = 00000002                  XX                            = 0000000E
TEST_OVERV                    = 00000001
TEXT_BUFFER                   = 000000C8
THREEMIN                        000001BC R      05
TIME_IT                         00000787 R      08
TIME_OUT                        00001F0F R      08
TMO_BADCODE                   = 00001B4A R      08
TMO_BYEBYE                    = 00001B4A R      08
TMO_BYEFORNOW                 = 00001B4A R      08
TMO_DOCLEAR                     00001C24 R      08
TMO_DOSET                       00001C1D R      08
TMO_HELLO                       00001C37 R      08
TMO_ICU_UCME                    00001C3F R      08
TMO_ILOSTU                    = 00001B4A R      08
TMO_INTERLOCK                 = 00001B4A R      08
TMO_ISCLEAR                   = 00001B4A R      08
TMO_ISSET                     = 00001B4A R      08
TMO_ME2ME                       00001C29 R      08
TMO_ME2MEDONE                 = 00001B4A R      08
TQCNT                           00001168 R      06
TTCHAN                          00000000 R      06
UETCOMEF                        000002AE R      05
UETCOMGS                        00000281 R      05
UETMA7800                       00000000 RG     08
UETMB                           000002A1 R      05
UETP                          = 00740000
UETP$_ABENDD                  = 007410E0
UETP$_ABORTC                  = 0074832B
UETP$_BEGIND                  = 00741038
UETP$_DATADEVERR              = 00748018
UETP$_DENOSU                  = 00748333
UETP$_ENDEDD                  = 00741080
UETP$_ERBOXPROC               = 00748020
UETP$_FACILITY                = 00000074
UETP$_OPENIN                  = 00741098
UETP$_TEXT                    = 00741130
UETSPREF                        000002BE R      05
UETSPRGS                        00000291 R      05
UETUNT$B_FLAGS                = 0000000B
UETUNT$B_TYPE                 = 00000008
UETUNT$C_DEVDEP               = 000001A4
UETUNT$C_INDSIZ               = 000001A4
UETUNT$K_FAB                  = 00000110
UETUNT$K_RAB                  = 00000160
UETUNT$M_TESTABLE             = 00000002
UETUNT$T_FILSPC               = 00000014
UETUNT$V_TESTABLE             = 00000001
UETUNT$W_SIZE                 = 00000009
UNIT_DESC                       000001C4 R      05
UNIT_LIST                       000001D8 R      06
UNIT_LOOP                       00002177 R      08
UNIT_NUMBER                     000001A6 R      06
UPDATE_FAILED                   000021DC R      08
UPDSEC_FAILED                   000004D4 R      05
```

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	0000464C (17996.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
INTER_PROC_MB	0000001F (31.)	03 (3.)	NOPIC USR CON ABS LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
INTERPROC_GS_DEF	00000EA8 (3752.)	04 (4.)	NOPIC USR CON ABS LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RODATA	00000C50 (3152.)	05 (5.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RWDATA	000014A8 (5288.)	06 (6.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
$RMSNAM	00000023 (35.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
MA780	00002235 (8757.)	08 (8.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.09	00:00:00.49
Command processing	140	00:00:00.69	00:00:03.15
Pass 1	1240	00:00:44.72	00:01:23.89
Symbol table sort	0	00:00:03.96	00:00:05.72
Pass 2	565	00:00:12.94	00.00:25.01
Symbol table output	5	00:00:00.44	00:00:01.31
Psect synopsis output	3	00:00:00.06	00:00:00.38
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1993	00:01:02.92	00:01:59.97

The working set limit was 2000 pages.
256661 bytes (502 pages) of virtual memory were used to buffer the intermediate code.
There were 140 pages of symbol table space allocated to hold 2509 non-local and 181 local symbols.
4165 source lines were read in Pass 1, producing 68 object records in Pass 2.
92 pages of virtual memory were used to define 84 macros.

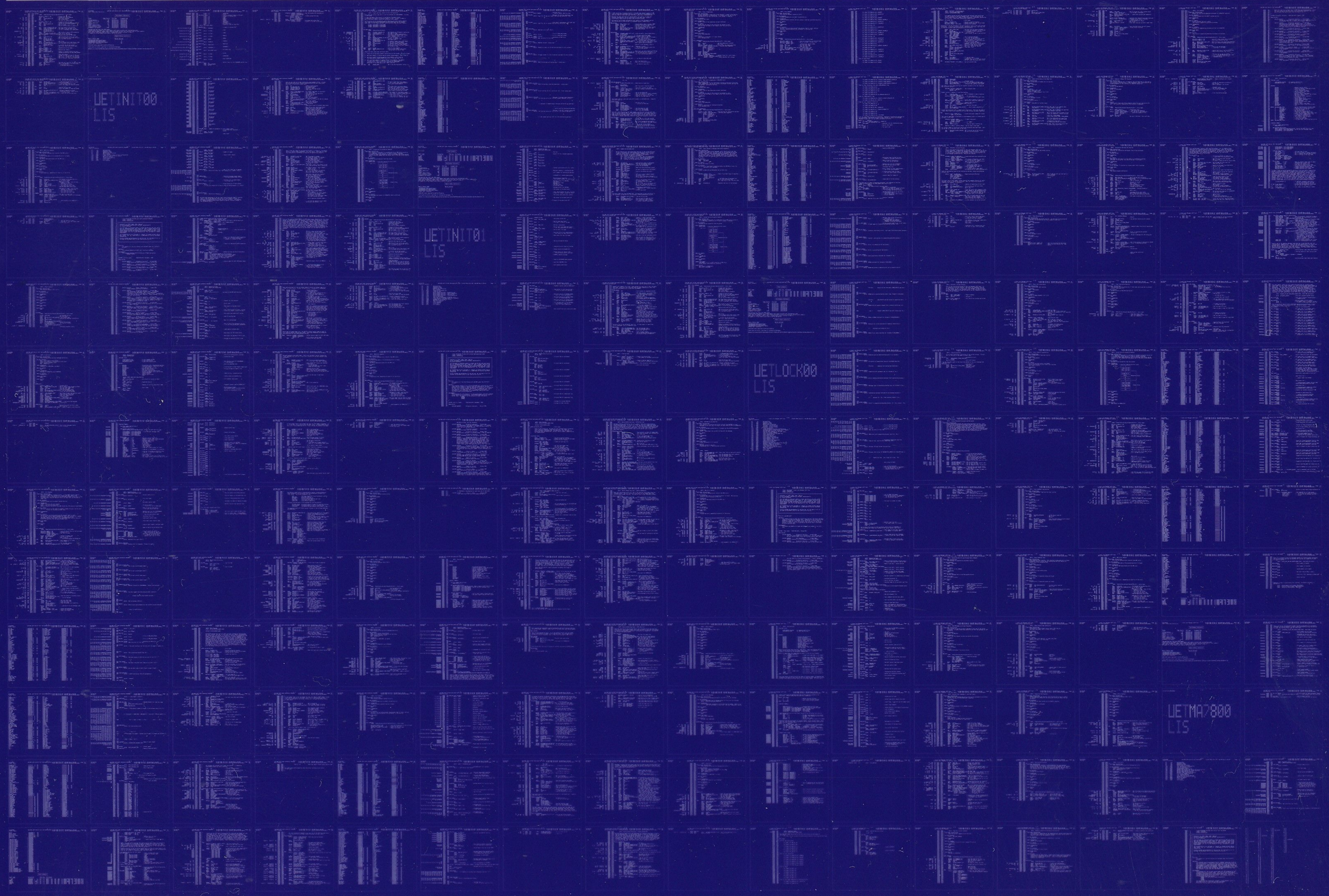
Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]UETP.MLB;1	2
_255DUA28:[SYS.OBJ]LIB.MLB;1	4
_255DUA28:[SYSLIB]STARLET.MLB;2	73
TOTALS (all libraries)	79

2798 GETS were required to define 79 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:UETMA7800/OBJ=OBJ$:UETMA7800 MSRC$:UETMA7800/UPDATE=(ENH$:UETMA7800)+EXECML$/LIB+SHRLIB$:UETP/LIB
```



DISKMON
MAP

UETMEMY01
LIS

UETSSMM00
LIS

DCLDIET
MAP

DISKQ
MAP

UTIL32

CHECKSUM
MAP