F 13

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

FILEID**UETMA7800


```
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 MM AA      AA       77 88      88 00    0000 00    0000
UU      UU EE             TT     MM MM MM MM AA      AA       77 88      88 00    0000 00    0000
UU      UU EEEEEEEE       TT     MM  MM  MM AA      AA      77    888888   00  00  00 00  00  00
UU      UU EEEEEEEE       TT     MM  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            .TITLE UETMA7800 VAX/VMS UETP DEVICE TEST FOR MA780
0000            .IDENT  'V04-000'
0000            .ENABLE SUPPRESSION
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000    ;++
0000    ; FACILITY:
0000    ;       This module will be distributed with VAX/VMS under the [SYSTEST]
0000    ;       account.
0000    ;
0000    ; ABSTRACT:
0000    ;       This program tests shared memory unit(s) (MA780) connected to one or
0000    ;       more VAX processors.  Should any other processors be running the same
0000    ;       test concurrently, the program takes advantage of that to use the
0000    ;       shared memory to communicate between them.  IT IS NOT THE INTENTION OF
0000    ;       THIS PROGRAM TO BE SELF DOCUMENTING, ALTHOUGH CONSIDERABLE EFFORT HAS
0000    ;       BEEN MADE TO PROVIDE USEFUL COMMENTS.  FOR AN EXPLANATION OF why THINGS
0000    ;       ARE DONE THE WAY THAT THEY ARE AND HOW VARIOUS DATA STRUCTURES AND
0000    ;       CODE INTERACT, PLEASE READ THE FUNCTIONAL SPEC.
0000    ;
0000    ; ENVIRONMENT:
0000    ;       This program will run in user access mode with the exception of two
0000    ;       routines which run in kernal mode.  The first routine finds memory
0000    ;       parameters associated with a given shared memory and the second returns
0000    ;       the operating system version and hardware ECO levels.  ASTs are
0000    ;       enabled except during error processing.  This program requires the
0000    ;       following privileges and quotas:
0000    ;               CMKRNL (to read shared memory parameters)
0000    ;               SHMEM
0000    ;               SYSGBL
0000    ;               PRMGBL
0000    ;       This module has one conditional assembly, FT_NO_INTERACT, which is
0000    ;       described where it is defined (or left undefined).
0000    ;       Successful linking requires SYS$SYSTEM:SYS.STB.
```

```
0000     ;       When this test was written, only 11/780 CPUs had shared memory.  Since
0000     ;       the test relies on the ECO level from a processor register, the test
0000     ;       must be modified should other CPUs eventually run it.
0000     ;
0000     ;--
0000     ;
0000     ; AUTHOR: Richard N. Holstein,  CREATION DATE: August, 1981
0000     ;
0000     ; MODIFIED BY:
0000     ;
0000     ;       V03-012 RNH0011         Richard N. Holstein,    13-Jul-1984
0000     ;               Include our node name in global section file names to allow for
0000     ;               simultaneous runs on a cluster with a common SYS$TEST.
0000     ;
0000     ;       V03-011 RNH0010         Richard N. Holstein,    15-Feb-1984
0000     ;               Take advantage of new UETP message codes.  Fix SSERROR
0000     ;               interaction with RMS_ERROR.  Account for the possibility of
0000     ;               11/785 systems with shared memory - modify what we believe the
0000     ;               ECO level field to be in the PR$_SID register.
0000     ;
0000     ;       V03-010 RNH0009         Richard N. Holstein,    19-Dec-1983
0000     ;               Give correct sentinels to Test Controller.
0000     ;
0000     ;       V03-009 RNH0008         Richard N. Holstein,    21-Nov-1983
0000     ;               Use decimal conversion routine for unit numbers.
0000     ;
0000     ;       V03-008 RNH0007         Richard N. Holstein,    11-Mar-1983
0000     ;               Don't signal ending message in EXIT_HANDLER.
0000     ;
0000     ;       V03-007 RNH0006         Richard N. Holstein,    25-Feb-1983
0000     ;               Allow for longer device names.
0000     ;
0000     ;       V03-006 RNH0005         Richard N. Holstein,    22-Dec-1982
0000     ;               Fixes from the V3B UETP Workplan: better message if sending
0000     ;               interprocessor mail fails, close INTERLOCKING_TEST window,
0000     ;               reduce kernel mode code.
0000     ;
0000     ;       V03-005 RNH0004         Richard N. Holstein,    15-Oct-1982
0000     ;               Miscellaneous fixes listed in the V3B UETP Workplan.
0000     ;
0000     ;       V03-004 LDJ0004         Larry D. Jones,         15-Apr-1982
0000     ;               Turn off interprocessor interaction - include definition of
0000     ;               FT_NO_INTERACT.
0000     ;
0000     ;       V03-003 RNH0003         Richard N. Holstein,    29-Mar-1982
0000     ;               Turn on interprocessor interaction - remove definition of
0000     ;               FT_NO_INTERACT.
0000     ;
0000     ;       V03-002 RNH0002         Richard N. Holstein,    05-Feb-1982
0000     ;               Don't let RCVMB_AST blithely continue if it sees a mailbox
0000     ;               before we're ready for one.  Don't $CMKRNL to do a $SETPRV,
0000     ;               SYSTEST now has SETPRV privilege.
0000     ;
0000     ;       V03-001 RNH0001         Richard N. Holstein,    28-Aug-1981
0000     ;               Final development for Edition A, Field Test 1.
0000     ;
0000     ;**
```

```
                   0000          .SBTTL  Declarations
                   0000  ;
                   0000  ; INCLUDE FILES:
                   0000  ;
                   0000  ;       SYS$LIBRARY:LIB.MLB      for general definitions
                   0000  ;       SHRLIB$:UETP.MLB         for UETP definitions
                   0000
                   0000  ;
                   0000  ; MACROS:
                   0000  ;
                   0000          $ADPDEF                          ; Adapter control block
                   0000          $CHFDEF                          ; Condition handler frame definitions
                   0000          $DEVDEF                          ; Device definitions
                   0000          $DIBDEF                          ; Device Information Block
                   0000          $DVIDEF                          ; $GETDVI ITMLST item codes
                   0000          $FABDEF                          ; File Access Block
                   0000          $JPIDEF                          ; $GETJPI ITMLST item codes
                   0000          $MPMDEF                          ; Multiport memory adapter reg. offsets
                   0000          $NAMDEF                          ; NAM block offsets
                   0000          $PRDEF                           ; Internal processor registers
                   0000          $PRVDEF                          ; Privileges
                   0000          $RABDEF                          ; Record Access Block
                   0000          $SHBDEF                          ; Shared memory control block
                   0000          $SHDDEF                          ; Shared memory datapage
                   0000          $SHRDEF                          ; Shared messages
                   0000          $SSDEF                           ; System Service status codes
                   0000          $STSDEF                          ; Status return
                   0000          $SYIDEF                          ; $GETSYI ITMLST item codes
                   0000          $UETUNTDEF                       ; UETP unit block offset definitions
                   0000          $UETPDEF                         ; UETP
                   0000
                   0000  .MACRO  MAILBOX_MSG_TYPES                ; Define interprocessor mailbox codes
                   0000          X       HELLO,10                 ; Announce me to another processor
                   0000          X       ICU_UCME,10              ; I see you, do you see me?
                   0000          X       BYEFORNOW                ; We see each other
                   0000          X       BADCODE                  ; You need glasses to see me
                   0000          X       DOSET,10                 ; Please set a common event flag
                   0000          X       ISSET                    ; I set the requested common event flag
                   0000          X       DOCLEAR,10               ; Please clear a common event flag
                   0000          X       ISCLEAR                  ; I cleared the requested CEF
                   0000          X       INTERLOCK                ; Start memory interlock test
                   0000          X       ME2ME,2                  ; I'm trying to talk to myself
                   0000          X       ME2MEDONE                ; I can talk to myself
                   0000          X       ILOSTU                   ; I think you went away
                   0000          X       BYEBYE                   ; I've gone away, this is my last action
                   0000  .ENDM   MAILBOX_MSG_TYPES
                   0000
                   0000  ;
                   0000  ; EQUATED SYMBOLS:
                   0000  ;
                   0000  ;   Facility number definitions:
         00000001  0000          RMS$_FACILITY = 1
                   0000
                   0000  ;   SHR message definitions:
         00740000  0000          UETP = UETP$_FACILITY@STS$V_FAC_NO ; Define the UETP facility code
         007410E0  0000          UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
         00741038  0000          UETP$_BEGIND = UETP!SHR$_BEGIND
```

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

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

```
          0000  ;
          0000  ; For each memory, there will be a data structure set up, called a node.
          0000  ; These nodes will be linked together in a self-relative queue whose header is
          0000  ; UNIT_LIST.  The first part of each node will be the standard definition from
          0000  ; $UETUNTDEF.  Following that will come the device test dependent stuff,
          0000  ; defined below.  NOTE THAT THIS DEFINITION IS DONE WITH AN ABSOLUTE PSECT.
          0000  ; This means that what look like declarations are really definitions and the
          0000  ; labels are really just offsets into a given node on the queue.  (A not
          0000  ; necessarily obvious consequence of using an ABS PSECT is that space must be
          0000  ; reserved with .BLKx operations, since .BYTE, etc., attempt to store data.)
          0000  ;
          0000          .PSECT  DEVDEP_STR_DEF,ABS,NOEXE,NOWRT,PAGE ; Note ABS attribute!
          0000
000001A4  0000          .BLKB   UETUNT$C_DEVDEP              ; Skip over standard UETUNT block
          01A4
          01A4  MA_Q_DAYTIME:                                ; Code used to ensure interprocessor...
000001AC  01A4          .BLKQ   1                            ; ...communication
          01AC
          01AC  MA_B_MYPORT:                                 ; My port number on this memory
000001AD  01AC          .BLKB   1                            ; Since BB%%I needs a longword, allow...
000001B0  01AD          .BLKB   3                            ; ...filler to save much code later on
          01B0
          01B0  ; The following all denote error conditions and may be accessed as a single
          01B0  ; block; hence there is a reliance on their being contiguous.  There is no
          01B0  ; restriction on their ordering.
          01B0  MA_L_ERRBLK:                                 ; Start of contiguous area
          01B0
          01B0  MA_L_SILOSTU:                                ; Bit mask. Bit lit if memory port...
000001B4  01B0          .BLKL   1                            ; ...was sent an ILOSTU mailbox
          01B4
          01B4  MA_L_RILOSTU:                                ; Bit mask. Bit lit if memory port...
000001B8  01B4          .BLKL   1                            ; ...sent me an ILOSTU mailbox
          01B8
          01B8  MA_L_RBADCODE:                               ; Bit mask. Bit lit if memory port...
000001BC  01B8          .BLKL   1                            ; ...sent me a BADCODE mailbox
          01BC
          01BC  MA_L_SBADCODE:                               ; Bit mask. Bit lit if memory port...
000001C0  01BC          .BLKL   1                            ; ...was sent a BADCODE mailbox
          01C0
          01C0  MA_L_DOSETFAIL:                              ; Bit mask. Bit lit if memory port...
000001C4  01C0          .BLKL   1                            ; ...failed to set requested bit
          01C4
          01C4  MA_L_DOCLEARFAIL:                            ; Bit mask. Bit lit if memory port...
000001C8  01C4          .BLKL   1                            ; ...failed to clear requested bit
          01C8
          01C8  MA_L_NOCAUSE:                                ; Bit mask. Bit lit if memory port...
000001CC  01C8          .BLKL   1                            ; ...sent me a mailbox for no cause
          01CC
          01CC  MA_L_OLDSTUFF:                               ; Bit mask. Bit lit if memory port...
000001D0  01CC          .BLKL   1                            ; ...sent me mailbox before I was ready
          01D0
          01D0  MA_L_INTLKTMO:                               ; Bit mask. Bit lit if memory port...
000001D4  01D0          .BLKL   1                            ; ...failed to finish interlock test
          01D4
          01D4  MA_L_IADAWI:                                 ; Bit mask. Bit lit if memory port...
000001D8  01D4          .BLKL   1                            ; ...produced an inconsistent ADAWI sum
          01D8
```

```
          01D8   MA_L_IQUEUE:                        ; Bit mask. Bit lit if memory port...
000001DC  01D8           .BLKL   1                   ; ...left an inconsistent queue
          01DC
0000002C  01DC   MA_K_ERRLEN = .-MA_L_ERRBLK         ; End of items which must be contiguous
          01DC
          01DC   MA_L_HELLO:                         ; Bit mask. Bit lit if memory port...
000001E0  01DC           .BLKL   1                   ; ...was sent HELLO mailbox, need reply
          01E0
          01E0   MA_L_ICU_UCME:                      ; Bit mask. Bit lit if memory port...
000001E4  01E0           .BLKL   1                   ; ...was sent ICU_UCME MB, need reply
          01E4
          01E4   MA_L_DOSET:                         ; Bit mask. Bit lit if memory port...
000001E8  01E4           .BLKL   1                   ; ...was sent DOSET MB, need reply
          01E8
          01E8   MA_L_DOCLEAR:                       ; Bit mask. Bit lit if memory port...
000001EC  01E8           .BLKL   1                   ; ...was sent DOCLEAR MB, need reply
          01EC
          01EC   MA_W_UNIT:                          ; Unit number of this memory
000001EE  01EC           .BLKW   1                   ; To parallel code in the exec, a word
000001F0  01EE           .BLKW   1                   ; is allocated, but really only a byte
          01F0                                       ; is used.  Another word is allocated
          01F0                                       ; to allow PUSHL for system service calls
          01F0
          01F0   MA_T_UNIT:                          ; Holds ASCII equivalent of hex unit
000001F1  01F0           .BLKB   1
          01F1
          01F1   MA_Q_MEMNAM:                        ; Descriptor for...
000001F9  01F1           .BLKQ   1
          01F9   MA_T_MEMNAM:                        ; ...memory name from SYSGEN
00000208  01F9           .BLKB   15
          0208
          0208   MA_Q_COMGSNAM:                      ; Descriptor for...
00000210  0208           .BLKQ   1
          0210   MA_T_COMGSNAM:                      ; ...interprocessor global section name
0000022F  0210           .BLKB   15+1+15             ; shared-mem-name:global-sec-name
          022F
          022F   MA_Q_COMEFNAM:                      ; Descriptor for...
00000237  022F           .BLKQ   1
          0237   MA_T_COMEFNAM:                      ; ...interprocessor common EF cluster
00000256  0237           .BLKB   15+1+15             ; shared-mem-name:cluster-name
          0256
          0256   MA_W_COMMBCHN:                      ; Channel for another processor's MB
00000258  0256           .BLKW   1
          0258
          0258   MA_Q_COMMBNAM:                      ; Skeleton of descriptor for...
00000260  0258           .BLKQ   1
          0260   MA_T_COMMBNAM:                      ; ...another processor's mailbox
0000027F  0260           .BLKB   15+1+15             ; shared-mem-name:mailbox-name
          027F
          027F   MA_Q_SPRGSNAM:                      ; Descriptor for...
00000287  027F           .BLKQ   1
          0287   MA_T_SPRGSNAM:                      ; ...my single processor global section
000002A6  0287           .BLKB   15+1+15             ; shared-mem-name:global-sec-name
          02A6
          02A6   MA_Q_SPREFNAM:                      ; Descriptor for...
000002AE  02A6           .BLKQ   1
          02AE   MA_T_SPREFNAM:                      ; ...Single processor CEF cluster
```

```
000002CD  02AE          .BLKB   15+1+15                         ; shared-mem-name:cluster-name
          02CD
          02CD  MA_L_SPREFSTATE:                                ; Single processor CEF state vector
000002D1  02CD          .BLKL   1
          02D1
          02D1  MA_W_SPRMBCHN:                                  ; Channel for this processor's MB
000002D3  02D1          .BLKW   1
000002D5  02D3          .BLKW   1                               ; Filler allows easier use in SS calls
          02D5
          02D5  MA_Q_SPRMBNAM:                                  ; Descriptor for...
000002DD  02D5          .BLKQ   1
          02DD  MA_T_SPRMBNAM:                                  ; ...this processor's mailbox
000002FC  02DD          .BLKB   15+1+15                         ; shared-mem-name:mailbox-name
          02FC
00000014  02FC  MA_T_COMFAB = UETUNT$T_FILSPC                   ; Define these symbols as the...
00000110  02FC  MA_K_COMFAB = UETUNT$K_FAB                      ; ...standard UETUNT symbols...
00000160  02FC  MA_K_COMRAB = UETUNT$K_RAB                      ; ...just so all structures look alike
          02FC
          02FC  MA_T_SPRFAB:                                    ; .ASCII filespec for single processor GS
000003FB  02FC          .BLKB   NAM$C_MAXRSS
          03FB
          03FB          .ALIGN  LONG
          03FC  MA_K_SPRFAB:                                    ; Space for single processor section FAB
0000044C  03FC          .BLKB   FAB$K_BLN                       ; See note where COMGSFAB is defined
          044C
          044C  MA_K_SPRRAB:                                    ; Space for single processor section RAB
00000490  044C          .BLKB   RAB$K_BLN                       ; See note where COMGSFAB is defined
          0490
          0490  MA_T_CHKFAB:                                    ; .ASCII filespec to check single proc GS
0000058F  0490          .BLKB   NAM$C_MAXRSS
          058F
          058F          .ALIGN  LONG
          0590  MA_K_CHKFAB:                                    ; Space for single proc section check FAB
000005E0  0590          .BLKB   FAB$K_BLN                       ; See note where COMGSFAB is defined
          05E0
          05E0  MA_K_CHKRAB:                                    ; Space for single proc section check RAB
00000624  05E0          .BLKB   RAB$K_BLN                       ; See note where COMGSFAB is defined
          0624
          0624  MA_Q_COMGSIAD:                                  ; Suggested starting address for interproc G
0000062C  0624          .BLKL   2
          062C
          062C  MA_Q_COMGSRAD:                                  ; Actual starting address for interproc GS
00000634  062C          .BLKL   2
          0634
          0634  MA_Q_SPRGSIAD:                                  ; Suggested starting addr for single proc GS
0000063C  0634          .BLKL   2
          063C
          063C  MA_Q_SPRGSRAD:                                  ; Actual starting addr for single proc GS
00000644  063C          .BLKL   2
          0644
          0644  MA_Q_SPRGSISB:                                  ; IOSB for $UPDSEC of single processor GS
0000064C  0644          .BLKQ   1
          064C
000004A8  064C          DEVDEP_SIZE    = .-UETUNT$C_DEVDEP ; Device dependent part of UETUNT
          064C
          064C  MA_K_SPRBUF:                                    ; Buffer to store single processor GS data
0000264C  064C          .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  ;
                 464C  ; There is a standard format for messages passed via interprocessor mailboxes.
                 464C  ; See the functional spec and the mailbox AST routine for summaries of what
                 464C  ; is actually passed.
                 464C  ;
                 464C          .PSECT  INTER_PROC_MB,ABS,NOEXE,NOWRT,PAGE ; Note ABS attribute!
                 0000
                 0000  MB_B_SENDER:                            ; Port number of sender
       00000001  0000          .BLKB   1
                 0001
                 0001  MB_B_RECEIVER:                          ; Port number of receiver
       00000002  0001          .BLKB   1
                 0002
                 0002  MB_B_UNIT:                              ; Shared memory unit number
       00000003  0002          .BLKB   1
                 0003
                 0003  MB_W_WHY:                               ; Why sent (one of MSG_* codes)
       00000005  0003          .BLKW   1
                 0005
                 0005  MB_K_MISC:                              ; Other misc info, dependent on MB_W_WHY
       0000001F  0005          .BLKB   26
                 001F
                 001F  MB_K_END:                               ; End of mailbox
                 001F
                 001F
                 001F
                 001F
                 001F  ;
                 001F  ; Whenever a timer is associated with a mailbox, the REQIDT parameter is
                 001F  ; defined as:
                 001F  ;
                 001F  ;       .BYTE   shared-memory-unit-number (MA_W_UNIT)
                 001F  ;       .BYTE   receiver's-port-number (MB_B_RECEIVER)
                 001F  ;       .WORD   reason-for-sending-mail (MB_W_WHY)
                 001F  ;
```

```
                001F  ;
                001F  ; There will be an interprocessor global section accessed by all processors
                001F  ; attached to a shared memory.  To get a good description of how the section
                001F  ; is used, see the functional spec for this test.
                001F  ;
                001F          .PSECT  INTERPROC_GS_DEF,ABS,NOEXE,NOWRT,PAGE ; Note ABS attribute!
                0000
                0000  GS_T_SECT_NAME:                            ; ASCIC global section name
00000010        0000          .BLKB   COMGS_NAME_LEN             ; Use this for consistency checking
                0010
                0010  GS_L_VERSION:                              ; UETMA7800 GS version, max ports
00000014        0010          .BLKL   1                          ; (Interprocessor consistency check)
                0014
                0014  GS_Q_SYSGQVERSION:                         ; VMS .ASCII version number
0000001C        0014          .BLKQ   1
                001C
                001C  GS_L_ECOLEVEL:                             ; Hardware ECO level of owner CPU
00000020        001C          .BLKL   1
                0020
                0020  GS_L_OWNER:                                ; Port number of creating processor
00000024        0020          .BLKL   1
                0024
                0024  GS_L_AVAILABLE:                            ; Bit mask.  Bit lit if memory port...
00000028        0024          .BLKL   1                          ; ...has cooperating processor
                0028
                0028  GS_L_CEF_FLAG:                             ; Flag which marks interprocessor...
0000002C        0028          .BLKL   1                          ; CEF test in progress (flag and port)
                002C
                002C  GS_L_PARTICIPATING:                        ; Bit mask of procs starting interlock test
00000030        002C          .BLKL   1                          ; Must immediately preceed GS_L_FINISHED
                0030
                0030  GS_L_FINISHED:                             ; Bit mask of procs finishing interlock test
00000034        0030          .BLKL   1                          ; Must immediately follow GS_L_PARTICIPATING
                0034
                0034  GS_L_INTLK_FLAG:                           ; Flag which marks interlocking...
00000038        0034          .BLKL   1                          ; ...test in progress (flag and port)
                0038
                0038          .ALIGN  WORD                       ; ADAWI must be word aligned
                0038  GS_L_ADAWI_COM:                            ; Common ADAWI count word - grand total
0000003C        0038          .BLKW   1+1                        ; Includes overflow word, as well
                003C
                003C  GS_K_ADAWI_PVT:                            ; Private ADAWI count per processor
0000004C        003C          .BLKW   PROCESSOR_MAX*<1+1>        ; ADAWI, overflow. Adjacent to ADAWI_COM
                004C
                004C  GS_K_Q_COUNT:                              ; Per processor count of ''busy''...
00000054        004C          .BLKW   PROCESSOR_MAX              ; ...queue entries
                0054
                0054          .ALIGN  QUAD
                0058  GS_Q_FREE:                                 ; ''Free'' queue header
00000060        0058          .BLKQ   1                          ; Keep adjacent to ''busy'' queue header
                0060
                0060  GS_Q_BUSY:                                 ; ''Busy'' queue header
00000068        0060          .BLKQ   1                          ; Keep adjacent to queue entries
                0068
                0068  GS_K_Q_ENTRIES:                            ; Busy and free queue entries
000006A8        0068          .BLKL   COMGS_QUEUE_MAX*<1+1+1+1> ; Flink, blink, port number, filler
                06A8
```

```
                  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'    0000   551              .ASCID  /SYSTEST/
                                  54    000E
                                        000F   552
                                        000F   553 TEST_NAME:                                       ; This test name
37 41 4D 54 45 55 00000017'010E0000'    000F   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'    0020   557              .ASCID  /UETSUPDEV/
                            56 45 44    002E
                                        0031   558
                                        0031   559 CONTROLLER:                                      ; Logical name of controller
41 4E 4C 52 54 43 00000039'010E0000'    0031   560              .ASCID  /CTRLNAME/
                               45 4D    003F
                                        0041   561
                                        0041   562 MODE:                                            ; Run mode logical name
      45 44 4F 4D 00000049'010E0000'    0041   563              .ASCID  /MODE/
                                        004D   564
                                        004D   565 NO_RMS_AST_TABLE:                                ; List of errors for which...
                           00000000'    004D   566              .LONG   RMS$_BLN                    ; ...RMS cannot deliver an AST...
                           00000000'    0051   567              .LONG   RMS$_BUSY                   ; ...even if one has an ERR= arg
                           00000000'    0055   568              .LONG   RMS$_CDA                    ; Note that we can search table...
                           00000000'    0059   569              .LONG   RMS$_FAB                    ; ...via MATCHC since <31:16>...
                           00000000'    005D   570              .LONG   RMS$_RAB                    ; ...pattern can't be in <15:0>
                            00000014    0061   571 NRAT_LENGTH = .-NO_RMS_AST_TABLE
                                        0061   572
                                        0061   573 SYS$INPUT:                                       ; Name of device from which...
4E 49 24 53 59 53 00000069'010E0000'    0061   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    0072   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'    0082   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'    00A3   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'    00C3   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'  01EE   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 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          .WORD   4,SYI$_SID                  ; Read PR$_SID for me
                   00000224' 0234          .ADDRESS COMGS_BUF+GS_L_ECOLEVEL
                   00000000  0238          .LONG   0
                  1000 0008  023C          .WORD   8,SYI$_VERSION              ; What version of VMS is running?
                   0000021C' 0240          .ADDRESS COMGS_BUF+GS_Q_SYSGQVERSION
                   00000000  0244          .LONG   0
                  1067 0006  0248          .WORD   6,SYI$_SCSNODE              ; What's our cluster node name?
                   000001F4' 024C          .ADDRESS SCSNODE
                   000001FA' 0250          .ADDRESS SCSNODE_LENGTH
                   00000000  0254          .LONG   0                           ; End of list of processor info items
                             0258
                             0258  GETJPI_ITMLST:                              ; Return timer request info
                  0315 0004  0258          .WORD   4,JPI$_TQCNT
                   0000116B' 025C          .ADDRESS TQCNT
          00000000 00000000  0260          .LONG   0,0
                             0268
                             0268  MPM_LITERAL:                                ; Literal string needed to check that...
  4D 50 4D 00000270'010E0000' 0268         .ASCID  /MPM/                       ; ...we're really given a shared memory
                             0273
                             0273  CS2:                                        ; No device type, any device class...
20 2A 2A 20 20 20 0000027B'010E0000' 0273  .ASCID  /   ** /                    ; ...to check UETSUPDEV.DAT
                             0281
                             0281  UETCOMGS:                                   ; Constant part of interproc GS name
4D 4F 43 54 45 55 00000289'010E0000' 0281  .ASCID  /UETCOMGS/                  ; Must be able to accomodate unit
                      53 47  028F
                             0291
                             0291  UETSPRGS:                                   ; Constant part of single processor GS name
52 50 53 54 45 55 00000299'010E0000' 0291  .ASCID  /UETSPRGS/                  ; Must be able to accomodate unit
                      53 47  029F
                             02A1
                             02A1  UETMB:                                      ; Constant part of mailbox names
   42 4D 54 45 55 000002A9'010E0000' 02A1  .ASCID  /UETMB/                     ; Must be able to accomodate unit and port
                             02AE
                             02AE  UETCOMEF:                                   ; Constant part of interproc common EF
4D 4F 43 54 45 55 000002B6'010E0000' 02AE  .ASCID  /UETCOMEF/                  ; Must be able to accomodate unit
                      46 45  02BC
                             02BE
                             02BE  UETSPREF:                                   ; Constant part of single processor EF
52 50 53 54 45 55 000002C6'010E0000' 02BE  .ASCID  /UETSPREF/                  ; Must be able to accomodate unit
                      46 45  02CC
                             02CE
                             02CE  SYS$TEST:                                   ; Logical name for [SYSTEST] on system disk
45 54 24 53 59 53 000002D6'010E0000' 02CE  .ASCID  /SYS$TEST:/
                   3A 54 53  02DC
                             02DF
                             02DF  DOTDAT:                                     ; File type of files used for disk sections
      54 41 44 2E 000002E7'010E0000' 02DF  .ASCID  /.DAT/
                             02EB
                             02EB  ONE_SECOND:                                 ; 1-second delta time
          FFFFFFFF FF676980  02EB          .LONG   -10000000*1,-1
                             02F3
                             02F3  ALREADY_HERE:                               ; Possible consistency error message
61 63 69 64 6E 49 000002FB'010E0000' 02F3  .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  039E
            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
      79 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 75 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 73 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 6C  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
                                         0905  SBADCODE_MSG:                              ; Procs who read our date-time wrong
73 28 55 50 43 09 0000090D'010E0000'     0905          .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                  \!/             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                  \ section.\
                                         0978
                                         0978  DOSET_MSG:                                 ; Processors which didn't set a CEF
73 28 55 50 43 09 00000980'010E0000'     0978          .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                  \ common event flag.\
            2E 67 61 6C 66 20 74         09BC
                                         09C3
                                         09C3  DOCLEAR_MSG:                               ; Processors which didn't clear a CEF
73 28 55 50 43 09 000009CB'010E0000'     09C3          .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                  \ common event flag.\
            2E 67 61 6C 66 20 74         0A09
                                         0A10
                                         0A10  NOCAUSE_MSG:                               ; Msg from processors for no cause
73 28 55 50 43 09 00000A18'010E0000'     0A10          .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                  \!/             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
                                         0A72  OLDSTUFF_MSG:                              ; Msg from processor before we're ready
73 28 55 50 43 09 00000A7A'010E0000'     0A72          .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 72 6F 73 73      0AA4
20 78 6F 62 6C 69 61 6D 09 09 2F 21      0AAA                  \!/             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                  \ ready!/                to receive it.\
2E 74 69 20 65 76 69 65 63 65 72 20      0AE5
                                         0AF1
                                         0AF1  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 2F 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 65 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            .SBTTL  Read/Write Data
                             00000000            .PSECT  RWDATA,WRT,NOEXE,PAGE
                                 0000
                                 0000   TTCHAN:                                   ; Channel associated with ctrl. term.
                         0000    0000            .WORD   0
                                 0002
                                 0002   FLAG:                                     ; Miscellaneous flag bits
                         0000    0002            .WORD   0                        ; (See Equated Symbols for definitions)
                                 0004
                                 0004   FAO_BUF:                                  ; FAO output string descriptor
                    0000 00C8    0004            .WORD   TEXT_BUFFER,0
                    00000014'    0008            .ADDRESS BUFFER
                                 000C
                                 000C   BUFFER_PTR:                               ; Fake .ASCID buffer for misc. strings
                    0000 00C8    000C            .WORD   TEXT_BUFFER,0            ; A word for length, a word for desc.
                    00000014'    0010            .ADDRESS BUFFER
                                 0014
                                 0014   BUFFER:                                   ; FAO output and other misc. buffer
                     000000DC    0014            .BLKB   TEXT_BUFFER
                                 00DC
                                 00DC   DEVDSC:                                   ; Device name descriptor
                    0000 000A    00DC            .WORD   MAX_DEV_DESIG,0
                    000000FB'    00E0            .ADDRESS DEV_NAME
                                 00E4
                                 00E4   PROCESS_NAME:                             ; Process name
38 37 41 4D 000000EC'010E0000'   00E4            .ASCID  /MA78/
                     0000000B    00F0            PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
                     000000FB    00F0            .BLKB   PROCESS_NAME_FREE
                                 00FB
                                 00FB   DEV_NAME:                                 ; Device name buffer
                     0000010A    00FB            .BLKB   MAX_DEV_DESIG+MAX_UNIT_DESIG
                     0000000F    010A            NAME_LEN = .-DEV_NAME
                                 010A
                                 010A   DIB:                                      ; Device Information Block
                    0000 0074    010A            .WORD   DIB$K_LENGTH,0
                    00000112'    010E            .ADDRESS DIBBUF
                                 0112   DIBBUF:
                     00000186    0112            .BLKB   DIB$K_LENGTH
                                 0186
                                 0186   ERROR_COUNT:                              ; Cumulative error count at runtime
                     00000000    0186            .LONG   0
                                 018A
                                 018A   STATUS:                                   ; Status value on program exit
                     00000000    018A            .LONG   0
                                 018E
                                 018E   QUAD_STATUS:                              ; IO status block for misc sys. svcs.
            00000000 00000000    018E            .QUAD   0
                                 0196
                                 0196   INADDRESS:                                ; $CRMPSC address storage
            00000000 00000000    0196            .LONG   0,0
                                 019E   OUTADDRESS:
            00000000 00000000    019E            .LONG   0,0
                                 01A6
                                 01A6   UNIT_NUMBER:                              ; Current dev unit number
                         0000    01A6            .WORD   0
                                 01A8
                                 01A8   DEVNAM_LEN:                               ; Current device name length
```

```
                      0000  01A8          .WORD   0
                            01AA
                            01AA  RANDOM1:                              ; Random word #1
                  AAAAAAAA  01AA          .LONG   ^XAAAAAAAA
                            01AE
                            01AE  RANDOM2:                              ; Random word #2
                  A72EA72E  01AE          .LONG   ^XA72EA72E
                            01B2
                            01B2  ITERATION:                            ; # of times all tests were executed
                  00000000  01B2          .LONG   0
                            01B6
                            01B6  PASS:                                 ; Pass count
                  00000000  01B6          .LONG   0
                            01BA
                            01BA  MSG_BLOCK:                            ; Auxiliary $GETMSG info
                  000001BE  01BA          .BLKB   4
                            01BE
                            01BE  EXIT_DESC:                            ; Exit handler descriptor
                  00000000  01BE          .LONG   0
                  00002125' 01C2          .ADDRESS EXIT_HANDLER
                  00000001  01C6          .LONG   1
                  0000018A' 01CA          .ADDRESS STATUS
                            01CE
                            01CE  ARG_COUNT:                            ; Argument counter used by ERROR_EXIT
                  00000000  01CE          .LONG   0
                            01D2
                            01D2          .ALIGN QUAD                   ; Self-relative queue must be aligned
                            01D8  UNIT_LIST:                            ; Head of unit block circular queue
         00000000 00000000  01D8          .QUAD   0
                            01E0
                            01E0  SHM_INFO_ARGS:                        ; Arg list for $CMKRNL SHM_INFO
                  00000004  01E0          .LONG   4
                  000001A6' 01E4          .ADDRESS UNIT_NUMBER
                  000001F4  01E8          .BLKL   3
                            01F4
                            01F4  SCSNODE:                              ; The name of our node in a cluster
                  000001FA  01F4          .BLKB   6
                            01FA
                            01FA  SCSNODE_LENGTH:                       ; Length of our cluster node name
                  000001FE  01FA          .BLKL   1
                            01FE
                            01FE  NEW_NODE:                             ; Newly acquired node address
         00000000 00000000  01FE          .QUAD   0
                            0206
                            0206          .ALIGN  QUAD                  ; Needed for interlocked queue stuff
                            0208  COMGS_BUF:                            ; Buffer for creating interprocessor GS
                  000010B0  0208          .BLKB   GS_K_END
                            10B0
                            10B0  SYNCH_MBBUF:                          ; Buffer to create/receive MB during...
                  000010CF  10B0          .BLKB   MB_K_END              ; ...synchronous operations
                            10CF
                            10CF  SYNCH_MBCHN:                          ; Channel to access random MB during...
                  000010D1  10CF          .BLKW   1                     ; ...synchronous operations
                  000010D3  10D1          .BLKW   1                     ; Filler to make SS calls easier
                            10D3
                            10D3  SYNCH_MBISB:                          ; IOSB for synchronous QIO's to MBs
                  000010DB  10D3          .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        .SBTTL  RMS-32 Data Structures
          116C        .ALIGN  LONG
          116C
          116C  SYSIN_FAB:                                  ; Allocate FAB for SYS$INPUT
          116C        $FAB-
          116C        FNM = <SYS$INPUT>
          11BC
          11BC  SYSIN_RAB:                                  ; Allocate RAB for SYS$INPUT
          11BC        $RAB-
          11BC        FAB = SYSIN_FAB,-
          11BC        ROP = PMT,-
          11BC        PBF = PROMPT,-
          11BC        PSZ = PMTSIZ,-
          11BC        UBF = DEV_NAME,-
          11BC        USZ = NAME_LEN
          1200
          1200  INI_FAB:                                    ; Allocate FAB for UETINIDEV
          1200        $FAB-
          1200        FAC = <GET,PUT,UPD>,-
          1200        RAT = CR,-
          1200        SHR = <GET,PUT,UPI>,-
          1200        FNM = <UETINIDEV.DAT>
          1250
          1250  INI_RAB:                                    ; Allocate RAB for UETINIDEV
          1250        $RAB-
          1250        FAB = INI_FAB,-
          1250        RBF = BUFFER,-
          1250        UBF = BUFFER,-
          1250        USZ = REC_SIZE
          1294
          1294  DDB_RFA:                                    ; RFA storage for INI_RAB
0000129A  1294        .BLKB   6
          129A
          129A        .ALIGN  LONG
          129C  SUP_FAB:                                    ; Allocate FAB for UETSUPDEV
          129C        $FAB-
          129C        FAC = GET,-
          129C        SHR = <UPI,GET>,-
          129C        RAT = CR,-
          129C        FOP = UFO,-
          129C        FNM = <UETSUPDEV.DAT>
          12EC
          12EC  ;
          12EC  ; The dummy FABs and RABs defined below will be copied into their respective
          12EC  ; areas in the node set up for a particular shared memory.  A single MOVC is
          12EC  ; used for each FAB/RAB set, so the sets must remain with the RAB immediately
          12EC  ; following the FAB.
          12EC  ;
          12EC  COMGSFAB:                                   ; Dummy FAB used for interproc GS
          12EC        $FAB-                                 ; FAB for interproc GS is in MA_K_COMFAB
          12EC        ALQ = COMGS_SIZE,-
          12EC        FAC = <BIO,PUT>,-
          12EC        FOP = SUP,-
          12EC        MRS = GS_K_END,-
          12EC        RFM = FIX
          133C
          133C  COMGSRAB:                                   ; Dummy RAB used for interproc GS
```

```
133C                 $RAB-                          ; RAB for interproc GS is in MA_K_COMRAB
133C                 RAC = SEQ,-
133C                 RBF = COMGS_BUF,-
133C                 ROP = BIO,-
133C                 RSZ = GS_K_END
1380
1380    SPRGSFAB:                                   ; Dummy FAB used for single processor GS
1380                 $FAB-                          ; FAB for single proc GS is MA_K_SPRFAB
1380                 ALQ = SPRGS_SIZE,-
1380                 FAC = <BIO,PUT>,-
1380                 FOP = SUP,-
1380                 MRS = WRITE_SIZE,-
1380                 RFM = FIX
13D0
13D0    SPRGSRAB:                                   ; Dummy RAB used for single processor GS
13D0                 $RAB-                          ; RAB for single proc GS is MA_K_SPRRAB
13D0                 RAC = SEQ,-
13D0                 ROP = BIO,-
13D0                 RSZ = WRITE_SIZE
1414
1414    CHKGSFAB:                                   ; Dummy FAB used to check single proc GS
1414                 $FAB-                          ; FAB to check single proc GS is MA_K_CHKFAB
1414                 ALQ = SPRGS_SIZE,-
1414                 FAC = <BIO,GET>,-
1414                 FOP = SUP,-
1414                 MRS = WRITE_SIZE,-
1414                 RFM = FIX
1464
1464    CHKGSRAB:                                   ; Dummy RAB used to check single proc GS
1464                 $RAB-                          ; RAB to check single proc GS is MA_K_CHKRAB
1464                 RAC = SEQ,-
1464                 ROP = BIO,-
1464                 USZ = WRITE_SIZE
```

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

```
                   50    51   C2  00C0              SUBL2   R1,R0                          ; Overwrite handle otherwise...
              00E4'CF    0F   B0  00C3              MOVW    #MAX_PROC_NAME,PROCESS_NAME ; ...and define the maximum length
                                  00C8       10$:
              80    5F 8F   90  00C8              MOVB    #^A/ /,(R0)+                   ; Separate handle from device name
    60   00FB'CF   00DC'CF   28  00CC              MOVC3   DEVDSC,DEV_NAME,(R0)           ; Concatenate handle with device name
                        7E   D4  00D4              CLRL    -(SP)                          ; Set the time stamp flag
                   000F'CF   DF  00D6              PUSHAL  TEST_NAME                      ; Set the test name
                        02   DD  00DA              PUSHL   #2                             ; Push the argument count
               00741039 8F   DD  00DC              PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
            00000000'GF 04   FB  00E2              CALLS   #4,G^LIB$SIGNAL               ; Print the startup message
              0002'CF   08   A8  00E9              BISW2   #BEGIN_MSGM,FLAG              ; Set flag so we don't print it again
                                  00EE              $SETPRN_S PRCNAM = PROCESS_NAME ; Set the process name to UETMA7800_x
                                  00F9
                        02   E1  00F9              BBC     S^#DEV$V_TRM,-                 ; BR if SYS$INPUT is NOT a terminal
              66 11AC'CF          00FB                      SYSIN_FAB+FAB$L_DEV,20$
                                  00FF              $GETDVI_S DEVNAM = SYS$INPUT,-     ; Get the name of...
                                  00FF                      EFN    = #SS_SYNCH_EFN,- ; ...device which may abort test
                                  00FF                      ITMLST = INPUT_ITMLST,-
                                  00FF                      IOSB   = QUAD_STATUS
              45 018E'CF     E9  011B              BLBC    QUAD_STATUS,20$               ; Avoid CTRL/C handler if any error
                                  0120              $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handler
                                  0120                      CHAN   = TTCHAN
                                  0131              $QIOW_S CHAN     = TTCHAN,-         ; Enable CTRL/C AST's...
                                  0131                      FUNC     = #IO$_SETMODE!IO$M_CTRLCAST,-
                                  0131                      P1       = CCASTHAND
                   00E4'CF   DF  0152              PUSHAL  PROCESS_NAME                   ; ...and tell the user...
                        01   DD  0156              PUSHL   #1                             ; ...
               0074832B 8F   DD  0158              PUSHL   #UETP$_ABORTC!STS$K_SUCCESS ; ...how to abort gracefully...
            00000000'GF 03   FB  015E              CALLS   #3,G^LIB$SIGNAL               ; ...
                                  0165       20$:
                                  0165              $TRNLOG_S LOGNAM = MODE,-          ; Get the run mode
                                  0165                      RSLLEN = BUFFER_PTR,-
                                  0165                      RSLBUF = FAO_BUF
              0014'CF   20   8A  017E              BICB2   #LC_BITM,BUFFER               ; Convert to upper case
         0014'CF    4F 8F   91  0183              CMPB    #^A70/,BUFFER                  ; Is this a one shot?
                        05   12  0189              BNEQ    25$                            ; BR if not
              0002'CF   12   A8  018B              BISW2   #ONE_SHOTM!TEST_OVERM,FLAG ; Set flags for one-shot mode
                                  0190       25$:
```

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

```
                                0290   ;
                                0290   ; Check here (only once) to see if we're the correct test for the
                                0290   ; device we were given to check.  Note that this section differs substantially
                                0290   ; from the standard UETP Device Test Template because there is no way to get
                                0290   ; the "device characteristics" of a multiport memory.
                                0290   ;
                                0290   20$:
   00  00FB'CF  00DC'CF   2D    0290           CMPC5   DEVDSC,DEV_NAME,#0,-      ; Match controller name vs. "MPM"
       0270'CF  0268'CF         0298                   MPM_LITERAL,MPM_LITERAL+8
                     1A   12    029E           BNEQ    30$                       ; BR if another device was given
   56  027B'CF  0273'CF   39    02A0           MATCHC  CS2,CS2+8,R6,@OUTADDRESS  ; Find our line in UETSUPDEV.DAT
                019E'DF         02A8
                     0D   12    02AB           BNEQ    30$                       ; BR if we're not there
            55  000F'CF   9A    02AD           MOVZBL  TEST_NAME,R5              ; Get the test name length
       0017'CF    63  55  29    02B2           CMPC3   R5,(R3),TEST_NAME+8       ; Are we the right test?
                     1F   13    02B8           BEQL    40$                       ; BR if we are
                                02BA   30$:
                00DC'CF   DF    02BA           PUSHAL  DEVDSC                    ; Push device not supported message
                00E4'CF   DF    02BE           PUSHAL  PROCESS_NAME              ; Parameters on the stack
                     02   DD    02C2           PUSHL   #2                        ; Push the argument count
            00748333 8F   DD    02C4           PUSHL   #UETP$_DENOSU
                     02   F0    02CA           INSV    #STS$K_ERROR,-            ; Set the severi~~ code...
                     00         02CC                   #STS$V_SEVERITY,-
               6E    03         02CD                   #STS$S_SEVERITY,(SP)
         018A'CF     6E   D0    02CF           MOVL    (SP),STATUS               ; ...and save it as the exit status
                     04   DD    02D4           PUSHL   #4                        ; Push the partial arg count...
                   1DC7   31    02D6           BRW     ERROR_EXIT                ; ...and split this scene
                                02D9
                                02D9   40$:
                                02D9           $GETSYIW_S  EFN = #SS_SYNCH_EFN,-  ; Get our processor's ECO level...
                                02D9                   ITMLST = GETSYI_ITMLST,-  ; ...the version of VMS and...
                                02D9                   IOSB   = QUAD_STATUS      ; ...our SCS node name
       01F4'CF    06  20  3A    02F0           LOCC    #^A/ /,#6,SCSNODE         ; Remove the blanks...
       01FA'CF    06  50  C3    02F6           SUBL3   R0,#6,SCSNODE_LENGTH      ; ...that pad out the node name
                                02FC
                                02FC   FOUND_IT:
                                02FC           $GET    RAB = INI_RAB,-           ; Get a record
                                02FC                   ERR = RMS_ERROR
                01CC'CF   DF    030B           PUSHAL  CONT_DESC                 ; Make sure...
                01CC'CF   DF    030F           PUSHAL  CONT_DESC                 ; ...that this line...
          00000000'GF  02 FB    0313           CALLS   #2,G^STR$UPCASE           ; ...is all uppercase letters
        0014'CF   55  8F  91    031A           CMPB    #^A/U/,BUFFER             ; Is this a UCB?
                     24   13    0320           BEQL    30$                       ; BR if it is
        0014'CF   44  8F  91    0322           CMPB    #^A/D/,BUFFER             ; Is this a DDB?
                     19   13    0328           BEQL    20$                       ; BR if yes
        0014'CF   45  8F  91    032A           CMPB    #^A/E/,BUFFER             ; Is this the end?
                     11   13    0330           BEQL    20$                       ; BR if yes
                                0332   10$:
                0130'CF   DF    0332           PUSHAL  ILLEGAL_REC               ; Then this is an error in the record
                     01   DD    0336           PUSHL   #1                        ; Push the error message
            00741132 8F   DD    0338           PUSHL   #UETP$_TEXT!STS$K_ERROR   ; Push the signal name
                     03   DD    033E           PUSHL   #3                        ; Push the temp arg count
                   1D5D   31    0340           BRW     ERROR_EXIT                ; Finish for good
                                0343   20$:
                   02C6   31    0343           BRW     ALL_SET                   ; Found DDB or END
                                0346   30$:
        0018'CF   54  8F  91    0346           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  ;
                                0364  ; We've found a valid memory to test.  Allocate space to store unit-specific
                                0364  ; information and initialize the information.  Interprocessor data structures
                                0364  ; are unique between memories; single processor data structures are unique
                                0364  ; both between memories and within a shared memory.  See the FT_NO_INTERACT
                                0364  ; definition for a discussion of the use of OTHER_CHARS.
                                0364  ;
                                0364  60$:
                                0364          $EXPREG_S PAGCNT = #PAGES,-      ; Get a new node of demand zero memory
                                0364                  RETADR = NEW_NODE
     01D8'CF   01FE'DF    5D    0375          INSQTI  @NEW_NODE,UNIT_LIST      ; Put the new node in the unit list
          56   01FE'CF    D0    037C          MOVL    NEW_NODE,R6              ; Save a copy of its address
          08 A6     01    90    0381          MOVB    #1,UETUNT$B_TYPE(R6)     ; Set the structure type
               064C 8F    B0    0385          MOVW    #UETUNT$C_INDSIZ+DEVDEP_SIZE,-
                  09 A6         0389                  UETUNT$W_SIZE(R6)        ; Set the structure size
                                038B          $GETTIM_S TIMADR = MA_Q_DAYTIME(R6) ; Save time as interprocessor code
     01EC C6   01A6'CF    B0    0396          MOVW    UNIT_NUMBER,MA_W_UNIT(R6)        ; Given memory unit number...
     01E8'CF   01AC C6    9E    039D          MOVAB   MA_B_MYPORT(R6),SHM_INFO_ARGS+08 ; ...find my port on it,...
     01EC'CF   01F9 C6    9E    03A4          MOVAB   MA_T_MEMNAM(R6),SHM_INFO_ARGS+12 ; ...the memory name...
     01F0'CF   01F1 C6    7E    03AB          MOVAQ   MA_Q_MEMNAM(R6),SHM_INFO_ARGS+16 ; ...and the name length...
                                03B2          $CMKRNL_S ROUTIN = SHM_INFO,-    ; ...
                                03B2                  ARGLST = SHM_INFO_ARGS
               01F9 C6    9E    03C1          MOVAB   MA_T_MEMNAM(R6),-        ; Finish .ASCID pointer to memory name
               01F5 C6          03C5                  MA_Q_MEMNAM+4(R6)
                    02    88    03C8          BISB2   #UETUNT$M_TESTABLE,-     ; Assume for now that MPM is testable
                 0B A6          03CA                  UETUNT$B_FLAGS(R6)
                                03CC  ; Note that the unit number is treated as a word above, but as a byte
                                03CC  ; forevermore.  (The current hardware uses a 2-bit unit number.)
          50   01EC C6    90    03CC          MOVB    MA_W_UNIT(R6),R0         ; Get hex ASCII unit number...
                  0E5D    30    03D1          BSBW    HEXUNIT                  ; ...
          01F0 C6   50    90    03D4          MOVB    R0,MA_T_UNIT(R6)         ; Put it away for safekeeping...
               59   50    90    03D9          MOVB    R0,R9                    ; ...and for convenience
                                03DC
                                03DC
               01F1 C6    28    03DC          MOVC3   MA_Q_MEMNAM(R6),-        ; Interproc GS name is memory-name...
     0210 C6   01F9 C6          03E0                  MA_T_MEMNAM(R6),MA_T_COMGSNAM(R6)
               83   3A    90    03E6          MOVB    #^A/:/,(R3)+             ; ...colon...
  63 0289'CF   0281'CF    28    03E9          MOVC3   UETCOMGS,UETCOMGS+8,(R3) ; ...section-name...
               83   59    90    03F1          MOVB    R9,(R3)+                 ; ...and unit to make it unique
  63    01AC C6     30    81    03F4          ADDB3   #^A/0/,MA_B_MYPORT(R6),(R3) ; Append port in case FT_NO_INTERACT
  57 0281'CF   01F1 C6    81    03FA          ADDB3   MA_Q_MEMNAM(R6),UETCOMGS,R7 ; Figure total name length...
     0208 C6   57   02    81    0402          ADDB3   #OTHER_CHARS,R7,-        ; ...
                                0408                  MA_Q_COMGSNAM(R6)
               0210 C6    DE    0408          MOVAL   MA_T_COMGSNAM(R6),-      ; Finish .ASCID pointer
               020C C6          040C                  MA_Q_COMGSNAM+4(R6)
```

```
                 01F1 C6    28  040F      MOVC3   MA_Q_MEMNAM(R6),-               ; Single proc GS name is memory-name...
       0287 C6   01F9 C6        0413              MA_T_MEMNAM(R6),MA_T_SPRGSNAM(R6)
                 83   3A    90  0419      MOVB    #^A/:/,(R3)+                    ; ...colon...
   63  0299'CF   0291'CF    28  041C      MOVC3   UETSPRGS,UETSPRGS+8,(R3)        ; ...section-name...
                 83   59    90  0424      MOVB    R9,(R3)+                        ; ...unit...
   63  01AC C6        30    81  0427      ADDB3   #^A/0/,MA_B_MYPORT(R6),(R3)     ; ...and port to make it unique
   57  0291'CF   01F1 C6    81  042D      ADDB3   MA_Q_MEMNAM(R6),UETSPRGS,R7     ; Figure total name length...
       027F C6   57   03    81  0435      ADDB3   #3,R7,MA_Q_SPRGSNAM(R6)         ; ...
                 0287 C6    DE  043B      MOVAL   MA_T_SPRGSNAM(R6),-             ; Finish .ASCID pointer
                 0283 C6        043F              MA_Q_SPRGSNAM+4(R6)
                                0442
                                0442
                 01F1 C6    28  0442      MOVC3   MA_Q_MEMNAM(R6),-               ; Interproc CEF name is memory-name...
       0237 C6   01F9 C6        0446              MA_T_MEMNAM(R6),MA_T_COMEFNAM(R6)
                 83   3A    90  044C      MOVB    #^A/:/,(R3)+                    ; ...colon...
   63  02B6'CF   02AE'CF    28  044F      MOVC3   UETCOMEF,UETCOMEF+8,(R3)        ; ...cluster-name...
                 83   59    90  0457      MOVB    R9,(R3)+                        ; ...and unit to make it unique
   63  01AC C6        30    81  045A      ADDB3   #^A/0/,MA_B_MYPORT(R6),(R3)     ; Append port in case FT_NO_INTERACT
   57  02AE'CF   01F1 C6    81  0460      ADDB3   MA_Q_MEMNAM(R6),UETCOMEF,R7     ; Figure total name length...
       022F C6   57   02    81  0468      ADDB3   #OTHER_CHARS,R7,-               ; ...
                                046E              MA_Q_COMEFNAM(R6)
                 0237 C6    DE  046E      MOVAL   MA_T_COMEFNAM(R6),-             ; Finish .ASCID pointer
                 0233 C6        0472              MA_Q_COMEFNAM+4(R6)
                                0475
                                0475
                 01F1 C6    28  0475      MOVC3   MA_Q_MEMNAM(R6),-               ; Single proc CEF name is memory-name...
       02AE C6   01F9 C6        0479              MA_T_MEMNAM(R6),MA_T_SPREFNAM(R6)
                 83   3A    90  047F      MOVB    #^A/:/,(R3)+                    ; ...colon...
   63  02C6'CF   02BE'CF    28  0482      MOVC3   UETSPREF,UETSPREF+8,(R3)        ; ...cluster-name...
                 83   59    90  048A      MOVB    R9,(R3)+                        ; ...unit...
   63  01AC C6        30    81  048D      ADDB3   #^A/0/,MA_B_MYPORT(R6),(R3)     ; ...and port to make it unique
   57  02BE'CF   01F1 C6    81  0493      ADDB3   MA_Q_MEMNAM(R6),UETSPREF,R7     ; Figure total name length...
       02A6 C6   57   03    81  049B      ADDB3   #3,R7,MA_Q_SPREFNAM(R6)         ; ...
                 02AE C6    DE  04A1      MOVAL   MA_T_SPREFNAM(R6),-             ; Finish .ASCID pointer
                 02AA C6        04A5              MA_Q_SPREFNAM+4(R6)
```

```
                 01F1 C6   28 04A8 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$B_FNS(R7)    ; ...
                    14 A6  DE 054E 1354   MOVAL   MA_T_COMFAB(R6),-          ; Set the FNA field in the FAB
                    2C A7     0551 1355           FAB$L_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   0291'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 unit 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               COMGS_BUF+GS_L_OWNER,-
57     000008B0'8F      0690               #COMGS_BUF+GS_R_PVT_AREAS,R7
             51  57  D0 0696       MOVL    R7,R1
50     0000007E 8F  D0 0699       MOVL    #COMGS_PVT_CODE/4,R0     ; ...and fill it...
              0B9A  30 06A0       BSBW    RANDOM_FILL              ; ...with random trash
           01A4 C6  7D 06A3       MOVQ    MA_Q_DAYTIME(R6),-       ; Set the special message, used as...
           01F8 C7      06A7               COMGS_PVT_CODE(R7)       ; ...a code to identify my processor
                        06AA
                        06AA       $CREATE  FAB = MA_K_COMFAB(R6),- ; Set up file for interprocessor GS
                        06AA                ERR = RMS_ERROR
                        06B9       $CONNECT RAB = MA_R_COMRAB(R6),-
                        06B9                ERR = RMS_ERROR
                        06C8       $WRITE   RAB = MA_R_COMRAB(R6),-
                        06C8                ERR = RMS_ERROR
                        06D7       $CLOSE   FAB = MA_R_COMFAB(R6),-
                        06D7                ERR = RMS_ERROR
      00020000 8F   C8 06E6       BISL2   #FAB$M_UFO,-             ; Turn on the UFO bit...
          0114 C6       06EC               MA_K_COMFAB+FAB$L_FOP(R6) ; ...so RMS sets us up OK later
                        06EF
                        06EF       $CREATE  FAB = MA_K_SPRFAB(R6),- ; Set up file for single processor GS
                        06EF                ERR = RMS_ERROR
                        06FE       $CONNECT RAB = MA_R_SPRRAB(R6),-
                        06FE                ERR = RMS_ERROR
                        070D       $WRITE   RAB = MA_R_SPRRAB(R6),-
                        070D                ERR = RMS_ERROR
                        071C       $CLOSE   FAB = MA_R_SPRFAB(R6),-
                        071C                ERR = RMS_ERROR
      00020000 8F   C8 072B       BISL2   #FAB$M_UFO,-             ; Turn on the UFO bit...
          0400 C6       0731               MA_K_SPRFAB+FAB$L_FOP(R6) ; ...so RMS sets us up OK later
                        0734
                        0734       $CREMBX_S CHAN   = MA_W_SPRMBCHN(R6),- ; Create my mailbox
                        0734                 LOGNAM = MA_Q_SPRMBNAM(R6)
                        074B       $QIO_S   CHAN   = MA_W_SPRMBCHN(R6),- ; Be ready to read if...
                        074B                FUNC   = #IO$_READVBLK,-     ; ...someone writes...
                        074B                IOSB   = AST_MBISB,-         ; ...to my mailbox
                        074B                ASTADR = RCVMB_AST,-
                        074B                ASTPRM = R6,-
                        074B                P1     = AST_MBBUF,-
                        074B                P2     = #MB_K_END
                        0772
          56  66   C0 0772       ADDL2   (R6),R6                  ; Point to node for next memory
56     000001D8'8F  D1 0775       CMPL    #UNIT_LIST,R6            ; Back at the start of the queue?
              03   13 077C       BEQL    30$                      ; BR if we are - we're finished
            FEEB   31 077E       BRW     20$                      ; More to go so loop
                        0781  30$:
  13 0002'CF   04  E0 0781       BBS     #ONE_SHOTV,FLAG,RESTART  ; In one-shot mode start testing now...
                        0787                                        ; ...else fall into TIME_IT
                        0787  TIME_IT:
                        0787       $SETIMR_S DAYTIM = THREEMIN,-    ; Set timer AST to 3 minutes
                        0787                 ASTADR = TIME_OUT,-
                        0787                 EFN    = #EFN2
```

```
                          079A          .SBTTL  Test the Multiport Memory
                          079A
                          079A  RESTART:
                          079A  ;+
                          079A  ;       Start the normal test by mapping to the interprocessor global section
                          079A  ;       and announcing our presence.
                          079A  ;-
                          079A
      03 0002'CF    04  E1   079A       BBC     #ONE_SHOTV,FLAG,10$      ; Skip all interprocessor stuff...
                  00E7  31   07A0       BRW     ONEPROC_GS               ; ...if we are in one-shot mode
                          07A3  10$:
  56  01D8'CF  000001D8'8F  C1  07A3    ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; R6 points to node for current memory
                          07AD  20$:
                          07AD
                          07AD  ; Map to the interprocessor global section for this memory.  If it turns out we
                          07AD  ; also create it, we're ready to handle messages from other processors which
                          07AD  ; may subsequently map to it.
                          07AD  ;
                          07AD          $OPEN   FAB = MA_K_COMFAB(R6),- ; Reopen its interprocessor GS file
                          07AD                  ERR = RMS_ERROR
        0624 C6     01  D0   07BC       MOVL    #1,MA_Q_COMGSIAD(R6)    ; Say that file maps to P0 space
                          07C1          $CRMPSC_S INADR = MA_Q_COMGSIAD(R6),- ; Map that file to a GS
                          07C1                  RETADR = MA_Q_COMGSRAD(R6),-
                          07C1                  FLAGS  = #SEC$M_GBL!SEC$M_EXPREG!SEC$M_WRT!SEC$M_PERM,-
                          07C1                  PAGCNT = #<GS_K_END+511>/512,-
                          07C1                  GSDNAM = MA_Q_COMGSNAM(R6),-
                          07C1                  CHAN   = MA_K_COMFAB+FAB$L_STV(R6)
        57   062C C6    D0   07EB       MOVL    MA_Q_COMGSRAD(R6),R7    ; Save GS's starting address in our P0
   50    00000619 8F    D1   07F0       CMPL    #SS$_CREATED,R0         ; Did we create-and-map or just map?
                    03  12   07F7       BNEQ    30$
                  007F  31   07F9       BRW     90$                     ; BR if we created
                          07FC  30$:
                          07FC  ;
                          07FC  ; We've mapped to an existing section.  Check that we don't already appear in
                          07FC  ; the mask of cooperating processors.  Consistency check that we've got a good
                          07FC  ; section.  Check for matching ECO levels of our CPU versus the CPU which
                          07FC  ; created the global section.  Set up our private area and announce us to the
                          07FC  ; world.
                          07FC  ;
               01AC C6  E7   07FC       BBCCI   MA_B_MYPORT(R6),-       ; BR if we're not already here
              08 24 A7       0800               GS_L_AVAILABLE(R7),40$
        50    02F3'CF   DE   0803       MOVAL   ALREADY_HERE,R0
                  0A7B  30   0808       BSBW    DO_FAO_SIGNAL           ; Warn of possible consistency problems
                          080B  40$:
   0289'CF    0281'CF   29   080B       CMPC3   UETCOMGS,UETCOMGS+8,-   ; Did we find the correct...
                 01 A7       0812               GS_T_SECT_NAME+1(R7)    ; ...global section name?
                    11  12   0814       BNEQ    50$                     ; BR if not
        63    01F0 C6   91   0816       CMPB    MA_T_UNIT(R6),(R3)      ; More checking of GS name
                    0A  12   081B       BNEQ    50$                     ; Again, BR if not as expected
  10 A7   30390004 8F   D1   081D       CMPL    #CONSIS_CHECK,GS_L_VERSION(R7) ; Version and max ports match?
                    08  13   0825       BEQL    60$                     ; BR if they do - we're consistent
                          0827  50$:
        50    036C'CF   DE   0827       MOVAL   INCONSISTENCY,R0
                  0A77  30   082C       BSBW    DO_FAO_EXIT             ; We got the wrong GS - die
                          082F  60$:
              0224'CF   D1   082F       CMPL    COMGS_BUF+GS_L_ECOLEVEL,- ; Do we have the same CPU...
                 1C A7       0833               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$:
     58  06A8 C7  DE 0850  1575         MOVAL   GS_K_PVT_AREAS(R7),R8
     00000200 8F  7A 0855  1576         EMUL    #COMGS_PVT_AREA,-                ; Point to start of my private area...
58   58  01AC C6     085B  1577                 MA_B_MYPORT(R6),R8,R8
           51 58  D0 0860  1578         MOVL    R8,R1
50   0000007E 8F  D0 0863  1579         MOVL    #COMGS_PVT_CODE/4,R0             ; ...and fill it...
            09D0  30 086A  1580         BSBW    RANDOM_FILL                      ; ...with random trash
         01A4 C6  7D 086D  1581         MOVQ    MA_Q_DAYTIME(R6),-               ; Set the special message, used as...
         01F8 C8     0871  1582                 COMGS_PVT_CODE(R8)               ; ...a code to identify my processor
         01AC C6  E6 0874  1583         BBSSI   MA_B_MYPORT(R6),-                ; Let other processors know of us!
        00 24 A7     0878  1584                 GS_L_AVAILABLE(R7),90$
                     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   ONEPROC_GS:
                               088A   ;+
                               088A   ;       Play with the single processor global section.  Create and map to a
                               088A   ;       global section in each shared memory which is used exclusively by our
                               088A   ;       processor.  Modify the contents and update each section back to disk.
                               088A   ;       At AST level, confirm that the updates succeeded and delete each
                               088A   ;       section and its address space.
                               088A   ;-
                               088A
 56  01D8'CF  000001D8'8F  C1  088A           ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; R6 points to node for current memory
                               0894   10$:
                               0894   ;
                               0894   ; Map to the single processor global section in this memory.  If we only map,
                               0894   ; but not create it, give a warning.  To prevent possible conflicts with
                               0894   ; previous passes through this code, make sure that the $UPDSEC-in-progress
                               0894   ; flag is set to allow us to proceed.
                               0894   ;
                               0894           $WAITFR_S EFN = #SPRGS_EFN      ; Allow possible $UPDSEC to finish
                               089D           $OPEN   FAB = MA_K_SPRFAB(R6),- ; Reopen single processor GS file
                               089D                   ERR = RMS_ERROR
          0634 C6    01    D0  08AC           MOVL    #1,MA_Q_SPRGSIAD(R6)    ; Say that file maps to P0 space
                               08B1           $CRMPSC_S INADR = MA_Q_SPRGSIAD(R6),- ; Map that file to a GS
                               08B1                   RETADR = MA_Q_SPRGSRAD(R6),-
                               08B1                   FLAGS  = #SEC$M_GBL!SEC$M_EXPREG!SEC$M_WRT!SEC$M_PERM,-
                               08B1                   PAGCNT = #SPRGS_SIZE,-
                               08B1                   GSDNAM = MA_Q_SPRGSNAM(R6),-
                               08B1                   CHAN   = MA_K_SPRFAB+FAB$L_STV(R6)
        57    063C C6      D0  08DB           MOVL    MA_Q_SPRGSRAD(R6),R7    ; Save GS's starting address in our P0
   50    00000619 8F       D1  08E0           CMPL    #SS$_CREATED,R0         ; Did we create-and-map or just map?
                    08     13  08E7           BEQL    20$                     ; BR if we created
        50    03B2'CF      DE  08E9           MOVAL   OLD_ONEPROC,R0
                  0995     30  08EE           BSBW    DO_FAO_SIGNAL           ; We've a rusty old single proc GS - warn
                               08F1   20$:
                               08F1   ;
                               08F1   ; Use a bufferful of pseudo-random data as a means to verify that the section
                               08F1   ; can be written to correctly and then updated correctly to disk.  Only issue
                               08F1   ; the $UPDSEC here; the actual verification will take place at AST level.  That
                               08F1   ; will allow us to test the next memory or otherwise proceed with the test.
                               08F1   ;
        51    064C C6      DE  08F1           MOVAL   MA_K_SPRBUF(R6),R1      ; Fill a buffer...
   50    00000800 8F       D0  08F6           MOVL    #SPRGS_SIZE*128,R0      ; ...of longwords...
                  093D     30  08FD           BSBW    RANDOM_FILL             ; ...with random trash
               2000 8F     28  0900           MOVC3   #SPRGS_SIZE*512,-       ; Move that trash to the...
        67    064C C6          0904                   MA_K_SPRBUF(R6),(R7)    ; ...single processor global section
               2000 8F     29  0908           CMPC3   #SPRGS_SIZE*512,-       ; Make sure that the pages...
        67    064C C6          090C                   MA_K_SPRBUF(R6),(R7)    ; ...all get updated
                    49     13  0910           BEQL    30$                     ; BR if all data transferred
                    02     8A  0912           BICB2   #UETUNT$M_TESTABLE,-    ; The MPM is not testable...
                 0B A6         0914                   UETUNT$B_FLAGS(R6)      ; ...(as far as one-shot mode thinks)
              7E    63     9A  0916           MOVZBL  (R3),-(SP)              ; Not written OK. Save the bad byte...
              7E    61     9A  0919           MOVZBL  (R1),-(SP)              ; ...the corresponding good byte...
        7E    53    57     C3  091C           SUBL3   R7,R3,-(SP)             ; ...where in the buffer it was...
               01F1 C6     DF  0920           PUSHAL  MA_Q_MEMNAM(R6)         ; ...the memory name...
                    04     DD  0924           PUSHL   #4                      ; ...the argument count...
           0074801A 8F     DD  0926           PUSHL   #UETP$_DATADEVERR!STS$K_ERROR ; ...and the error type
               0402'CF     DF  092C           PUSHAL  BAD_MOVC3               ; Tell what the error was from
           000F0001 8F     DD  0930           PUSHL   #^XF0001
```

```
        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
                FF0D  31  0984  1666          BRW     10$                          ; More to go so loop
                          0987  1667 40$:
```

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

```
                                      0A4E  ONEPROC_EF:
                                      0A4E  ;+
                                      0A4E  ;       Play with the single processor common event flag cluster.  Associate
                                      0A4E  ;       with a cluster in each shared memory and test to see that all bits in
                                      0A4E  ;       each can be set and cleared when used exclusively by our processor.
                                      0A4E  ;       Disassociate from the clusters.
                                      0A4E  ;-
                                      0A4E
56  01D8'CF   000001D8'8F   C1        0A4E          ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; R6 points to node for current memory
                                      0A58  10$:
                                      0A58          $ASCEFC_S EFN = #SPREF_EFN,-    ; Associate with a CEF cluster
                                      0A58                  NAME = MA_Q_SPREFNAM(R6)
                                      0A6D  ;
                                      0A6D  ; Set and clear each flag in sequence.
                                      0A6D  ;
          57  00000040 8F   D0        0A6D          MOVL    #SPREF_EFN,R7           ; Initialize EF position counter
                    58  01  D0        0A74          MOVL    #1,R8                   ; Initialize EF mask
                                      0A77  20$:
                                      0A77          $SETEF_S  EFN   = R7            ; Set a common event flag
                                      0A80          $READEF_S EFN   = R7,-          ; Read back the cluster
                                      0A80                  STATE = MA_L_SPREFSTATE(R6)
                  00AA      30        0A8D          BSBW    100$                    ; Check that exactly our flags are set
                                      0A90          $CLREF_S  EFN   = R7            ; Clear what we just set
                                      0A99          $READEF_S EFN   = R7,-          ; Read back the cluster
                                      0A99                  STATE = MA_L_SPREFSTATE(R6)
                  009D      30        0AA6          BSBW    200$                    ; Check that our flag is clear
                    57      D6        0AA9          INCL    R7                      ; Point to the next flag in the cluster
          58    58  01      78        0AAB          ASHL    #1,R8,R8                ; Mask the next flag in the cluster
                    C6      12        0AAF          BNEQ    20$                     ; BR if there are more bits to test
                                      0AB1  ;
                                      0AB1  ; Set each flag, one at a time, until the entire cluster is set.
                                      0AB1  ;
          57  00000040 8F   D0        0AB1          MOVL    #SPREF_EFN,R7           ; Initialize EF position counter
                    58      D4        0AB8          CLRL    R8                      ; Initialize total EF mask
                59  01      D0        0ABA          MOVL    #1,R9                   ; Initialize individual EF mask
                                      0ABD  30$:
                58  59      C8        0ABD          BISL2   R9,R8                   ; Mask next flag to set in the cluster
                                      0AC0          $SETEF_S  EFN   = R7            ; Set a common event flag
                                      0AC9          $READEF_S EFN   = R7,-          ; Read back the cluster
                                      0AC9                  STATE = MA_L_SPREFSTATE(R6)
                  0061      30        0AD6          BSBW    100$                    ; Check that exactly our flags are set
                    57      D6        0AD9          INCL    R7                      ; Point to the next flag in the cluster
          59    59  01      78        0ADB          ASHL    #1,R9,R9                ; Mask the next flag in the cluster
                    DC      12        0ADF          BNEQ    30$                     ; BR if there are more bits to test
                                      0AE1  ;
                                      0AE1  ; Clear each flag, one at a time, until the entire cluster is clear.
                                      0AE1  ;
          57  00000040 8F   D0        0AE1          MOVL    #SPREF_EFN,R7           ; Initialize EF position counter
                58  00      D2        0AE8          MCOML   #0,R8                   ; Initialize total EF mask
                59  01      D0        0AEB          MOVL    #1,R9                   ; Initialize individual EF mask
                                      0AEE  40$:
                58  59      CA        0AEE          BICL2   R9,R8                   ; Mask all flags set in the cluster
                                      0AF1          $CLREF_S  EFN   = R7            ; Clear a common event flag
                                      0AFA          $READEF_S EFN   = R7,-          ; Read back the cluster
                                      0AFA                  STATE = MA_L_SPREFSTATE(R6)
                  0030      30        0B07          BSBW    100$                    ; Check that exactly our flags are clear
                    57      D6        0B0A          INCL    R7                      ; Point to the next flag in the cluster
```

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

```
                          0B3A  ;
                          0B3A  ; Subroutine to check that R8 matches exactly the common event flags in our
                          0B3A  ; cluster.
                          0B3A  ;
                          0B3A  100$:
            54   58   D0  0B3A          MOVL    R8,R4                     ; Set up if error message needed
5A   58   02CD C6   CD  0B3D          XORL3   MA_L_SPREFSTATE(R6),R8,R10 ; Exact match?
                 11   12  0B43          BNEQ    300$                      ; BR if not - issue error message
                      05  0B45          RSB
                          0B46
                          0B46
                          0B46  ;
                          0B46  ; Subroutine to check that the flags masked in R8 are clear.
                          0B46  ;
                          0B46  200$:
                 54   D4  0B46          CLRL    R4                        ; Set up if error message needed...
5A   58   02CD C6   CD  0B48          XORL3   MA_L_SPREFSTATE(R6),R8,R10 ; ...
     02CD C6     58   D3  0B4E          BITL    R8,MA_L_SPREFSTATE(R6)    ; Is requested bit clear?
                 01   12  0B53          BNEQ    300$                      ; BR if not - issue error message
                      05  0B55          RSB
                          0B56
                          0B56
                          0B56  ;
                          0B56  ; Subroutine which, given expected flags to be set in R4, actual flags set in
                          0B56  ; MA_L_SPREFSTATE(R6) and the difference between the two in R10, prints an
                          0B56  ; error message.
                          0B56  ;
                          0B56  300$:
                 02   8A  0B56          BICB2   #UETUNT$M_TESTABLE,-      ; The MPM is not testable...
              0B A6       0B58                  UETUNT$B_FLAGS(R6)        ; ...(as far as one-shot mode thinks)
      55   01F1 C6   DE  0B5A          MOVAL   MA_Q_MEMNAM(R6),R5        ; Form $FAO parameter
                          0B5F          $FAO_S  CTRSTR = BAD_CEF,-        ; Form an error message
                          0B5F                  OUTLEN = BUFFER_PTR,-
                          0B5F                  OUTBUF = FAO_BUF,-
                          0B5F                  P1     = R5,-
                          0B5F                  P2     = MA_W_UNIT(R6),-
                          0B5F                  P3     = R4,-
                          0B5F                  P4     = MA_L_SPREFSTATE(R6),-
                          0B5F                  P5     = R10
          000C'CF     DF  0B80          PUSHAL  BUFFER_PTR
                 01   DD  0B84          PUSHL   #1
      00741132 8F     DD  0B86          PUSHL   #UETP$_TEXT!STS$K_ERROR
   00000000'GF   03   FB  0B8C          CALLS   #3,G^LIB$SIGNAL
                      05  0B93          RSB
```

```
                                       0B94   INTERPROC_GS:
                                       0B94   ;+
                                       0B94   ;       Start the interprocessor communications part of the shared memory test.
                                       0B94   ;       Mail to each processor which appears in the "available" mask in the
                                       0B94   ;       interprocessor global section, a message which includes the date-time
                                       0B94   ;       quadword at the end of that section's private area in the
                                       0B94   ;       interprocessor global section.  We will expect a reply to our mailbox,
                                       0B94   ;       to be received at AST level.  Set a timer to catch the case where
                                       0B94   ;       another processor doesn't reply.
                                       0B94   ;-
                                       0B94
                                       0B94   ; NOTE WELL!  Within the following loop, registers R6 through R10 are set and
                                       0B94   ; assumed to remain unchanged.
 56   01D8'CF    000001D8'8F    C1     0B94           ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; R6 points to node for current memory
                                       0B9E   10$:
           57    062C C6        D0     0B9E           MOVL    MA_Q_COMGSRAD(R6),R7    ; R7 will point into the interproc GS
                           58   D4     0BA3           CLRL    R8                      ; R8 is STRTPOS for FFS operations
                 59      04     D0     0BA5           MOVL    #PROCESSOR_MAX,R9       ; R9 is SIZE for FFS operations
                                       0BA8   20$:
     5A    24 A7    59   58     EA     0BA8           FFS     R8,R9,GS_L_AVAILABLE(R7),R10 ; Search for a willing processor
                         03     12     0BAE           BNEQ    30$                     ; BR if we found one
                       00E2     31     0BB0           BRW     50$                     ; BR if there are none left
                                       0BB3   30$:
                    01AC C6     90     0BB3           MOVB    MA_B_MYPORT(R6),-                ; Start filling in message...
                    10B0'CF            0BB7                   SYNCH_MBBUF+MB_B_SENDER          ; ...sender's port number...
           10B1'CF       5A     90     0BBA           MOVB    R10,SYNCH_MBBUF+MB_B_RECEIVER    ; ...receiver's port number...
                    01EC C6     90     0BBF           MOVB    MA_W_UNIT(R6),-                  ; ...unit number...
                    10B2'CF            0BC3                   SYNCH_MBBUF+MB_B_UNIT            ; ...of the memory we share...
                         01     B0     0BC6           MOVW    #MSG_HELLO,-                     ; ...reason for the message...
                    10B3'CF            0BC8                   SYNCH_MBBUF+MB_W_WHY
                 5B    06A8 C7  DE     0BCB           MOVAL   GS_K_PVT_AREAS(R7),R11
 5B    5B    5A   00000200 8F   7A     0BD0           EMUL    #COMGS_PVT_AREA,R10,R11,R11      ; ...(point to)...
                    01F8 CB     7D     0BD9           MOVQ    COMGS_PVT_CODE(R11),-            ; ...and the other processor's...
                    10B5'CF            0BDD                   SYNCH_MBBUF+MB_K_MISC            ; ...date/time
                    0258 C6     28     0BE0           MOVC3   MA_Q_COMMBNAM(R6),-              ; Form the logical name...
        10E3'CF     0260 C6            0BE4                   MA_T_COMMBNAM(R6),SYNCH_MBNAM    ; ...of other processor's MB
        10DB'CF     0258 C6     D0     0BEA           MOVL    MA_Q_COMMBNAM(R6),SYNCH_MBPTR    ; Form descriptor to...
        10DF'CF     10E3'CF     DE     0BF1           MOVAL   SYNCH_MBNAM,SYNCH_MBPTR+4        ; ...the logical name
                 50      5A     D0     0BF8           MOVL    R10,R0                           ; Include the port number...
                       0633     30     0BFB           BSBW    HEXUNIT                          ; ...
              FF A3      50     90     0BFE           MOVB    R0,-1(R3)                        ; ...in the name
                                       0C02           $CREMBX_S CHAN = SYNCH_MBCHN,-   ; Establish a link to other proc's MB
                                       0C02                   LOGNAM = SYNCH_MBPTR
     1102'CF     FA0A1F00 8F    D0     0C19           MOVL    #HELLO_TIMEOUT,SYNCH_MBDAY ; $SETIMR parameters - DAYTIM...
           1106'CF       00     D2     0C22           MCOML   #0,SYNCH_MBDAY+4         ; ...
        54    10    10   01     F0     0C27           INSV    #MSG_HELLO,#16,#16,R4    ; ...REQIDT...
        54    08    08   5A     F0     0C2C           INSV    R10,#8,#8,R4             ; ...
           54    01EC C6        90     0C31           MOVB    MA_W_UNIT(R6),R4         ; ...
                                       0C36           $SETIMR_S DAYTIM = SYNCH_MBDAY,-     ; A timer will let us know...
                                       0C36                     ASTADR = MB_TIMEOUT_AST,-  ; ...if the other processor...
                                       0C36                     REQIDT = R4                ; ...doesn't reply to the following
       00 01DC C6        5A     E2     0C49           BBSS    R10,MA_L_HELLO(R6),40$   ; Log that we expect reply to our mail
                                       0C4F   40$:
                                       0C4F           $QIO_S  CHAN    = SYNCH_MBCHN,-    ; Say "Hi!" to the other processor
                                       0C4F                   FUNC    = #IO$_WRITEVBLK!IO$M_NOW,-
                                       0C4F                   IOSB    = SYNCH_MBISB,-
                                       0C4F                   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  INTERPROC_EF:
                                  0CA4  ;+
                                  0CA4  ;       Start the shared common event flag test.  For this test to work, one
                                  0CA4  ;       processor must have control of the test across all processors.  To
                                  0CA4  ;       ensure that, there is a lock in the interprocessor global section which
                                  0CA4  ;       can be owned by only one process.  For each shared memory, try to grab
                                  0CA4  ;       the lock.  If that fails, try to grab the next memory's lock or proceed
                                  0CA4  ;       to the next part of the test.  If we do grab a lock, associate with a
                                  0CA4  ;       common event flag cluster in that memory.  For each processor connected
                                  0CA4  ;       to the memory, choose a flag in that cluster and send the process a
                                  0CA4  ;       mailbox message requesting the processor to set that bit in the
                                  0CA4  ;       cluster.  Allow a reasonable timeout for that to occur.  If the
                                  0CA4  ;       processor responds by setting the bit, pick another processor to clear
                                  0CA4  ;       the bit, again, allowing a reasonable timeout for that action.  Verify
                                  0CA4  ;       that the bits get set and cleared.  Release the lock in the shared
                                  0CA4  ;       memory.
                                  0CA4  ;-
                                  0CA4
                                  0CA4  ; NOTE WELL!  Within the following loop, registers R6 through R10 are set and
                                  0CA4  ; assumed to remain unchanged.
 56   01D8'CF   000001D8'8F   C1  0CA4          ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; R6 points to node for current memory
                                  0CAE  10$:
            57    062C C6     D0  0CAE          MOVL    MA_Q_COMGSRAD(R6),R7    ; R7 will point into the interproc GS
        08 28 A7    00        E6  0CB3          BBSSI   #0,GS_L_CEF_FLAG(R7),20$ ; Try to grab the interlock
              01AC C6         90  0CB8          MOVB    MA_B_MYPORT(R6),-       ; Bit was clear...
                 2B A7            0CBC                  GS_L_CEF_FLAG+3(R7)
                    03        11  0CBE          BRB     30$                     ; ...my port has the interlock
                                  0CC0  20$:
                  0278        31  0CC0          BRW     150$                    ; Bit was set - someone else has it
                                  0CC3  30$:
                                  0CC3          $ASCEFC_S EFN = #COMEF_EFN,-    ; Associate with interprocessor cluster
                                  0CC3                  NAME = MA_Q_COMEFNAM(R6)
                    58        D4  0CD8          CLRL    R8                      ; R8 is STRTPOS for FFS operations
              59    04        D0  0CDA          MOVL    #PROCESSOR_MAX,R9       ; R9 is SIZE for FFS operations
                                  0CDD  40$:
    5A   24 A7   59    58     EA  0CDD          FFS     R8,R9,GS_L_AVAILABLE(R7),R10 ; Search for a willing processor
                    03        12  0CE3          BNEQ    50$                     ; BR if we found one
                  0243        31  0CE5          BRW     140$                    ; BR if there are none left
                                  0CE8  50$:
              01AC C6         90  0CE8          MOVB    MA_B_MYPORT(R6),-               ; Start filling in message...
              10B0'CF             0CEC                  SYNCH_MBBUF+MB_B_SENDER         ; ...sender's port number...
       10B1'CF    5A          90  0CEF          MOVB    R10,SYNCH_MBBUF+MB_B_RECEIVER   ; ...receiver's port number...
              01EC C6         90  0CF4          MOVB    MA_W_UNIT(R6),-                 ; ...unit no. of shared memory...
              10B2'CF             0CF8                  SYNCH_MBBUF+MB_B_UNIT
                    05        B0  0CFB          MOVW    #MSG_DOSET,-                    ; ...reason for the message...
              10B3'CF             0CFD                  SYNCH_MBBUF+MB_W_WHY
    01AA'CF   01AE'CF         C0  0D00          ADDL2   RANDOM2,RANDOM1                 ; ...
    01AA'CF   05    06        EF  0D07          EXTZV   #6,#5,RANDOM1,-                 ; ...and a random bit to set
              10B5'CF             0D0D                  SYNCH_MBBUF+MB_K_MISC
              0258 C6         28  0D10          MOVC3   MA_Q_COMMBNAM(R6),-             ; Form the logical name...
    10E3'CF   0260 C6             0D14                  MA_T_COMMBNAM(R6),SYNCH_MBNAM   ; ...of other processor's MB
    10DB'CF   0258 C6         D0  0D1A          MOVL    MA_Q_COMMBNAM(R6),SYNCH_MBPTR   ; Form descriptor to...
    10DF'CF   10E3'CF         DE  0D21          MOVAL   SYNCH_MBNAM,SYNCH_MBPTR+4       ; ...the logical name
              50    5A        D0  0D28          MOVL    R10,R0                  ; Include the port number...
                  0503        30  0D2B          BSBW    HEXUNIT                 ; ...
         FF A3    50          90  0D2E          MOVB    R0,-1(R3)               ; ...in the name
```

```
                                0D32         $CREMBX_S CHAN = SYNCH_MBCHN,-  ; Establish a link to other proc's MB
                                0D32                   LOGNAM = SYNCH_MBPTR
 1102'CF   FA0A1F00 8F     D0   0D49         MOVL     #DOSET_TIMEOUT,SYNCH_MBDAY ; $SETIMR parameters - DAYTIM...
         1106'CF   00      D2   0D52         MCOML    #0,SYNCH_MBDAY+4         ; ...
      54   10   10   05    F0   0D57         INSV     #MSG_DOSET,#16,#16,R4    ; ...REQIDT...
      54   08   08   5A    F0   0D5C         INSV     R10,#8,#8,R4             ; ...
      54   01EC C6         90   0D61         MOVB     MA_W_UNIT(R6),R4         ; ...
                                0D66         $SETIMR_S DAYTIM = SYNCH_MBDAY,-    ; A timer will let us know...
                                0D66                   ASTADR = MB_TIMEOUT_AST,- ; ...if the other processor...
                                0D66                   REQIDT = R4               ; ...doesn't reply to the following
   00 01E4 C6      5A      E2   0D79         BBSS     R10,MA_L_DOSET(R6),60$   ; Log that we expect reply to our mail
                                0D7F    60$:
                                0D7F         $QIO_S   CHAN   = SYNCH_MBCHN,-   ; Tell the other processor to set a bit
                                0D7F                  FUNC   = #IO$_WRITEVBLK!IO$M_NOW,-
                                0D7F                  IOSB   = SYNCH_MBISB,-
                                0D7F                  P1     = SYNCH_MBBUF,-
                                0D7F                  P2     = #MB_K_END
                   54      DD   0DA2         PUSHL    R4
         1422'CF   01      FB   0DA4         CALLS    #1,QIOMB_SYNCH_CHECK     ; Check the completion status
                                0DA9         $DASSGN_S CHAN = SYNCH_MBCHN      ; We no longer need the MB
                                0DB5    ;
                                0DB5    ; Now we must wait.  The logic of the interprocessor CEF test dictates that we
                                0DB5    ; know for sure that either the other processor successfully set the bit or we
                                0DB5    ; know for sure that it did not.  We will wait for either the receipt of a
                                0DB5    ; mailbox in reply (must happen at AST level) or the expiration of the timer.
                                0DB5    ; Note that on return, we cannot assume the contents of the mailbox received
                                0DB5    ; at AST level because another mailbox may have been received as well, however
                                0DB5    ; the mailbox receipt routine is smart enough to realize this and will move
                                0DB5    ; the contents of the reply we want into the synchronous mailbox buffer.  If
                                0DB5    ; the buffer is unchanged, assume the transfer failed and we timed out.
                                0DB5    ;
                                0DB5         $HIBER_S                          ; Wait for mailbox AST or time out
                   06      B1   0DBC         CMPW     #MSG_ISSET,-             ; Message acknowledged?
             10B3'CF            0DBE                  SYNCH_MBBUF+MB_W_WHY
                   13      13   0DC1         BEQL     70$                      ; BR if it was
         00000060 8F       C1   0DC3         ADDL3    #COMEF_EFN,-             ; Timed out. Event already logged,....
      50   10B5'CF              0DC9                  SYNCH_MBBUF+MB_K_MISC,R0 ; ...
                                0DCD         $SETEF_S EFN = R0                 ; ...so just recover
                                0DD6    70$:
                                0DD6         $READEF_S EFN = #COMEF_EFN,-      ; Check to see...
                                0DD6                   STATE = SYNCH_MBBUF+MB_K_MISC+4
             10B5'CF       9C   0DE7         ROTL     SYNCH_MBBUF+MB_K_MISC,-  ; ...if the bit we wanted...
              5B   01           0DEB                  #1,R11
      5B   10B9'CF         D1   0DED         CMPL     SYNCH_MBBUF+MB_K_MISC+4,R11 ; ...got set
                   19      13   0DF2         BEQL     80$                      ; BR if it did
   00 01C0 C6      5A      E2   0DF4         BBSS     R10,MA_L_DOSETFAIL(R6),75$ ; Bit wasn't set, log the error
                                0DFA    75$:
         00000060 8F       C1   0DFA         ADDL3    #COMEF_EFN,-             ; Event is now logged,...
      50   10B5'CF              0E00                  SYNCH_MBBUF+MB_K_MISC,R0 ; ...
                                0E04         $SETEF_S EFN = R0                 ; ...so just recover
                                0E0D    80$:
```

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

```
                        0ECA      ;
                        0ECA      ; Again, for the same reasons as above, we must wait.  Likewise, the mailbox
                        0ECA      ; AST routine and the mailbox timeout routine will act on our buffer as they
                        0ECA      ; did before.
                        0ECA      ;
                        0ECA              $HIBER_S                              ; Wait for mailbox AST or time out
              08    B1  0ED1              CMPW    #MSG_ISCLEAR,-                ; Message acknowledged?
         10B3'CF        0ED3                      SYNCH_MBBUF+MB_W_WHY
              13    13  0ED6              BEQL    120$                          ; BR if it was
     00000060 8F    C1  0ED8              ADDL3   #COMEF_EFN,-                  ; Timed out. Event already logged,....
     50  10B5'CF        0EDE                      SYNCH_MBBUF+MB_K_MISC,R0      ; ...
                        0EE2              $CLREF_S EFN = R0                     ; ...so just recover
                        0EEB      120$:
                        0EEB              $READEF_S EFN = #COMEF_EFN,-          ; Check to see if the bit we wanted...
                        0EEB                      STATE = SYNCH_MBBUF+MB_K_MISC+4
         10B9'CF    D5  0EFC              TSTL    SYNCH_MBBUF+MB_K_MISC+4       ; ...got cleared
              19    13  0F00              BEQL    130$                          ; BR if it did
 00 01C4 C6   5B    E2  0F02              BBSS    R11,MA_L_DOCLEARFAIL(R6),125$ ; Bit wasn't cleared, log it
                        0F08      125$:
     00000060 8F    C1  0F08              ADDL3   #COMEF_EFN,-                  ; Event is now logged,...
     50  10B5'CF        0F0E                      SYNCH_MBBUF+MB_K_MISC,R0      ; ...
                        0F12              $CLREF_S EFN = R0                     ; ...so just recover
                        0F1B      130$:
     5B   5A  58    C3  0F1B              SUBL3   R8,R10,R11                    ; Get number of bits we searched
              5B    D6  0F1F              INCL    R11                           ; Correct off-by-one (inclusive search)
          59  5B    C2  0F21              SUBL2   R11,R9                        ; Adjust number of bits left to search
     58   5A  01    C1  0F24              ADDL3   #1,R10,R8                     ; Point to new first bit to search
            FDB2    31  0F28              BRW     40$                           ; Loop thru all procs on this memory
                        0F2B      140$:
                        0F2B              $DACEFC_S EFN = #COMEF_EFN            ; Disassociate from shared CEF cluster
           28 A7    D4  0F38              CLRL    GS_L_CEF_FLAG(R7)             ; Release the interlock
                        0F3B      150$:
          56  66    C0  0F3B              ADDL2   (R6),R6                       ; Point to node for next memory
 56  000001D8'8F    D1  0F3E              CMPL    #UNIT_LIST,R6                 ; Back at the start of the queue?
              03    13  0F45              BEQL    160$                          ; BR if we are - we're finished
            FD64    31  0F47              BRW     10$                           ; More to go so loop
                        0F4A      160$:
```

```
                                   0F4A  MEM_INTERLOCK:
                                   0F4A  ;+
                                   0F4A  ;       Start the part of the test which tests use of interlocking instructions
                                   0F4A  ;       in the shared memory.  There is an interlock (please forgive the
                                   0F4A  ;       overloading of the word ''interlock'') longword in the interprocessor
                                   0F4A  ;       global section for this part of the test.  If we can't get that lock
                                   0F4A  ;       for a given memory, try to get it for the next memory or proceed to the
                                   0F4A  ;       next part of the test.  If we do get it, send mail to each processor
                                   0F4A  ;       connected to the memory instructing the processor to drop everything
                                   0F4A  ;       else that it is doing and begin the interlocking test.
                                   0F4A  ;-
                                   0F4A
                                   0F4A  ; NOTE WELL!  Within the following loop, registers R6 through R10 are set and
                                   0F4A  ; assumed to remain unchanged.
  56  01D8'CF   000001D8'8F   C1   0F4A          ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; R6 points to node for current memory
                                   0F54  10$:
              57   062C C6    D0   0F54          MOVL    MA_Q_COMGSRAD(R6),R7     ; Point to the start of interproc GS
           08 34 A7     00    E6   0F59          BBSSI   #0,GS_L_INTLK_FLAG(R7),20$ ; Try to grab interlock
                  01AC C6     90   0F5E          MOVB    MA_B_MYPORT(R6),-        ; Bit was clear...
                     37 A7         0F62                  GS_L_INTLK_FLAG+3(R7)
                        03    11   0F64          BRB     30$                      ; ...my port has the interlock
                                   0F66  20$:
                      017C    31   0F66          BRW     210$                     ; Bit was set, someone else has it
                                   0F69  30$:
                     2C A7    7C   0F69          CLRQ    GS_L_PARTICIPATING(R7)   ; Clear participating and finished masks
             00    67   00    2C   0F6C          MOVC5   #0,(R7),#0,-             ; Clear the common ADAWI words...
                        14         0F70                  #<PROCESSOR_MAX+1>*4,-   ; ...and the private ADAWI words
                     38 A7         0F71                  GS_L_ADAWI_COM(R7)
                     60 A7    7C   0F73          CLRQ    GS_Q_BUSY(R7)            ; Set up the busy queue...
                     58 A7    7C   0F76          CLRQ    GS_Q_FREE(R7)            ; ...clear out the free queue...
         5B   00000064 8F     D0   0F79          MOVL    #COMGS_QUEUE_MAX,R11     ; ...and then set it up...
              55     68 A7    DE   0F80          MOVAL   GS_K_Q_ENTRIES(R7),R5    ; ...
                                   0F84  40$:
              58 A7     65    5C   0F84          INSQHI  (R5),GS_Q_FREE(R7)       ; Put an entry on the free queue
                     55 10    C0   0F88          ADDL2   #16,R5                   ; Skip flink, blink, port, filler
                     F6 5B    F5   0F8B          SOBGTR  R11,40$
     08     00    67    00    2C   0F8E          MOVC5   #0,(R7),#0,#PROCESSOR_MAX*2,- ; Clear out the counters...
                     4C A7         0F93                  GS_K_Q_COUNT(R7)              ; ...for each processor's entries
                                   0F95
                        58    D4   0F95          CLRL    R8                       ; R8 is STRTPOS for FFS operations
                     59 04    D0   0F97          MOVL    #PROCESSOR_MAX,R9        ; R9 is SIZE for FFS operations
                                   0F9A  50$:
   5A     24 A7     59  58    EA   0F9A          FFS     R8,R9,GS_L_AVAILABLE(R7),R10 ; Search for a willing processor
                        03    12   0FA0          BNEQ    60$                      ; BR if we found one
                      00AA    31   0FA2          BRW     100$                     ; BR if there are none left
                                   0FA5  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)
         59   5B        C2   1045  2193            SUBL2   R11,R9                       ; Adjust number of bits left to search
    58   5A   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  ;
                     105D  ; We've finished our portion of the test; it's a fair guess to say that other
                     105D  ; processors are probably close to finishing also.  However, to be on the safe
                     105D  ; side, allow some extra time for them to finish.  We'll know they're finished
                     105D  ; because the mask of processors which have finished will be the same as the
                     105D  ; mask as those which started.
                     105D  ;
            5B   D4  105D          CLRL    R11                         ; Clear out our counter
                     105F  120$:
         2C A7   CD  105F          XORL3   GS_L_PARTICIPATING(R7),-    ; Has everyone finished yet?
     52  30 A7       1062                  GS_L_FINISHED(R7),R2
            24   13  1065          BEQL    130$                        ; BR if they have
                     1067          $SCHDWK_S DAYTIM = ONE_SECOND       ; Wait a second...
                     1078          $HIBER_S                            ; ...if they haven't
      DC 5B      0A  F2  107F      AOBLSS  #INTERLOCK_LIMIT,R11,120$   ; Loop a reasonable number of times
   01D0 C6       52  C8  1083      BISL2   R2,MA_L_INTLKTMO(R6)        ; Timed out, remember who is left
          0057   31  1088          BRW     200$                        ; Consistency checking is useless
                     108B  130$:
                     108B
                     108B  ;
                     108B  ; To see if the memory interlocking worked correctly, we check the results of
                     108B  ; the two tests performed.  For the ADAWI test, the common longword should
                     108B  ; equal the sum of the private longwords.  For the queue test, the busy queue
                     108B  ; should be empty, all entries should be on the free queue and no processor
                     108B  ; should have any queue entries outstanding.
                     108B  ;
            5B   D4  108B          CLRL    R11                         ; Initialize sum
            55   D4  108D          CLRL    R5                          ; Initialize counter and pointer
                     108F  140$:
   5B   3C A745  C0  108F          ADDL2   GS_K_ADAWI_PVT(R7)[R5],R11  ; Add in this processor's count
      F7 55   04 F2  1094          AOBLSS  #PROCESSOR_MAX,R5,140$      ; Loop through all count longwords
     5B  38 A7   D1  1098          CMPL    GS_L_ADAWI_COM(R7),R11      ; Expected sum equal actual sum?
            06   13  109C          BEQL    150$                        ; BR if it is
         30 A7   C8  109E          BISL2   GS_L_FINISHED(R7),-         ; Otherwise, remember the group...
       01D4 C6       10A1                  MA_L_IADAWI(R6)             ; ...of CPUs who didn't add up
                     10A4  150$:
                     10A4
     50  60 A7   7D  10A4          MOVQ    GS_Q_BUSY(R7),R0            ; Is busy queue empty?  (R0 is trash)
            32   12  10A8          BNEQ    190$                        ; BR if it isn't
            55   D4  10AA          CLRL    R5                          ; Initialize counter and pointer
                     10AC  160$:
       4C A745   B5  10AC          TSTW    GS_K_Q_COUNT(R7)[R5]        ; Any entries for this processor?
            2A   12  10B0          BNEQ    190$                        ; BR if there are - we're screwed up
      F6 55   04 F2  10B2          AOBLSS  #PROCESSOR_MAX,R5,160$      ; Loop to check all processors
                     10B6
         58 A7   D5  10B6          TSTL    GS_Q_FREE(R7)               ; Anything on free queue?
            21   13  10B9          BEQL    190$                        ; BR if not - all screwed up
     5B  58 A7   DE  10BB          MOVAL   GS_Q_FREE(R7),R11           ; Point to header of free queue
         54  5B  D0  10BF          MOVL    R11,R4                      ; Set up a constant to compare with
 55  00000064 8F D0  10C2          MOVL    #COMGS_QUEUE_MAX,R5         ; Initialize counter
                     10C9  170$:
         5B  6B  C0  10C9          ADDL2   (R11),R11                   ; Point to the next queue entry
             55  D7  10CC          DECL    R5                          ; Count down entries in queue
             07  19  10CE          BLSS    180$                        ; BR if we've examined all entries
         5B  54  D1  10D0          CMPL    R4,R11                      ; Pointing back at the header now?
             07  13  10D3          BEQL    190$                        ; Shouldn't be!  BR if we are
             F2  11  10D5          BRB     170$                        ; See where the next entry points us
```

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

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

```
                             112F   ;
                             112F   ; For the second test of memory interlocking, processors will take an entry
                             112F   ; from a "free" queue and add it to a "busy" queue.  They will keep count in
                             112F   ; a per processor area of the number of entries they have in the busy queue.
                             112F   ; When the free queue is exhausted, processors will start to remove queue
                             112F   ; entries (anyone's) from the busy queue and replace them in the free queue,
                             112F   ; decrementing the appropriate count.  The sequence repeats a specified number
                             112F   ; of times.  When the dust dies down, each processor's count should be zero,
                             112F   ; the busy queue empty and the free queue full (the same as our initial state).
                             112F   ;
       58    01AC C6    9A   112F          MOVZBL  MA_B_MYPORT(R6),R8          ; Set up a pointer to...
       58    4C A748    3E   1134          MOVAW   GS_K_Q_COUNT(R7)[R8],R8     ; ...my counter
                  59    D4   1139          CLRL    R9                          ; Clear loop counter
                             113B   110$:
        5A    58 A7     5E   113B          REMQHI  GS_Q_FREE(R7),R10           ; Pick off a "free" queue entry
                  04    1C   113F          BVC     120$                        ; BR if we got one from the queue
                  F8    1F   1141          BCS     110$                        ; BR if secondary interlock failed
                  11    11   1143          BRB     140$                        ; Queue is empty, start refilling it
                             1145   120$:
   08 AA     01AC C6    9A   1145          MOVZBL  MA_B_MYPORT(R6),8(R10)      ; Mark the entry as mine
                             114B   130$:
       60 A7      6A    5C   114B          INSQHI  (R10),GS_Q_BUSY(R7)         ; Put the entry in the "busy" queue
                  FA    1F   114F          BCS     130$                        ; BR if secondary interlock failed
          68      01    58   1151          ADAWI   #1,(R8)                     ; Count my entries in the busy queue
                  E5    11   1154          BRB     110$                        ; Loop until free queue is exhausted
                             1156   140$:
        5A    60 A7     5F   1156          REMQTI  GS_Q_BUSY(R7),R10           ; Pick off a "busy" queue entry
                  04    1C   115A          BVC     150$                        ; BR if we got one from the queue
                  F8    1F   115C          BCS     140$                        ; BR if secondary interlock failed
                  13    11   115E          BRB     170$                        ; Queue is empty, count an iteration
                             1160   150$:
        5B    08 AA     D0   1160          MOVL    8(R10),R11                  ; Who owned this entry?
  4C A74B    FFFF 8F    58   1164          ADAWI   #-1,GS_K_Q_COUNT(R7)[R11] ; Decrement the owner's count
                             116B   160$:
       58 A7      6A    5D   116B          INSQTI  (R10),GS_Q_FREE(R7)         ; Replace the entry in the free queue
                  FA    1F   116F          BCS     160$                        ; BR if secondary interlock failed
                  E3    11   1171          BRB     140$                        ; Loop until busy queue is empty
                             1173   170$:
  C0 59   000000C8 8F   F2   1173          AOBLSS  #QUEUE_LIMIT,R9,110$        ; Loop until counter is exhausted
                             117B
                             117B   ;
                             117B   ; This processor has finished its share of the memory interlock test.  However
                             117B   ; the success or failure of the test can be checked only by the initiating
                             117B   ; processor and then only when all processors which indicated that they were
                             117B   ; participating have indicated that they finished.  So for now, just indicate
                             117B   ; that we're done.
                             117B   ;
             01AC C6    E6   117B          BBSSI   MA_B_MYPORT(R6),-           ; Say that we've finished
             00 30 A7        117F                  GS_L_FINISHED(R7),200$
                             1182   200$:
                        04   1182          RET
```

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

```
                         1231          .SBTTL  Quickie Subroutines
                         1231  ;++
                         1231  ; FUNCTIONAL DESCRIPTION:
                         1231  ;       ...being a set of short subroutines not worthy of much comment.
                         1231  ;
                         1231  ; TYPICAL CALLING SEQUENCE:
                         1231  ;       MOVx    arg,R0
                         1231  ;       BSBW    subroutine
                         1231  ;       result returned in R0
                         1231  ;
                         1231  ; INPUT PARAMETERS:
                         1231  ;       defined for each routine
                         1231  ;
                         1231  ; OUTPUT PARAMETERS:
                         1231  ;       defined for each routine
                         1231  ;
                         1231  ;--
                         1231
                         1231  ;
                         1231  ; Given the unit number of a multiport memory (one byte), return its
                         1231  ; ASCII equivalent.  Routine is very trusting.
                         1231  ;
                         1231  HEXUNIT:
          50    09    91 1231          CMPB    #9,R0                   ; Is unit within decimal digits?
                03    18 1234          BGEQ    10$                     ; BR if it is
          50    37    80 1236          ADDB2   #<^A/A/-10>,R0          ; Add a kludge factor if it isn't
                         1239  10$:
          50    30    80 1239          ADDB2   #^A/0/,R0               ; Convert hex to ASCII
                      05 123C          RSB
                         123D
                         123D
                         123D  ;
                         123D  ; Given the address of a buffer of longwords in R1 and the buffer size in
                         123D  ; longwords in R0, fill the buffer with pseudo-random data.
                         123D  ;
                         123D  RANDOM_FILL:
 01AA'CF   01AE'CF    C0 123D          ADDL2   RANDOM2,RANDOM1         ; Create a "random" longword
      81   01AA'CF    D0 1244          MOVL    RANDOM1,(R1)+           ; Stuff it in our buffer...
             F1 50    F5 1249          SOBGTR  R0,RANDOM_FILL          ; ...until the buffer is full
                      05 124C          RSB
```

```
                               124D  ;
                               124D  ; Take a bit mask in R0 and from it produce a list on the stack of those bit
                               124D  ; positions in R0 which have a 1 in them.  The list is suitable for use with
                               124D  ; the !#() $FAO directive.  R0 returns the number of items in the list and R1
                               124D  ; is trashed.
                               124D  ;
                               124D  MASK_2_ARGLST:
                      02   BA  124D          POPR    #^M<R1>                 ; Save our return address
                      00   DD  124F          PUSHL   #0                      ; Set up counter on stack
                               1251  10$:
                      50   D5  1251          TSTL    R0                      ; Will FFS find any bits?
                      0F   13  1253          BEQL    20$                     ; BR if it won't
            7E    6E  01   C1  1255          ADDL3   #1,(SP),-(SP)           ; Count one it will find into new counter
    04 AE   50    20  00   EA  1259          FFS     #0,#32,R0,4(SP)         ; Find a lit bit, save its position
         ED 50     04 AE   E4  125F          BBSC    4(SP),R0,10$            ; Ensure we find it only once. Look for next
                               1264  20$:
                50    6E   D0  1264          MOVL    (SP),R0                 ; Return subroutine value
                      61   17  1267          JMP     (R1)                    ; RSB, a la kludge
                               1269
                               1269
                               1269  ;
                               1269  ; Given the address of an $FAO control string in R0, do a "typical" call to the
                               1269  ; $FAO system service - a call which includes only the shared memory name and
                               1269  ; its unit number.  R1 is trashed.  R0 has result of $FAO.  Assumes that we are
                               1269  ; "typical" in that R6 contains the address of the unit block of the current
                               1269  ; memory.
                               1269  ;
                               1269  DO_FAO:
        51    01F1 C6      DE  1269          MOVAL   MA_Q_MEMNAM(R6),R1      ; Set up $FAO parameter
                               126E          $FAO_S  CTRSTR = (R0),-
                               126E                  OUTLEN = BUFFER_PTR,-
                               126E                  OUTBUF = FAO_BUF,-
                               126E                  P1     = R1,-
                               126E                  P2     = MA_W_UNIT(R6)
                      05       1285          RSB
```

```
                                    1286  ;
                                    1286  ; With the same inputs as the DO_FAO subroutine, do the $FAO and follow it
                                    1286  ; by a call to LIB$SIGNAL which treats the result of the $FAO as a warning
                                    1286  ; or info message.  R0 and R1 are trashed.
                                    1286  ;
                                    1286  DO_FAO_SIGNAL:
                         00    DD   1286          PUSHL   #STS$K_WARNING
                                    1288          .ENABLE LSB
                         02    11   1288          BRB     10$
                                    128A  DO_FAO_INFO:
                         03    DD   128A          PUSHL   #STS$K_INFO
                                    128C  10$:
                                    128C          .DISABLE LSB
                         DB    10   128C          BSBB    DO_FAO                        ; Get the $FAO over with
 51    8E    00741130 8F   C9   128E          BISL3   #UETP$_TEXT,(SP)+,R1          ; Form the correct message code
                  000C'CF   DF   1296          PUSHAL  BUFFER_PTR
                         01    DD   129A          PUSHL   #1
                         51    DD   129C          PUSHL   R1
         00000000'GF     03    FB   129E          CALLS   #3,G^LIB$SIGNAL
                         05         12A5          RSB
                                    12A6
                                    12A6
                                    12A6  ;
                                    12A6  ; Similar to DO_FAO_SIGNAL, but instead of doing the call to LIB$SIGNAL, die
                                    12A6  ; by way of ERROR_EXIT, which does its own call to LIB$SIGNAL.  No return.
                                    12A6  ;
                                    12A6  DO_FAO_EXIT:
                         C1    10   12A6          BSBB    DO_FAO                        ; Get the $FAO over with
                  000C'CF   DF   12A8          PUSHAL  BUFFER_PTR
                         01    DD   12AC          PUSHL   #1
             00741132 8F   DD   12AE          PUSHL   #UETP$_TEXT!STS$K_ERROR
                         03    DD   12B4          PUSHL   #3
                       0DE7    31   12B6          BRW     ERROR_EXIT
```

```
                          12B9         .SBTTL  Shared Memory Name Routine
                          12B9   ;++
                          12B9   ; FUNCTIONAL DESCRIPTION:
                          12B9   ;       Given the unit number of a shared memory, extract from the SHD and ADP
                          12B9   ;       the corresponding memory name and name length and the port to which our
                          12B9   ;       CPU is connected.
                          12B9   ;
                          12B9   ; CALLING SEQUENCE:
                          12B9   ;       $CMKRNL_S ROUTIN = SHM_INFO,-
                          12B9   ;                 ARGLST = argument-list-as-described-below
                          12B9   ;
                          12B9   ; INPUT PARAMETERS:
                          12B9   ;       @04(AP) - Unit number of multiport memory
                          12B9   ;
                          12B9   ; IMPLICIT INPUTS:
                          12B9   ;       NONE
                          12B9   ;
                          12B9   ; OUTPUT PARAMETERS:
                          12B9   ;       @08(AP) - Address of byte for shared memory port
                          12B9   ;       @12(AP) - Address for .ASCII string of shared memory name
                          12B9   ;       @16(AP) - Address of word for length of shared memory name
                          12B9   ;
                          12B9   ; IMPLICIT OUTPUTS:
                          12B9   ;       NONE
                          12B9   ;
                          12B9   ; COMPLETION CODES:
                          12B9   ;       SS$_NORMAL
                          12B9   ;       UETP$_DENOSU if no arglist supplied or can't find specified unit
                          12B9   ;
                          12B9   ; SIDE EFFECTS:
                          12B9   ;       NONE
                          12B9   ;
                          12B9   ;--
                          12B9
                          12B9   SHM_INFO:
                    0FFC  12B9         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                          12BB
              5C     D5   12BB         TSTL    AP                      ; Were we supplied an argument list?
              48     13   12BD         BEQL    30$                     ; BR if not
 56    00000000'GF   D0   12BF         MOVL    G^EXE$GL_SHBLIST,R6     ; Point to SHB list
              3F     13   12C6         BEQL    30$                     ; BR if none - can't find ours
                          12C8   10$:
      57    1C A6    D0   12C8         MOVL    SHB$L_ADP(R6),R7        ; Get ADP pointer
              39     13   12CC         BEQL    30$                     ; BR if there is none - we've goofed
            14 A6    91   12CE         CMPB    SHB$B_NEXUS(R6),-       ; Does ADP match our SHB?
            0C A7         12D1                 ADP$W_TR(R7)
              32     12   12D3         BNEQ    30$                     ; BR if not - someone goofed
         57   67     D0   12D5         MOVL    ADP$L_CSR(R7),R7        ; Get CSR of adapter
              2D     13   12D8         BEQL    30$                     ; BR if something is screwey
      57    1C A7    D0   12DA         MOVL    MPM$L_MR(R7),R7         ; (I/O space reference restriction)
              0E     EF   12DE         EXTZV   #MPM$V_MR_UNIT,-        ; Get the unit number of our memory
  57     57   02          12E0                 #MPM$S_MR_UNIT,R7,R7
      57    04 BC    B1   12E3         CMPW    @04(AP),R7              ; Got the right memory unit?
              19     12   12E7         BNEQ    20$                     ; BR if not
      57    04 A6    D0   12E9         MOVL    SHB$L_DATAPAGE(R6),R7   ; Got correct memory. Get its SHD
 10 BC      20 A7    9B   12ED         MOVZBW  SHD$T_NAME(R7),@16(AP)  ; Get the length of the memory name...
            10 BC    28   12F2         MOVC3   @16(AP),-               ; ...and the name itself
```

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

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

```
                          1422          .SBTTL  Mailbox $QIO Completion Status Routines
                          1422  ;++
                          1422  ; FUNCTIONAL DESCRIPTION:
                          1422  ;       Check completion status when we send an interprocessor mailbox.  Print
                          1422  ;       a message if there was an error.  Note that we have two entry points -
                          1422  ;       one for synchronous sends and one for AST sends.
                          1422  ;
                          1422  ; CALLING SEQUENCE:
                          1422  ;       CALLS   #1,QIOMB_%%%%%_CHECK
                          1422  ;
                          1422  ; INPUT PARAMETERS:
                          1422  ;       06(AP) - Message type (word).
                          1422  ;       05(AP) - Receiver's port number on shared memory (byte).
                          1422  ;       04(AP) - Shared memory unit number (byte).
                          1422  ;
                          1422  ; IMPLICIT INPUTS:
                          1422  ;       The IOSB from either synchronous or AST sending.
                          1422  ;
                          1422  ; OUTPUT PARAMETERS:
                          1422  ;       NONE
                          1422  ;
                          1422  ; IMPLICIT OUTPUTS:
                          1422  ;       NONE
                          1422  ;
                          1422  ; COMPLETION CODES:
                          1422  ;       NONE
                          1422  ;
                          1422  ; SIDE EFFECTS:
                          1422  ;       Message if there was a problem sending.
                          1422  ;
                          1422  ;--
                          1422
                          1422  QIOMB_SYNCH_CHECK:
                    0FFC  1422          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
   52    10D3'CF     3C   1424          MOVZWL  SYNCH_MBISB,R2          ; Save synchronous unique data
             07      11   1429          BRB     QIOMB_CHECK
                          142B  QIOMB_AST_CHECK:
                    0FFC  142B          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
   52    112D'CF     3C   142D          MOVZWL  AST_MBISB,R2            ; Save AST unique data
                          1432
                          1432  QIOMB_CHECK:
          4D 52      E8   1432          BLBS    R2,10$                  ; Exit quickly if there was no error
             52      DD   1435          PUSHL   R2                      ; Form a message...
      7E  06 AC      3C   1437          MOVZWL  06(AP),-(SP)            ; ...with the error code...
      7E  05 AC      9A   143B          MOVZBL  05(AP),-(SP)            ; ...and the info from ASTPRM...
      7E  04 AC      9A   143F          MOVZBL  04(AP),-(SP)            ; ...on the stack...
          55 5E      D0   1443          MOVL    SP,R5                   ; ...so that we don't interfere...
        FF30 CE      DF   1446          PUSHAL  -TEXT_BUFFER-8(SP)      ; ...with any other messages...
    000000C8 8F      DD   144A          PUSHL   #TEXT_BUFFER            ; ...that may be being formed...
          54 5E      D0   1450          MOVL    SP,R4                   ; ...
5E  000000C8 8F      C2   1453          SUBL2   #TEXT_BUFFER,SP         ; ...
                          145A          $FAOL_S CTRSTR = BAD_MB_QIO,-   ; ...
                          145A                  OUTLEN = (R4),-
                          145A                  OUTBUF = (R4),-
                          145A                  PRMLST = (R5)
             54      DD   146B          PUSHL   R4
             01      DD   146D          PUSHL   #1
```

```
                 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            .SBTTL  Mailbox Received AST Routine
                        1483    ;++
                        1483    ; FUNCTIONAL DESCRIPTION:
                        1483    ;       Interprocessor mailbox messages are received here.  Based on the
                        1483    ;       MB_W_WHY field, dispatch to a section which will process the message.
                        1483    ;
                        1483    ; CALLING SEQUENCE:
                        1483    ;       Called via AST upon receipt of a message to this processor's mailbox.
                        1483    ;
                        1483    ; INPUT PARAMETERS:
                        1483    ;       04(AP) = address of the node for the shared memory through which the
                        1483    ;                message was sent
                        1483    ;
                        1483    ; IMPLICIT INPUTS:
                        1483    ;       AST_MBISB - IOSB when mail is received
                        1483    ;       AST_MBBUF - Buffer into which mail is written
                        1483    ;
                        1483    ; OUTPUT PARAMETERS:
                        1483    ;       NONE
                        1483    ;
                        1483    ; IMPLICIT OUTPUTS:
                        1483    ;       NONE
                        1483    ;
                        1483    ; COMPLETION CODES:
                        1483    ;       NONE
                        1483    ;
                        1483    ; SIDE EFFECTS:
                        1483    ;       NONE
                        1483    ;
                        1483    ;--
                        1483
                        1483    RCVMB_AST:
                  0FFC  1483            .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        1485
   6D  1F17'CF    DE    1485            MOVAL   SSERROR,(FP)               ; Set up AST level exception handler
   56    04 AC    D0    148A            MOVL    04(AP),R6                  ; Get the pointer to node for this MB
   7E  110D'CF    B0    148E            MOVW    AST_MBBUF+MB_W_WHY,-(SP)   ; Set up QIOMB_AST_CHECK parameter...
   7E  110B'CF    90    1493            MOVB    AST_MBBUF+MB_B_RECEIVER,-(SP) ; ...
   7E    01EC C6  90    1498            MOVB    MA_B_UNIT(R6),-(SP)        ; ...
   8A AF     01   FB    149D            CALLS   #1,QIOMB_AST_CHECK         ; Check $QIO completion status
       110B'CF    91    14A1            CMPB    AST_MBBUF+MB_B_RECEIVER,-  ; Do a minimal consistency check...
         01AC C6        14A5                    MA_B_MYPORT(R6)            ; ...to see if message is for me
             43   12    14A8            BNEQ    40$                        ; BR if it wasn't
       110C'CF    91    14AA            CMPB    AST_MBBUF+MB_B_UNIT,-      ; More consistency checking
         01EC C6        14AE                    MA_B_UNIT(R6)
             3A   12    14B1            BNEQ    40$                        ; BR if wrong unit
             0A   B1    14B3            CMPW    #MSG_ME2ME,-               ; I can legitimately send myself...
       110D'CF          14B5                    AST_MBBUF+MB_W_WHY         ; ...mail anytime...
             13   13    14B8            BEQL    20$                        ; ...so BR if I, in fact, did...
         062C C6  D5    14BA            TSTL    MA_Q_COMGSRAD(R6)          ; ...but other mail needs interproc GS
             0D   12    14BE            BNEQ    20$                        ; BR if interproc GS was set up
       110A'CF    9C    14C0            ROTL    AST_MBBUF+MB_B_SENDER,-    ; It wasn't. Log that I...
         50  01         14C4                    #1,R0                      ; ...probably got mail...
 01CC C6     50   C8    14C6            BISL2   R0,MA_L_OLDSTUFF(R6)       ; ...which was intended for...
                        14CB                                               ; ...an image which died suddenly
             4B   11    14CB            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  RCV_HELLO:
                              1540  ;
                              1540  ; Some processor is (re-)starting the test and wants to say hi.  It's sent
                              1540  ; the date-time quadword that appears at the end of my private area in the
                              1540  ; interprocessor global section.  If that turns out to be incorrect, let the
                              1540  ; processor know by way of a return mailbox.  If the date-time is correct,
                              1540  ; reply with a different message so saying and expect the processor to
                              1540  ; acknowledge my message.  There is some code which will be common to either
                              1540  ; path; get that out of the way first.  The code paths rejoin later.
                              1540  ;
              110A'CF    90   1540          MOVB    AST_MBBUF+MB_B_SENDER,- ; The proc which sent now will receive
              110B'CF         1544                  AST_MBBUF+MB_B_RECEIVER
              01AC C6    90   1547          MOVB    MA_B_MYPORT(R6),-       ; I am the sender
              110A'CF         154B                  AST_MBBUF+MB_B_SENDER
              0258 C6    28   154E          MOVC3   MA_Q_COMMBNAM(R6),-     ; Form the logical name...
     113D'CF  0260 C6         1552                  MA_T_COMMBNAM(R6),AST_MBNAM ; ...of the other processor's MB
     1135'CF  0258 C6    D0   1558          MOVL    MA_Q_COMMBNAM(R6),AST_MBPTR ; Form descriptor of...
     1139'CF  113D'CF    DE   155F          MOVAL   AST_MBNAM,AST_MBPTR+4   ; ...the logical name
          50  110B'CF    D0   1566          MOVL    AST_MBBUF+MB_B_RECEIVER,R0 ; Include the port number...
                 FCC3    30   156B          BSBW    HEXUNIT                 ; ...
             FF A3   50  90   156E          MOVB    R0,-1(R3)               ; ...in the mailbox name
                              1572          $CREMBX_S CHAN = AST_MBCHN,-    ; Establish a link to other proc's MB
                              1572                  LOGNAM = AST_MBPTR
              110B'CF    F0   1589          INSV    AST_MBBUF+MB_B_RECEIVER,- ; Set up common parts of REQIDT...
          57  08     08       158D                  #8,#8,R7
          57  01EC C6    90   1590          MOVB    MA_W_UNIT(R6),R7        ; ...and/or status check, below
                              1595  ; End of common code
                              1595
        01A4 C6     08   29   1595          CMPC3   #8,MA_Q_DAYTIME(R6),-   ; Did I get passed my date-time?
              110F'CF         159A                  AST_MBBUF+MB_K_MISC
                   17    13   159D          BEQL    10$                     ; BR if I did
              110B'CF    9C   159F          ROTL    AST_MBBUF+MB_B_RECEIVER,- ; I got passed trash, so...
          50  01              15A3                  #1,R0
        01BC C6     50   C8   15A5          BISL2   R0,MA_L_SBADCODE(R6)    ; ...log the error
                   04    B0   15AA          MOVW    #MSG_BADCODE,-          ; Set up the reason for the mailbox
              110D'CF         15AC                  AST_MBBUF+MB_W_WHY
  57  10    10     04    F0   15AF          INSV    #MSG_BADCODE,#16,#16,R7 ; Finish info for status check, below
                   36    11   15B4          BRB     20$                     ; Rejoin common code
                              15B6  10$:
                   02    B0   15B6          MOVW    #MSG_ICU_UCME,-         ; The reason for the mailbox
              110D'CF         15B8                  AST_MBBUF+MB_W_WHY
 115C'CF  FA0A1F00 8F    D0   15BB          MOVL    #ICU_UCME_TIMEOUT,AST_MBDAY ; $SETIMR parameters - DAYTIM...
              1160'CF    00 D2 15C4         MCOML   #0,AST_MBDAY+4          ; ...
  57  10    10     02    F0   15C9          INSV    #MSG_ICU_UCME,#16,#16,R7 ; ...and REQIDT (also ststus check)
                              15CE          $SETIMR_S DAYTIM = AST_MBDAY,-  ; We'll need a reply for this one
                              15CE                  ASTADR = MB_TIMEOUT_AST,-
                              15CE                  REQIDT = R7
              110B'CF    9C   15E1          ROTL    AST_MBBUF+MB_B_RECEIVER,- ; We're going to reply...
          50  01              15E5                  #1,R0
        01E0 C6     50   C8   15E7          BISL2   R0,MA_L_ICU_UCME(R6)    ; ...so log that we expect a return
                              15EC          ;BRB    20$                     ; Rejoin common code
                              15EC  20$:
                              15EC          $QIO_S  CHAN    = AST_MBCHN,-   ; Reply to the hello
                              15EC                  FUNC    = #IO$_WRITEVBLK!IO$M_NOW,-
                              15EC                  IOSB    = AST_MBISB,-
                              15EC                  P1      = AST_MBBUF,-
                              15EC                  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  RCV_ICU_UCME:
                              1625  ;
                              1625  ; A processor to which we've sent a hello message liked what we sent and is
                              1625  ; trying to tell us so.  If we did indeed send the message, let the other
                              1625  ; processor know that we received its mail.
                              1625  ;
          50  110A'CF    9A   1625          MOVZBL  AST_MBBUF+MB_B_SENDER,R0 ; Find out who sent us a message
     09 01DC C6     50   E4   162A          BBSC    R0,MA_L_HELLO(R6),10$    ; BR if we expected it...
          50   01   50   9C   1630          ROTL    R0,#1,R0                 ; ...else...
        01C8 C6     50   C8   1634          BISL2   R0,MA_L_NOCAUSE(R6)      ; ...log the error
                              1639  10$:
      57   10   10   01   F0  1639          INSV    #MSG_HELLO,#16,#16,R7    ; We've a reply to our hello...
  57   08   08  110A'CF   F0  163E          INSV    AST_MBBUF+MB_B_SENDER,#8,#8,R7 ; ...
           57  01EC C6   90   1645          MOVB    MA_Q_UNIT(R6),R7         ; ...
                              164A          $CANTIM_S REQIDT = R7            ; ...so we need not worry time out
                              1655
              110A'CF    90   1655          MOVB    AST_MBBUF+MB_B_SENDER,-  ; The proc which sent now will receive
              110B'CF         1659                  AST_MBBUF+MB_B_RECEIVER
              01AC C6    90   165C          MOVB    MA_B_MYPORT(R6),-        ; I am the sender
              110A'CF         1660                  AST_MBBUF+MB_B_SENDER
              0258 C6    28   1663          MOVC3   MA_Q_COMMBNAM(R6),-      ; Form the logical name...
     113D'CF  0260 C6         1667                  MA_T_COMMBNAM(R6),AST_MBNAM ; ...of the other processor's MB
     1135'CF  0258 C6    D0   166D          MOVL    MA_Q_COMMBNAM(R6),AST_MBPTR ; Form descriptor of...
     1139'CF  113D'CF    DE   1674          MOVAL   AST_MBNAM,AST_MBPTR+4    ; ...the logical name
          50  110B'CF    D0   167B          MOVL    AST_MBBUF+MB_B_RECEIVER,R0 ; Include the port number...
                 FBAE    30   1680          BSBW    HEXUNIT                  ; ...
           FF A3    50   90   1683          MOVB    R0,-1(R3)                ; ...in the mailbox name
                              1687          $CREMBX_S CHAN = AST_MBCHN,-     ; Establish a link to other proc's MB
                              1687                  LOGNAM = AST_MBPTR
                   03    B0   169E          MOVW    #MSG_BYEFORNOW,-         ; The reason for the mailbox
              110D'CF         16A0                  AST_MBBUF+MB_W_WHY
                              16A3          $QIO_S  CHAN = AST_MBCHN,-       ; Reply to the hello
                              16A3                  FUNC = #IO$_WRITEVBLK!IO$M_NOW,-
                              16A3                  IOSB = AST_MBISB,-
                              16A3                  P1   = AST_MBBUF,-
                              16A3                  P2   = #MB_K_END
              7E   03    B0   16C6          MOVW    #MSG_BYEFORNOW,-(SP)     ; Set up QIOMB_AST_CHECK parameter...
          7E  110B'CF    90   16C9          MOVB    AST_MBBUF+MB_B_RECEIVER,-(SP) ; ...
          7E  01EC C6    90   16CE          MOVB    MA_Q_UNIT(R6),-(SP)      ; ...
        FD53 CF     01   FB   16D3          CALLS   #1,QIOMB_AST_CHECK       ; Check $QIO completion status
                              16D8          $DASSGN_S CHAN = AST_MBCHN       ; Free our hold on the mailbox
                 FE31    31   16E4          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_B_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_W_UNIT(R6),R7          ; ...
                              172B   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  RCV_DOSET:
                                175A  ;
                                175A  ; Some processor wants us to set a bit in the shared common event flag cluster.
                                175A  ; Set the bit and reply.
                                175A  ;
                                175A          $ASCEFC_S EFN = #COMEF_EFN,-      ; Associate with interprocessor cluster
                                175A                  NAME = MA_Q_COMEFNAM(R6)
          00000060 8F     C1    176F          ADDL3   #COMEF_EFN,-               ; Figure which bit to set in it
       57     110F'CF           1775                  AST_MBBUF+MB_K_MISC,R7
                                1779          $SETEF_S EFN = R7                  ; Set the requested bit
             110A'CF      91    1782          CMPB    AST_MBBUF+MB_B_SENDER,-    ; Did I ask myself to set the bit?
             01AC C6            1786                  MA_B_MYPORT(R6)
                  0D      13    1789          BEQL    10$                        ; BR if so - I'll expect cluster on return
                                178B          $DACEFC_S EFN = #COMEF_EFN         ; Release the cluster
                                1798  10$:
                                1798
             110A'CF      90    1798          MOVB    AST_MBBUF+MB_B_SENDER,-    ; The proc which sent now will receive
             110B'CF            179C                  AST_MBBUF+MB_B_RECEIVER
             01AC C6      90    179F          MOVB    MA_B_MYPORT(R6),-          ; I am the sender
             110A'CF            17A3                  AST_MBBUF+MB_B_SENDER
      110D'CF     06      B0    17A6          MOVW    #MSG_ISSET,AST_MBBUF+MB_W_WHY ; Say that we've set the bit
             0258 C6      28    17AB          MOVC3   MA_Q_COMMBNAM(R6),-        ; Form the logical name...
113D'CF      0260 C6            17AF                  MA_T_COMMBNAM(R6),AST_MBNAM ; ...of the other processor's MB
1135'CF      0258 C6      D0    17B5          MOVL    MA_Q_COMMBNAM(R6),AST_MBPTR ; Form descriptor of...
1139'CF      113D'CF      DE    17BC          MOVAL   AST_MBNAM,AST_MBPTR+4      ; ...the logical name
      50     110B'CF      D0    17C3          MOVL    AST_MBBUF+MB_B_RECEIVER,R0 ; Include the port number...
                FA66      30    17C8          BSBW    HEXUNIT                    ; ...
         FF A3    50      90    17CB          MOVB    R0,-1(R3)                  ; ...in the mailbox name
                                17CF          $CREMBX_S CHAN = AST_MBCHN,-       ; Establish a link to other proc's MB
                                17CF                  LOGNAM = AST_MBPTR
                                17E6          $QIO_S  CHAN = AST_MBCHN,-         ; Let the other proc know...
                                17E6                  FUNC = #IO$_WRITEVBLK!IO$M_NOW,-
                                17E6                  IOSB = AST_MBISB,-         ; ...that we've set the bit
                                17E6                  P1   = AST_MBBUF,-
                                17E6                  P2   = #MB_K_END
             7E   06      B0    1809          MOVW    #MSG_ISSET,-(SP)           ; Set up QIOMB_AST_CHECK parameter...
       7E    110B'CF      90    180C          MOVB    AST_MBBUF+MB_B_RECEIVER,-(SP) ; ...
       7E    01EC C6      90    1811          MOVB    MA_B_UNIT(R6),-(SP)        ; ...
     FC10 CF      01      FB    1816          CALLS   #1,QIOMB_AST_CHECK         ; Check $QIO completion status
                                181B          $DASSGN_S CHAN = AST_MBCHN         ; Free our hold on the mailbox
                FCEE      31    1827          BRW     EXIT_FROM_RCV              ; Exit via common routine
```

```
                          182A  RCV_ISSET:
                          182A  ;
                          182A  ; A processor has let us know that a bit which we wanted to be set has been
                          182A  ; set.  We really don't need to do anything about that here, but the
                          182A  ; synchronous code testing the shared common event flag cluster needs to know
                          182A  ; and is hibernating while waiting to be told.  Let the synchronous code know
                          182A  ; that the bit has been set by modifying the synchronous mailbox buffer to the
                          182A  ; extent of changing the reason for the mailbox.  Awaken the synchronous code.
                          182A  ;
        50   110A'CF   9A 182A          MOVZBL  AST_MBBUF+MB_B_SENDER,R0 ; Find out who sent us a message
     0C 01E4 C6    50  E4 182F          BBSC    R0,MA_L_DOSET(R6),10$   ; BR if we expected it...
        50   01    50  9C 1835          ROTL    R0,#1,R0                ; ...else...
        01C8 C6    50  C8 1839          BISL2   R0,MA_L_NOCAUSE(R6)     ; ...log the error
                 FCD7  31 183E          BRW     EXIT_FROM_RCV           ; Doing the rest could screw us up!
                          1841  10$:
   57   10   10    05  F0 1841          INSV    #MSG_DOSET,#16,#16,R7   ; We've gotten a reply to our...
57 08   08   110A'CF   F0 1846          INSV    AST_MBBUF+MB_B_SENDER,#8,#8,R7 ; ...request for setting an EF
        57   01EC C6   90 184D          MOVB    MA_W_UNIT(R6),R7        ; ...
                          1852          $CANTIM_S REQIDT = R7           ; ...so we need not worry time out
                          185D
                   06  B0 185D          MOVW    #MSG_ISSET,-            ; Change the reason for the mailbox...
              10B3'CF     185F                  SYNCH_MBBUF+MB_W_WHY    ; ...in the synchronous buffer
                          1862          $WAKE_S                         ; Get the synchronous code moving again
                 FCA8  31 186D          BRW     EXIT_FROM_RCV           ; Exit via common routine
```

```
                                1870  RCV_DOCLEAR:
                                1870  ;
                                1870  ; Some processor has requested that we clear a bit in the common event flag
                                1870  ; cluster.  Clear the bit and reply.
                                1870  ;
                                1870          $ASCEFC_S EFN = #COMEF_EFN,-        ; Associate with interprocessor cluster
                                1870                  NAME = MA_Q_COMEFNAM(R6)
      00000060 8F       C1      1885          ADDL3   #COMEF_EFN,-                ; Figure which bit to clear in it
          57   110F'CF          188B                  AST_MBBUF+MB_K_MISC,R7
                                188F          $CLREF_S EFN = R7                   ; Clear the requested bit
             110A'CF    91      1898          CMPB    AST_MBBUF+MB_B_SENDER,-     ; Did I ask myself to clear the bit?
             01AC C6            189C                  MA_B_MYPORT(R6)
                  0D    13      189F          BEQL    10$                         ; BR if so - I'll expect cluster on return
                                18A1          $DACEFC_S EFN = #COMEF_EFN          ; Release the cluster
                                18AE  10$:
                                18AE
             110A'CF    90      18AE          MOVB    AST_MBBUF+MB_B_SENDER,-     ; The proc which sent now will receive
             110B'CF            18B2                  AST_MBBUF+MB_B_RECEIVER
             01AC C6    90      18B5          MOVB    MA_B_MYPORT(R6),-           ; I am the sender
             110A'CF            18B9                  AST_MBBUF+MB_B_SENDER
                  08    B0      18BC          MOVW    #MSG_ISCLEAR,-              ; Say that the bit is cleared
             110D'CF            18BE                  AST_MBBUF+MB_W_WHY
             0258 C6    28      18C1          MOVC3   MA_Q_COMMBNAM(R6),-         ; Form the logical name...
     113D'CF 0260 C6            18C5                  MA_T_COMMBNAM(R6),AST_MBNAM ; ...of the other processor's MB
     1135'CF 0258 C6    D0      18CB          MOVL    MA_Q_COMMBNAM(R6),AST_MBPTR ; Form descriptor of...
     1139'CF 113D'CF    DE      18D2          MOVAL   AST_MBNAM,AST_MBPTR+4       ; ...the logical name
          50 110B'CF    D0      18D9          MOVL    AST_MBBUF+MB_B_RECEIVER,R0  ; Include the port number...
                F950    30      18DE          BSBW    HEXUNIT                     ; ...
          FF A3   50    90      18E1          MOVB    R0,-1(R3)                   ; ...in the mailbox name
                                18E5          $CREMBX_S CHAN = AST_MBCHN,-        ; Establish a link to other proc's MB
                                18E5                  LOGNAM = AST_MBPTR
                                18FC          $QIO_S  CHAN = AST_MBCHN,-          ; Let the other proc know...
                                18FC                  FUNC = #IO$_WRITEVBLK!IO$M_NOW,-
                                18FC                  IOSB = AST_MBISB,-          ; ...that the bit is cleared
                                18FC                  P1   = AST_MBBUF,-
                                18FC                  P2   = #MB_K_END
          7E      08    B0      191F          MOVW    #MSG_ISCLEAR,-(SP)          ; Set up QIOMB_AST_CHECK parameter...
          7E 110B'CF    90      1922          MOVB    AST_MBBUF+MB_B_RECEIVER,-(SP) ; ...
          7E 01EC C6    90      1927          MOVB    MA_Q_UNIT(R6),-(SP)         ; ...
        FAFA CF   01    FB      192C          CALLS   #1,QIOMB_AST_CHECK          ; Check $QIO completion status
                                1931          $DASSGN_S CHAN = AST_MBCHN          ; Free our hold on the mailbox
                FBD8    31      193D          BRW     EXIT_FROM_RCV               ; Exit via common routine
```

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

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

```
                            19CC  RCV_ME2ME:
                            19CC  ;
                            19CC  ; We've tested that it's possible to send mail to our own processor.  To let
                            19CC  ; the synchronous code know that we've been successful, change the reason for
                            19CC  ; the mailbox (MB_W_WHY) in the synchronous mailbox buffer and wake up the
                            19CC  ; synchronous code, which has been waiting for us to do exactly that.
                            19CC  ;
       57   10   10   0A  F0 19CC          INSV    #MSG_ME2ME,#16,#16,R7        ; We've gotten our own mailbox..
  57   08   08   01AC C6  F0 19D1          INSV    MA_B_MYPORT(R6),#8,#8,R7     ; ...
            57   01EC C6  90 19D8          MOVB    MA_W_UNIT(R6),R7             ; ...
                             19DD          $CANTIM_S REQIDT = R7                ; ...so we need not worry time out
                             19E8
                      0B  B0 19E8          MOVW    #MSG_ME2MEDONE,-             ; Change the reason for the mailbox...
                 10B3'CF     19EA                  SYNCR_MBBUF+MB_W_WHY         ; ...in the synchronous buffer
                             19ED          $WAKE_S                              ; Get the synchronous code moving again
                    FB1D  31 19F8          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  RCV_BYEBYE:
                               1A15  ;
                               1A15  ; Some processor is terminating execution.  We'd like to get consistent so that
                               1A15  ; there is minimal impact due to its going away.  In particular, we no longer
                               1A15  ; expect any mailbox messages from that processor.
                               1A15  ;
              110A'CF    9C    1A15          ROTL    AST_MBBUF+MB_B_SENDER,- ; Figure out who's going awa,
              50   01          1A19                  #1,R0
         01DC C6   50    CA    1A1B          BICL2   R0,MA_L_HELLO(R6)       ; We no longer expect any replies...
         01E0 C6   50    CA    1A20          BICL2   R0,MA_L_ICU_UCME(R6)    ; ...
         01E4 C6   50    CA    1A25          BICL2   R0,MA_L_DOSET(R6)       ; ...
         01E8 C6   50    CA    1A2A          BICL2   R0,MA_L_DOCLEAR(R6)     ; ...
  57   08   08  110A'CF  F0    1A2F          INSV    AST_MBBUF+MB_B_SENDER,#8,#8,R7 ; Insert stuff that...
            57  01EC C6  90    1A36          MOVB    MA_Q_UNIT(R6),R7        ; ...will be the same for all $CANTIMs
                               1A3B
  57   10   10   01      F0    1A3B          INSV    #MSG_HELLO,#16,#16,R7   ; Cancel critical timer requests...
                               1A40          $CANTIM_S REQIDT = R7           ; ...for the departing processor
                               1A4B
  57   10   10   02      F0    1A4B          INSV    #MSG_ICU_UCME,#16,#16,R7
                               1A50          $CANTIM_S REQIDT = R7
                               1A5B
                               1A5B          $GETJPI_S ITMLST = GETJPI_ITMLST ; Get the current timer queue count
              1168'CF    DD    1A70          PUSHL   TQCNT                   ; Save it for later comparison
  57   10   10   05      F0    1A74          INSV    #MSG_DOSET,#16,#16,R7
                               1A79          $CANTIM_S REQIDT = R7
                               1A84          $GETJPI_S ITMLST = GETJPI_ITMLST ; Get the new timer queue count
       8E     1168'CF    D1    1A99          CMPL    TQCNT,(SP)+             ; Has it changed?
                   0B    13    1A9E          BEQL    10$                     ; BR if not - no request outstanding
                               1AA0          $WAKE_S                         ; We have a $HIBER pending
                               1AAB  10$:
                               1AAB
                               1AAB          $GETJPI_S ITMLST = GETJPI_ITMLST ; Get the current timer queue count
              1168'CF    DD    1AC0          PUSHL   TQCNT                   ; Save it for later comparison
  57   10   10   07      F0    1AC4          INSV    #MSG_DOCLEAR,#16,#16,R7
                               1AC9          $CANTIM_S REQIDT = R7
                               1AD4          $GETJPI_S ITMLST = GETJPI_ITMLST ; Get the new timer queue count
       8E     1168'CF    D1    1AE9          CMPL    TQCNT,(SP)+             ; Has it changed?
                   0B    13    1AEE          BEQL    20$                     ; BR if not - no request outstanding
                               1AF0          $WAKE_S                         ; We have a $HIBER pending
                               1AFB  20$:
                               1AFB
                 FA1A    31    1AFB          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 $SETIMR expiration.
                                  1AFE  3247  ;
                                  1AFE  3248  ; INPUT PARAMETERS:
                                  1AFE  3249  ;       04(AP) = .BYTE  shared memory unit number (MA_W_UNIT)
                                  1AFE  3250  ;                .BYTE  other processor's port on that memory (MB_B_RECEIVER)
                                  1AFE  3251  ;                .WORD  reason for sending mail (MB_W_WHY)
                                  1AFE  3252  ;
                                  1AFE  3253  ; IMPLICIT INPUTS:
                                  1AFE  3254  ;       NONE
                                  1AFE  3255  ;
                                  1AFE  3256  ; OUTPUT PARAMETERS:
                                  1AFE  3257  ;       NONE
                                  1AFE  3258  ;
                                  1AFE  3259  ; IMPLICIT OUTPUTS:
                                  1AFE  3260  ;       NONE
                                  1AFE  3261  ;
                                  1AFE  3262  ; COMPLETION CODES:
                                  1AFE  3263  ;       NONE
                                  1AFE  3264  ;
                                  1AFE  3265  ; SIDE EFFECTS:
                                  1AFE  3266  ;       NONE
                                  1AFE  3267  ;
                                  1AFE  3268  ;--
                                  1AFE  3269
                                  1AFE  3270  MB_TIMEOUT_AST:
                           0FFC   1AFE  3271          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                  1B00  3272
            6D    1F17'CF    DE   1B00  3273          MOVAL   SSERROR,(FP)                ; Set up AST level exception handler
 56  01D8'CF   000001D8'8F   C1   1B05  3274          ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; Must search for correct node
                                  1B0F  3275  10$:
       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   000001D8'8F    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          CASEW   06(AP),#1,#MAX_MSG_TYPE ; Dispatch based on message type
                                  1B30          .MACRO  X       MBCODE,DUMMY    ; A macro will define dispatch address
                                  1B30                  .WORD   TMO_'MBCODE-30$
                                  1B30          .ENDM   X
                                  1B30  30$:
                                  1B30          .SHOW   MEB
                                  1B30          MAILBOX_MSG_TYPES               ; Generate dispatch table
                           0107'  1B30                  .WORD   TMO_HELLO-30$
                           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                  .WORD   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          .NOSHOW MEB
                                  1B4A  40$:
                                  1B4A          .MACRO  X       MBCODE,SECONDS  ; A macro to define unused labels
                                  1B4A                  .IF B SECONDS
                                  1B4A                          TMO_'MBCODE = 40$
                                  1B4A                  .ENDC ; B SECONDS
                                  1B4A          .ENDM   X
                                  1B4A          MAILBOX_MSG_TYPES               ; Generate labels
              06D2'CF       DF    1B4A          PUSHAL  BAD_REQIDT              ; If we reach here...
                   01       DD    1B4E          PUSHL   #1                      ; ...we got a phoney AST
          00741130 8F       DD    1B50          PUSHL   #UETP$_TEXT!STS$K_WARNING
      00000000'GF  03       FB    1B56          CALLS   #3,G^LIB$SIGNAL
                                  1B5D          ;BRW    EXIT_FROM_TMO           ; Fall into common routine for exit
                                  1B5D
                                  1B5D  ;
                                  1B5D  ; All the routines dispatched via the above CASEW will exit through this little
                                  1B5D  ; routine.
                                  1B5D  ;
                                  1B5D  EXIT_FROM_TMO:
            06 AC   0A      B1    1B5D          CMPW    #MSG_ME2ME,06(AP)       ; Was this a single processor message?
                    14      13    1B61          BEQL    10$                     ; BR if so - GS may not be there
            06 AC   0B      B1    1B63          CMPW    #MSG_ME2MEDONE,06(AP)   ; Catch pathological case
                    0E      13    1B67          BEQL    10$                     ; BR if so - again GS may not be there
          57      062C C6   D0    1B69          MOVL    MA_Q_COMGSRAD(R6),R7    ; Point to start of global section
          58        05 AC   9A    1B6E          MOVZBL  05(AP),R8
         00 24 A7   58      E7    1B72          BBCCI   R8,GS_L_AVAILABLE(R7),10$ ; Assume receiver processor went away
                                  1B77  10$:
              01AC C6       90    1B77          MOVB    MA_B_MYPORT(R6),-       ; Send a MB so saying - sender...
              110A'CF             1B7B                  AST_MBBUF+MB_B_SENDER
                05 AC       90    1B7E          MOVB    05(AP),-                ; ...receiver...
              110B'CF             1B81                  AST_MBBUF+MB_B_RECEIVER ; ...(the one we think is gone)...
              01EC C6       90    1B84          MOVB    MA_Q_UNIT(R6),-         ; ...the shared memory involved...
              110C'CF             1B88                  AST_MBBUF+MB_B_UNIT
                   0C       B0    1B8B          MOVW    #MSG_ILOSTU,-           ; ...the reason for the mailbox...
              110D'CF             1B8D                  AST_MBBUF+MB_W_WHY
                06 AC       B0    1B90          MOVW    06(AP),-                ; ...and what we originally sent
              110F'CF             1B93                  AST_MBBUF+MB_K_MISC
```

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

```
                         1C1D  ;
                         1C1D  ; Save some code on this page by taking advantage of the simplicity of these
                         1C1D  ; routines.  DOSET, DOCLEAR and ME2ME need to awaken us from hibernation.
                         1C1D  ; The others need only say that certain events are no longer expected;
                         1C1D  ; everything else that needs to be done is handled by the common exit routine.
                         1C1D  ;
                         1C1D
                         1C1D
                         1C1D  TMO_DOSET:
                         1C1D  ;
                         1C1D  ; Some processor didn't set a bit in the shared common event flag.
                         1C1D  ;
01E4 C6     50    CA     1C1D          BICL2   R0,MA_L_DOSET(R6)       ; Don't expect flag to be set
            05    11     1C22          BRB     COMMON_WAKE_TMO
                         1C24
                         1C24
                         1C24  TMO_DOCLEAR:
                         1C24  ;
                         1C24  ; Some processor didn't clear a bit in the shared common event flag.
                         1C24  ;
01E8 C6     50    CA     1C24          BICL2   R0,MA_L_DOCLEAR(R6)     ; Don't expect flag to be cleared
                         1C29          ;BRB    COMMON_WAKE_TMO
                         1C29
                         1C29
                         1C29  TMO_ME2ME:
                         1C29  ;
                         1C29  ; I couldn't send mail to myself.
                         1C29  ;
                         1C29
                         1C29  COMMON_WAKE_TMO:
                         1C29          $WAKE_S                         ; Get the test going again
          FF26    31     1C34          BRW     EXIT_FROM_TMO
                         1C37
                         1C37
                         1C37  TMO_HELLO:
                         1C37  ;
                         1C37  ; Some processor didn't respond to our friendly hello.
                         1C37  ;
01DC C6     50    CA     1C37          BICL2   R0,MA_L_HELLO(R6)       ; No longer expect other processor
          FF1E    31     1C3C          BRW     EXIT_FROM_TMO
                         1C3F
                         1C3F
                         1C3F  TMO_ICU_UCME:
                         1C3F  ;
                         1C3F  ; Some processor didn't let us know that we could see it, too.
                         1C3F  ;
01E0 C6     50    CA     1C3F          BICL2   R0,MA_L_ICU_UCME(R6)    ; No longer expect other processor
          FF16    31     1C44          BRW     EXIT_FROM_TMO
```

```
                                      1C47        .SBTTL  Print Error Summary Routine
                                      1C47  ;++
                                      1C47  ; FUNCTIONAL DESCRIPTION:
                                      1C47  ;       This routine is called at the end of a thre-minute test iteration or
                                      1C47  ;       one shot pass to print whatever information we may have stored
                                      1C47  ;       regarding errors during the test.
                                      1C47  ;
                                      1C47  ; CALLING SEQUENCE:
                                      1C47  ;       CALLS   #0,ERROR_SUMMARY
                                      1C47  ;
                                      1C47  ; INPUT PARAMETERS:
                                      1C47  ;       NONE
                                      1C47  ;
                                      1C47  ; IMPLICIT INPUTS:
                                      1C47  ;       UNIT_LIST points to the list of nodes, one for each memory.  Within
                                      1C47  ;       each node, there are several mask words, each of which corresponds to
                                      1C47  ;       a particular condition.  The bits in each mask correspond to the ports
                                      1C47  ;       of the shared memory attached to processors.  A bit is lit if the
                                      1C47  ;       condition applies to that processor.
                                      1C47  ;
                                      1C47  ; OUTPUT PARAMETERS:
                                      1C47  ;       NONE
                                      1C47  ;
                                      1C47  ; IMPLICIT OUTPUTS:
                                      1C47  ;       NONE
                                      1C47  ;
                                      1C47  ; COMPLETION CODES:
                                      1C47  ;       NONE
                                      1C47  ;
                                      1C47  ; SIDE EFFECTS:
                                      1C47  ;       Messages to SYS$OUTPUT.  Masks described above are cleared out.
                                      1C47  ;
                                      1C47  ;--
                                      1C47
                                      1C47  ERROR_SUMMARY:
                                OFFC  1C47        .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                      1C49
 56   01D8'CF   000001D8'8F     C1    1C49        ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; R6 points to node for current memory
                                      1C53  10$:
                   2C    00     3B    1C53        SKPC    #0,#MA_K_ERRLEN,-        ; See if we have...
                   01B0  C6           1C56                MA_L_ERRBLK(R6)          ; ...anything to report for this memory
                         03     12    1C59        BNEQ    20$                      ; BR if we found a problem
                       007B     31    1C5B        BRW     100$                     ; BR if it's clean
                                      1C5E  20$:
           50   07A0'CF         DE    1C5E        MOVAL   SUMMARY_HEADER,R0        ; We've some error, print a header
                       F624     30    1C63        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          .SBTTL  Test Cleanup Routine
            1D54  ;++
            1D54  ; FUNCTIONAL DESCRIPTION:
            1D54  ;       This routine is called by the exit handler to ensure that all objects
            1D54  ;       which use system resources (e.g., global sections in shared memory)
            1D54  ;       are all properly disposed of and that we are consistent with respect to
            1D54  ;       other processors.
            1D54  ;
            1D54  ; CALLING SEQUENCE:
            1D54  ;       CALLS   #0,CLEANUP
            1D54  ;
            1D54  ; INPUT PARAMETERS:
            1D54  ;       NONE
            1D54  ;
            1D54  ; IMPLICIT INPUTS:
            1D54  ;       UNIT_LIST points to the list of nodes, one for each memory.  Within
            1D54  ;       node, there is a pointer to the interprocessor global section for that
            1D54  ;       memory as well as miscellaneous status info.
            1D54  ;
            1D54  ; OUTPUT PARAMETERS:
            1D54  ;       NONE
            1D54  ;
            1D54  ; IMPLICIT OUTPUTS:
            1D54  ;       NONE
            1D54  ;
            1D54  ; COMPLETION CODES:
            1D54  ;       NONE
            1D54  ;
            1D54  ; SIDE EFFECTS:
            1D54  ;       Remove as many traces as possible of our existence.
            1D54  ;
            1D54  ;--
            1D54
            1D54  CLEANUP:
      0FFC  1D54          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
            1D56
            1D56  ;+
            1D56  ;       The following objects (using system resources) are created during the
            1D56  ;       execution of the shared memory test:  one global section per memory
            1D56  ;       accessed by all processors connected to that memory, one global section
            1D56  ;       per memory accessed only by the current processor, the files associated
            1D56  ;       with those global sections, one mailbox per memory associated with the
            1D56  ;       current processor, other mailboxes in each memory associated with
            1D56  ;       processors connected to that memory, one common event flag cluster
            1D56  ;       per memory accessed by all processors connected to that memory, and one
            1D56  ;       common event flag cluster per memory accessed only by the current
            1D56  ;       processor.  Of those items, only the global sections and the associated
            1D56  ;       files are not automatically deleted or marked for deletion during image
            1D56  ;       rundown; the others have a "temporary" existence.
            1D56  ;
            1D56  ;       The interprocessor global section may not have actually been created by
            1D56  ;       this processor.  In that case, we cannot delete it, but only delete its
            1D56  ;       address space.  The file that was created in our attempt to create the
            1D56  ;       section still must be deleted, however.
            1D56  ;
            1D56  ;       Even though we may be able to try to delete the interprocess global
            1D56  ;       section, it may only be marked for deletion because other processors
```

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

```
          01AC C6   90  1DAE        MOVB    MA_B_MYPORT(R6),-              ; Start filling in message...
          110A'CF       1DB2                AST_MBBUF+MB_B_SENDER          ; ...sender's port number...
   110B'CF     5A   90  1DB5        MOVB    R10,AST_MBBUF+MB_B_RECEIVER   ; ...receiver's port number...
          01EC C6   90  1DBA        MOVB    MA_W_UNIT(R6),-                ; ...unit of the memory we share...
          110C'CF       1DBE                AST_MBBUF+MB_B_UNIT
               0D   B0  1DC1        MOVW    #MSG_BYEBYE,-                  ; ...reason for the message
          110D'CF       1DC3                AST_MBBUF+MB_W_WHY
          0258 C6   28  1DC6        MOVC3   MA_Q_COMMBNAM(R6),-            ; Form the logical name...
  113D'CF 0260 C6       1DCA                MA_T_COMMBNAM(R6),AST_MBNAM   ; ...of other processor's MB
  1135'CF 0258 C6   D0  1DD0        MOVL    MA_Q_COMMBNAM(R6),AST_MBPTR   ; Form descriptor to...
  1139'CF 113D'CF   DE  1DD7        MOVAL   AST_MBNAM,AST_MBPTR+4         ; ...the logical name
          50     5A   D0  1DDE      MOVL    R10,R0                        ; Include the port number...
              F44D   30  1DE1       BSBW    HEXUNIT                       ; ...
       FF A3    50   90  1DE4       MOVB    R0,-1(R3)                     ; ...in the name
                         1DE8       $CREMBX_S CHAN = AST_MBCHN,-          ; Establish a link to other proc's MB
                         1DE8               LOGNAM = AST_MBPTR
                         1DFF       $QIO_S  CHAN = AST_MBCHN,-            ; Say bye to the other processor
                         1DFF               FUNC = #IO$_WRITEVBLK!IO$M_NOW,-
                         1DFF               IOSB = AST_MBISB,-            ; NOTE: We ignore errors!
                         1DFF               P1   = AST_MBBUF,-
                         1DFF               P2   = #MB_K_END
                         1E22       $DASSGN_S CHAN = SYNCH_MBCHN          ; We no longer need the MB
    5B    5A    58   C3  1E2E       SUBL3   R8,R10,R11                    ; Get number of bits we searched
                5B   D6  1E32       INCL    R11                           ; Correct off-by-one (inclusive search)
          59    5B   C2  1E34       SUBL2   R11,R9                        ; Adjust number of bits left to search
    58    5A    01   C1  1E37       ADDL3   #1,R10,R8                     ; Point to new first bit to search
              FF65   31  1E3B       BRW     90$                           ; Loop thru all procs on this memory
                         1E3E 110$:
                         1E3E ;
                         1E3E ; Only update or delete the interprocessor global section if we were the
                         1E3E ; processor which created it.  Delete the file regardless.
                         1E3E ;
       58    20 A7   90  1E3E       MOVB    GS_L_OWNER(R7),R8             ; Save a copy of the GS creator's port
     01AC C6    58   91  1E42       CMPB    R8,MA_B_MYPORT(R6)            ; Did we create the section?
                20   12  1E47       BNEQ    120$                          ; BR if not - can't update it
                         1E49       $UPDSEC_S INADR = MA_Q_COMGSRAD(R6),- ; Updating the section now...
                         1E49               EFN   = #COMGS_EFN            ; ...means we avoid implicit update
                         1E60       $WAITFR_S EFN = #COMGS_EFN            ; Let the $UPDSEC finish in peace
                         1E69 120$:
                         1E69       $DELTVA_S INADR = MA_Q_COMGSRAD(R6)  ; Remove GS virtual address space
     01AC C6    58   91  1E78       CMPB    R8,MA_B_MYPORT(R6)            ; Did we create the section?
                0F   12  1E7D       BNEQ    130$                          ; BR if not - can't delete it
                         1E7F       $DGBLSC_S GSDNAM = MA_Q_COMGSNAM(R6) ; Delete the section...
                         1E8E 130$:
                         1E8E       $DASSGN_S CHAN = MA_K_COMFAB+FAB$L_STV(R6) ; ...and unhook RMS
                         1E9A 140$:
        0112 C6     B5   1E9A       TSTW    MA_K_COMFAB+FAB$W_IFI(R6)     ; Is the file still open?
                0B   13  1E9E       BEQL    150$                          ; BR if not
                         1EA0       $CLOSE  FAB = MA_K_COMFAB(R6)         ; Close it if it is
                         1EAB 150$:
                         1EAB       $ERASE  FAB = MA_K_COMFAB(R6)         ; Get rid of the associated file
                         1EB6                                             ; (We can swallow RMS-W-IFI errors)
                         1EB6 ;
                         1EB6 ; We can just blindly get rid of the single processor global section because no
                         1EB6 ; one else knows of it.  We must, however, wait to catch the situation where we
                         1EB6 ; reach this point but an $UPDSEC is still in progress.  See the note where
                         1EB6 ; SPRGS_EFN is defined.  If the section is already removed, there should be
```

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

```
                         1F0F          .SBTTL  Test Timer Expiration Routine
                         1F0F     ;++
                         1F0F     ; FUNCTIONAL DESCRIPTION:
                         1F0F     ;       This routine runs when the timer to mark the end of the test goes off.
                         1F0F     ;
                         1F0F     ; CALLING SEQUENCE:
                         1F0F     ;       Called via AST at $SETIMR expiration.
                         1F0F     ;
                         1F0F     ; INPUT PARAMETERS:
                         1F0F     ;       NONE
                         1F0F     ;
                         1F0F     ; IMPLICIT INPUTS:
                         1F0F     ;       NONE
                         1F0F     ;
                         1F0F     ; OUTPUT PARAMETERS:
                         1F0F     ;       NONE
                         1F0F     ;
                         1F0F     ; IMPLICIT OUTPUTS:
                         1F0F     ;       NONE
                         1F0F     ;
                         1F0F     ; COMPLETION CODES:
                         1F0F     ;       NONE
                         1F0F     ;
                         1F0F     ; SIDE EFFECTS:
                         1F0F     ;       Sets a flag to indicate timer expiration.
                         1F0F     ;
                         1F0F     ;--
                         1F0F
                         1F0F     TIME_OUT:
                   0FFC  1F0F             .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                         1F11
   0002'CF    02     A8  1F11             BISW2   #TEST_OVERM,FLAG          ; Indicate end of pass or of test
                     04  1F16             RET
```

```
1F17          .SBTTL  System Service Exception Handler
1F17    ;++
1F17    ; FUNCTIONAL DESCRIPTION:
1F17    ;       This routine is executed if a software or hardware exception occurs or
1F17    ;       if a LIB$SIGNAL system service is used to output a message.
1F17    ;
1F17    ; CALLING SEQUENCE:
1F17    ;       Entered via an exception from the system
1F17    ;
1F17    ; INPUT PARAMETERS:
1F17    ;       ERROR_COUNT    = previous cumulative error count
1F17    ;                        -----------------
1F17    ;          AP ---->      !       2       !
1F17    ;                        !---------------!
1F17    ;                        ! SIGNL ARY PNT !
1F17    ;                        !---------------!
1F17    ;                        ! MECH  ARY PNT !
1F17    ;                        !---------------!   ---------
1F17    ;                        !       4       !           ^
1F17    ;                        !---------------!           !
1F17    ;                        ! ESTABLISH FP  !           !
1F17    ;                        !---------------!           !
1F17    ;                        !     DEPTH     !   Mechanism Array
1F17    ;                        !---------------!           !
1F17    ;                        !      R0       !           !
1F17    ;                        !---------------!           !
1F17    ;                        !      R1       !           v
1F17    ;                        !---------------!   ---------
1F17    ;                        !       N       !           ^
1F17    ;                        !---------------!           !
1F17    ;                        !CONDITION NAME !           !
1F17    ;                        !---------------!           !
1F17    ;                        ! N-3 ADDITIONAL!     Signal Array
1F17    ;                        ! LONG WORD ARGS!           !
1F17    ;                        !---------------!           !
1F17    ;                        !      PC       !           !
1F17    ;                        !---------------!           !
1F17    ;                        !      PSL      !           v
1F17    ;                        !---------------!   ---------
1F17    ; IMPLICIT INPUTS:
1F17    ;       NONE
1F17    ;
1F17    ; OUTPUT PARAMETERS:
1F17    ;       NONE
1F17    ;
1F17    ; IMPLICIT OUTPUTS:
1F17    ;       NONE
1F17    ;
1F17    ; COMPLETION CODES:
1F17    ;       SS$_NORMAL if it's a UETP condition or RMS error.
1F17    ;       Error status from exception, otherwise.
1F17    ;
1F17    ; SIDE EFFECTS:
1F17    ;       May branch to ERROR_EXIT.
1F17    ;       May print a message.
1F17    ;--
1F17
```

```
                          1F17  SSERROR:
                   0FFC   1F17          .WORD    ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                          1F19
                          1F19          $SETAST_S ENBFLG = #0           ; Disable AST delivery
              01     DD   1F22          PUSHL    #1                     ; Assume ASTs were enabled
        50    09     D1   1F24          CMPL     S^#SS$_WASSET,R0       ; Were ASTs enabled?
              02     13   1F27          BEQL     10$                    ; BR if they were
              6E     D4   1F29          CLRL     (SP)                   ; Set ASTs to remain disabled
                          1F2B  10$:
                          1F2B          $SETSFM_S ENBFLG = #0           ; Disable SS failure mode
              01     DD   1F34          PUSHL    #1                     ; Assume SS failure mode was enabled
        50    09     D1   1F36          CMPL     S^#SS$_WASSET,R0       ; Was SS failure mode enabled?
              02     13   1F39          BEQL     20$                    ; BR if it was
              6E     D4   1F3B          CLRL     (SP)                   ; Set SS failure mode to remain off
                          1F3D  20$:
     56    04 AC     D0   1F3D          MOVL     CHF$L_SIGARGLST(AP),R6 ; Get the signal array pointer
     59    04 A6     7D   1F41          MOVQ     CHF$L_SIG_NAME(R6),R9  ; Get NAME in R9 and ARG1 in R10
              10     ED   1F45          CMPZV    #STS$V_FAC_NO,-        ; Is this a message from LIB$SIGNAL?
              0C          1F47                   #STS$S_FAC_NO,-
 00000074 8F  59          1F48                   R9,#UETP$_FACILITY
              14     12   1F4E          BNEQ     30$                    ; BR if this is not a UETP exception
        66    02     C2   1F50          SUBL2    #2,CHF$L_SIG_ARGS(R6)  ; Drop the PC and PSL
                          1F53          $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6) ; Print the message
              21     11   1F62          BRB      40$                    ; Restore ASTs and SS fail mode
                          1F64  30$:
 59   0000045C 8F    D1   1F64          CMPL     #SS$_SSFAIL,R9         ; RMS failures are SysSvc failures
              32     12   1F6B          BNEQ     50$                    ; BR if this can't be an RMS failure
              10     ED   1F6D          CMPZV    #STS$V_FAC_NO,-        ; Is it an RMS failure?
              0C          1F6F                   #STS$S_FAC_NO,-
        01    5A          1F70                   R10,#RMS$_FACILITY
              2B     12   1F72          BNEQ     50$                    ; BR if not
 5A   F0000000 8F    CA   1F74          BICL2    #^XF0000000,R10        ; Strip control bits from status code
        08 A6 04     39   1F7B          MATCHC   #4,CHF$L_SIG_ARG1(R6),- ; Is it an RMS failure for which...
              14          1F7F                   #NRAT_LENGTH,-
       004D'CF            1F80                   NO_RMS_AST_TABLE       ; ...no AST can be delivered?
              1A     13   1F83          BEQL     50$                    ; BR if so - must give error here
                          1F85  40$:
              01     BA   1F85          POPR     #^M<R0>                ; Restore SS failure mode...
                          1F87          $SETSFM_S ENBFLG = R0           ; ...
              01     BA   1F90          POPR     #^M<R0>                ; Restore AST enable...
                          1F92          $SETAST_S ENBFLG = R0           ; ...
        50    01     D0   1F9B          MOVL     S^#SS$_NORMAL,R0       ; Supply a standard status for exit
                     04   1F9E          RET                             ; Resume processing (or goto RMS_ERROR)
                          1F9F  50$:
     018A'CF  59     D0   1F9F          MOVL     R9,STATUS              ; Save the status
              58     D4   1FA4          CLRL     R8                     ; Assume for now it's not SS failure
 59   0000045C 8F    D1   1FA6          CMPL     #SS$_SSFAIL,R9         ; But is it a System Service failure?
              38     12   1FAD          BNEQ     70$                    ; BR if not - no special case message
                          1FAF          $GETMSG_S MSGID = R10,-         ; Get SS failure code associated text
                          1FAF                    MSGLEN = BUFFER_PTR,-
                          1FAF                    BUFADR = FAO_BUF,-
                          1FAF                    FLAGS  = #14,-
                          1FAF                    OUTADR = MSG_BLOCK
     01BB'CF  95          1FC6          TSTB     MSG_BLOCK+1            ; Get FAO arg count for SS failure code
              16     13   1FCA          BEQL     60$                    ; Don't use $GETMSG if no $FAO args...
     000C'CF  DF          1FCC          PUSHAL   BUFFER_PTR             ; ...else build up...
              01     DD   1FD0          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          .SBTTL  RMS Error Handler
                                 1FFA  ;++
                                 1FFA  ; FUNCTIONAL DESCRIPTION:
                                 1FFA  ;       This routine handles error returns from RMS calls.
                                 1FFA  ;
                                 1FFA  ; CALLING SEQUENCE:
                                 1FFA  ;       Called by RMS when a file processing error is found.
                                 1FFA  ;
                                 1FFA  ; INPUT PARAMETERS:
                                 1FFA  ;       Address of the FAB or RAB associated with the RMS call.
                                 1FFA  ;
                                 1FFA  ; IMPLICIT INPUTS:
                                 1FFA  ;       NONE
                                 1FFA  ;
                                 1FFA  ; OUTPUT PARAMETERS:
                                 1FFA  ;       NONE
                                 1FFA  ;
                                 1FFA  ; IMPLICIT OUTPUTS:
                                 1FFA  ;       Error message
                                 1FFA  ;
                                 1FFA  ; COMPLETION CODES:
                                 1FFA  ;       NONE
                                 1FFA  ;
                                 1FFA  ; SIDE EFFECTS:
                                 1FFA  ;       Program may exit, depending on severity of the error.
                                 1FFA  ;
                                 1FFA  ;--
                                 1FFA
                                 1FFA  RMS_ERROR:
                           0FFC  1FFA          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                 1FFC
           56    04 AC     D0    1FFC          MOVL    4(AP),R6                ; See whether we're dealing with...
              66    03     91    2000          CMPB    #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
                    16     12    2003          BNEQ    10$                     ; BR if it's a RAB
           57  01D4'CF     DE    2005          MOVAL   FILE,R7                 ; FAB-specific code:  text string...
              58    56     D0    200A          MOVL    R6,R8                   ; ...address of FAB...
                 0C A6     DD    200D          PUSHL   FAB$L_STV(R6)           ; ...STV field for error...
                 08 A6     DD    2010          PUSHL   FAB$L_STS(R6)           ; ...STS field for error...
       018A'CF   08 A6     D0    2013          MOVL    FAB$L_STS(R6),STATUS    ; ...and save the error code
                    15     11    2019          BRB     COMMON                  ; FAB and RAB share other code
                                 201B  10$:
           57  01E0'CF     DE    201B          MOVAL   RECORD,R7               ; RAB-specific code:  text string...
           58    3C A6     D0    2020          MOVL    RAB$L_FAB(R6),R8        ; ...address of associated FAB...
                 0C A6     DD    2024          PUSHL   RAB$L_STV(R6)           ; ...STV field for error...
                 08 A6     DD    2027          PUSHL   RAB$L_STS(R6)           ; ...STS field for error...
       018A'CF   08 A6     D0    202A          MOVL    RAB$L_STS(R6),STATUS    ; ...and save the error code
                                 2030  COMMON:
           5A    34 A8     9A    2030          MOVZBL  FAB$B_FNS(R8),R10       ; Get the file name size
                                 2034          $FAO_S  CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
                                 2034                  OUTLEN = BUFFER_PTR,-
                                 2034                  OUTBUF = FAO_BUF,-
                                 2034                  P1     = R7,-
                                 2034                  P2     = R10,-
                                 2034                  P3     = FAB$L_FNA(R8)
              000C'CF      DF    204E          PUSHAL  BUFFER_PTR              ; ...and arguments for ERROR_EXIT...
                    01     DD    2052          PUSHL   #1                      ; ...
         00741130 8F       DD    2054          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       .SBTTL  CTRL/C Handler
                              2069  ;++
                              2069  ; FUNCTIONAL DESCRIPTION:
                              2069  ;        This routine handles CTRL/C AST's
                              2069  ;
                              2069  ; CALLING SEQUENCE:
                              2069  ;        Called via AST
                              2069  ;
                              2069  ; INPUT PARAMETERS:
                              2069  ;        NONE
                              2069  ;
                              2069  ; IMPLICIT INPUTS:
                              2069  ;        NONE
                              2069  ;
                              2069  ; OUTPUT PARAMETERS:
                              2069  ;        NONE
                              2069  ;
                              2069  ; IMPLICIT OUTPUTS:
                              2069  ;        NONE
                              2069  ;
                              2069  ; COMPLETION CODES:
                              2069  ;        NONE
                              2069  ;
                              2069  ; SIDE EFFECTS:
                              2069  ;        NONE
                              2069  ;
                              2069  ;--
                              2069
                              2069  CCASTHAND:
                        0FFC  2069          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                              206B
             0082'CF    DF    206B          PUSHAL  CNTRLCMSG                   ; Set message pointer
                  01    DD    206F          PUSHL   #1                          ; Set arg count
         00741130 8F    DD    2071          PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
                  00    DD    2077          PUSHL   #0                          ; Indicate an abnormal termination
             00E4'CF    DF    2079          PUSHAL  PROCESS_NAME                ; ...
                  02    DD    207D          PUSHL   #2                          ; ...
         007410E0 8F    DD    207F          PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
   00000000'GF    07    FB    2085          CALLS   #7,G^LIB$SIGNAL             ; Output the message
                        D0    208C          MOVL    #<STS$M_INHIB_MSG!-         ; Set the exit status
                              208D                  SS$_CONTROLC-=
                              208D                  STS$K_SUCCESS+STS$K_WARNING>,-
018A'CF    10000650 8F        208D                  STATUS
                              2095          $EXIT_S STATUS                      ; Terminate program cleanly
```

```
                                    20A0        .SBTTL  Error Exit
                                    20A0  ;++
                                    20A0  ; FUNCTIONAL DESCRIPTION:
                                    20A0  ;         This routine prints an error message and exits.
                                    20A0  ;
                                    20A0  ; CALLING SEQUENCE:
                                    20A0  ;         MOVx  error status value,STATUS
                                    20A0  ;         PUSHx error specific information on the stack
                                    20A0  ;         PUSHL current argument count
                                    20A0  ;         BRW   ERROR_EXIT
                                    20A0  ;
                                    20A0  ; INPUT PARAMETERS:
                                    20A0  ;         Arguments to LIB$SIGNAL, as above
                                    20A0  ;
                                    20A0  ; IMPLICIT INPUTS:
                                    20A0  ;         NONE
                                    20A0  ;
                                    20A0  ; OUTPUT PARAMETERS:
                                    20A0  ;         Message to SYS$OUTPUT and SYS$ERROR
                                    20A0  ;
                                    20A0  ; IMPLICIT OUTPUTS:
                                    20A0  ;         Program exit
                                    20A0  ;
                                    20A0  ; COMPLETION CODES:
                                    20A0  ;         NONE
                                    20A0  ;
                                    20A0  ; SIDE EFFECTS:
                                    20A0  ;         NONE
                                    20A0  ;
                                    20A0  ;--
                                    20A0
                                    20A0  ERROR_EXIT:
                                    20A0
                                    20A0          $SETAST_S ENBFLG = #0           ; ASTs can play havoc with messages
      15 0002'CF    03       E0     20A9          BBS     #BEGIN_MSGV,FLAG,10$    ; BR if "begin" msg already printed
                    7E       D4     20AF          CLRL    -(SP)                   ; Set the time stamp flag
               000F'CF       DF     20B1          PUSHAL  TEST_NAME               ; Set the test name
                    02       DD     20B5          PUSHL   #2                      ; Push the argument count
           00741039 8F       DD     20B7          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
        00000000'GF    04    FB     20BD          CALLS   #4,G^LIB$SIGNAL         ; Print the startup message
                                    20C4  10$:
        01CE'CF   08   8E    C1     20C4          ADDL3   (SP)+,#8,ARG_COUNT      ; Get total # args, pop partial count
               0186'CF       D6     20CA          INCL    ERROR_COUNT             ; Keep running error count
                    00       DD     20CE          PUSHL   #0                      ; Push the time parameter
               00E4'CF       DF     20D0          PUSHAL  PROCESS_NAME            ; Push test name...
           000F0002 8F       DD     20D4          PUSHL   #^XF0002                ; ...arg count...
           007410E2 8F       DD     20DA          PUSHL   #UETP$_ABENDD!STS$K_ERROR ; ...and signal name
               0186'CF       DD     20E0          PUSHL   ERROR_COUNT             ; Finish off arg list...
               00E4'CF       DF     20E4          PUSHAL  PROCESS_NAME            ; ...
           00010002 8F       DD     20E8          PUSHL   #^X10002                ; ...
           00748022 8F       DD     20EE          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
    00000000'GF   01CE'CF    FB     20F4          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         .SBTTL  Exit Handler
                                2125  ;++
                                2125  ; FUNCTIONAL DESCRIPTION:
                                2125  ;       This routine handles cleanup at exit.  If the MODE logical name is
                                2125  ;       equated to "ONE", the routine will update the test flag in the
                                2125  ;       UETINIDEV.DAT file depending on the UETUNT$M_TESTABLE flag state in the
                                2125  ;       UETUNT$B_FLAGS field of the unit block for each unit for the device
                                2125  ;       under test.
                                2125  ;
                                2125  ; CALLING SEQUENCE:
                                2125  ;       Invoked automatically by $EXIT System Service.
                                2125  ;
                                2125  ; INPUT PARAMETERS:
                                2125  ;       STATUS  contains the exit status.
                                2125  ;       FLAG    has synchronizing bits.
                                2125  ;       DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV.
                                2125  ;
                                2125  ; IMPLICIT INPUTS:
                                2125  ;       UNIT_LIST points to the head of a doubly linked circular list of unit
                                2125  ;                 blocks for the device under test.
                                2125  ;
                                2125  ; OUTPUT PARAMETERS:
                                2125  ;       NONE
                                2125  ;
                                2125  ; IMPLICIT OUTPUTS:
                                2125  ;       Various files are de-accessed, the process name is reset, and any
                                2125  ;       necessary synchronization with UETPDEV01 is carried out.
                                2125  ;       If the MODE logical name is equated to "ONE", the routine will update
                                2125  ;       the test flag in the UETINIDEV.DAT file depending on the
                                2125  ;       UETUNT$M_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
                                2125  ;       block for each unit for the device under test.
                                2125  ;
                                2125  ; COMPLETION CODES:
                                2125  ;       NONE
                                2125  ;
                                2125  ; SIDE EFFECTS:
                                2125  ;       NONE
                                2125  ;
                                2125  ;--
                                2125
                                2125  EXIT_HANDLER:
                          0FFC  2125         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                2127
                                2127         $SETSFM_S ENBFLG = #0          ; Turn off System Service failure mode
                                2130         $SETAST_S ENBFLG = #0          ; We're finished - no more ASTs
      03 0002'CF     04     E0  2139         BBS     #ONE_SHOTV,FLAG,10$     ; If one-shot, update testability...
                   00B8     31  213F         BRW     END_UPDATE              ; ...else don't update UETINIDEV.DAT
                                2142  10$:
      03 0002'CF     02     E0  2142         BBS     #SAFE_TO_UPDV,FLAG,20$  ; Only update if it's safe
                   00AF     31  2148         BRW     END_UPDATE              ; Else forget it
                                214B  20$:
         5A   1250'CF       DE  214B         MOVAL   INI_RAB,R10             ; Set the RAB address
            1E AA    02     90  2150         MOVB    #RAB$C_RFA,RAB$B_RAC(R10) ; Set RFA mode
  10 AA   1294'CF    06     28  2154         MOVC3   #6,DDB_RFA,RAB$W_RFA(R10) ; Set RFA to DDB line
                                215B         $GET    RAB = (R10)             ; Go back to the DDB record
              75 50         E9  2164         BLBC    R0,UPDATE_FAILED        ; If failure then forget it
            1E AA    00     90  2167         MOVB    #RAB$C_SEQ,RAB$B_RAC(R10) ; Set back to sequential mode
```

```
    5B  01D8'CF  000001D8'8F  C1  216B             ADDL3   #UNIT_LIST,UNIT_LIST,R11 ; Set the unit block list header
                           59 D4  2175             CLRL    R9                       ; Init a counter
                              2177  UNIT_LOOP:
                           01 E1  2177             BBC     #UETUNT$V_TESTABLE,-     ; BR if this unit is not testable
                     02 0B AB                              UETUNT$B_FLAGS(R11),10$
                           59 D6  217C             INCL    R9                       ; Count testable units
                              217E  10$:
                      5B   6B C0  217E             ADDL2   (R11),R11                ; Next unit block
           000001D8'8F     5B D1  2181             CMPL    R11,#UNIT_LIST           ; Are we full circle in the list?
                           ED 12  2188             BNEQ    UNIT_LOOP                ; BR if not
                           59 D5  218A             TSTL    R9                       ; Any testable units?
                           12 12  218C             BNEQ    20$                      ; BR if yes...
              0018'CF   4E 8F 90  218E             MOVB    #^A/N/,BUFFER+4          ; ...else disable the DDB record...
                              2194                 $UPDATE RAB = (R10)              ; ...here
                     3C 50    E9  219D             BLBC    R0,UPDATE_FAILED         ; If error then forget it
                              21A0  20$:
                      5B   6B C0  21A0             ADDL2   (R11),R11                ; Next unit block
           000001D8'8F     5B D1  21A3             CMPL    R11,#UNIT_LIST           ; Are we full circle in the list?
                           4E 13  21AA             BEQL    END_UPDATE               ; BR if yes
                              21AC                 $GET    RAB = (R10)              ; Get a record
                     24 50    E9  21B5             BLBC    R0,UPDATE_FAILED         ; If error then forget it
              0014'CF      20 8A  21B8             BICB2   #LC_BITM,BUFFER          ; Convert to uppercase
              0014'CF   55 8F 91  21BD             CMPB    #^A/U/,BUFFER            ; Is it a UCB record?
                           35 12  21C3             BNEQ    END_UPDATE               ; BR if not
                           01 E0  21C5             BBS     #UETUNT$V_TESTABLE,-     ; BR if this unit is testable...
                     D6 0B AB                              UETUNT$B_FLAGS(R11),20$
              0018'CF   4E 8F 90  21CA             MOVB    #^A/N/,BUFFER+4          ; ...else disable the UCB record...
                              21D0                 $UPDATE RAB = (R10)              ; ...here
                     C4 50    E8  21D9             BLBS    R0,20$                   ; Look at the next record if no error
                              21DC  UPDATE_FAILED:
                     0C AA    DD  21DC             PUSHL   RAB$L_STV(R10)           ; Do a simple message...
                           50 DD  21DF             PUSHL   R0                       ; ...to tell of the failure
                      0197'CF DF  21E1             PUSHAL  INIDEV_UPDERR
                           01 DD  21E5             PUSHL   #1
                           00 EF  21E7             EXTZV   #STS$V_SEVERITY,-        ; Copy the severity from RMS status...
                  7E  50   03     21E9                     #STS$S_SEVERITY,R0,-(SP)
           6E  00741130 8F    C8  21EC             BISL2   #UETP$_TEXT,(SP)         ; ...to our message
              00000000'GF  05 FB  21F3             CALLS   #5,G^LIB$SIGNAL
                              21FA  END_UPDATE:
                 FB55 CF   00 FB  21FA             CALLS   #0,CLEANUP               ; Get rid of test specific leftovers
                           00 DD  21FF             PUSHL   #0                       ; Set the time flag
                      000F'CF DF  2201             PUSHAL  TEST_NAME                ; Push the test name
                           02 DD  2205             PUSHL   #2                       ; Push arg count
                           00 EF  2207             EXTZV   #STS$V_SEVERITY,-        ; Push the proper exit severity...
                           03     2209                     #STS$S_SEVERITY,-
                  7E   018A'CF    220A                     STATUS,-(SP)
           6E  00741080 8F    C8  220E             BISL2   #UETP$_ENDEDD,(SP)       ; ...and use it in our message code
                           04 DD  2215             PUSHL   #4
                     51    5E D0  2217             MOVL    SP,R1
                              221A                 $PUTMSG_S MSGVEC = (R1)          ; Output the message
                              2229                 $SETPRN_S PRCNAM = ACNT_NAME     ; Reset the process name
                           04     2234             RET                              ; That's all folks!
                              2235
                              2235                 .END    UETMA7800
```

```
$$.TAB                        = 00001464 R       06
$$.TABEND                     = 000014A8 R       06
$$.TMP                        = 00000800
$$.TMP1                       = 00000001
$$.TMP2                       = 0000006A
$$.TMPX                       = 00000016 R       07
$$.TMPX1                      = 0000000D
$$T1                          = 00000000
$$T2                          = 00000006
ACNT_NAME                       00000000 R       05
ADAWI_LIMIT                   = 00030D40
ADP$L_CSR                     = 00000000
ADP$W_TR                      = 0000000C
ALL_SET                         0000060C R       08
ALREADY_HERE                    000002F3 R       05
ARG_COUNT                       000001CE R       06
AST_MBBUF                       0000110A R       06
AST_MBCHN                       00001129 R       06
AST_MBDAY                       0000115C R       06
AST_MBISB                       0000112D R       06
AST_MBNAM                       0000113D R       06
AST_MBPTR                       00001135 R       06
BAD_CEF                         00000567 R       05
BAD_ECO                         0000071D R       05
BAD_MB_QIO                      0000064C R       05
BAD_MOVC3                       00000402 R       05
BAD_MSG_TYPE                    000005E2 R       05
BAD_REQIDT                      000006D2 R       05
BAD_UPDATE                      00000457 R       05
BEGIN_MSGM                    = 00000008
BEGIN_MSGV                    = 00000003
BUFFER                          00000014 R       06
BUFFER_PTR                      0000000C R       06
CCASTHAND                       00002069 R       08
CHF$L_SIGARGLST               = 00000004
CHF$L_SIG_ARG1                = 00000008
CHF$L_SIG_ARGS                = 00000000
CHF$L_SIG_NAME                = 00000004
CHKGSFAB                        00001414 R       06
CHKGSRAB                        00001464 R       06
CLEANUP                         00001D54 R       08
CNTRLCMSG                       00000082 R       05
COMEF_EFN                     = 00000060
COMGSFAB                        000012EC R       06
COMGSRAB                        0000133C R       06
COMGS_BUF                       00000208 R       06
COMGS_EFN                     = 0000003E
COMGS_NAME_LEN                = 00000010
COMGS_PVT_AREA                = 00000200
COMGS_PVT_CODE                = 000001F8
COMGS_QUEUE_MAX               = 00000064
COMGS_SIZE                    = 00000008
COMGS_VERSION                 = 00003039
COMMON                          00002030 R       08
COMMON_WAKE_TMO                 00001C29 R       08
CONSIS_CHECK                  = 30390004
CONTROLLER                      00000031 R       05

CONT_DESC                       000001CC R       05
CS2                             00000273 R       05
DDB_RFA                         00001294 R       06
DEAD_CTRLNAME                   000000C3 R       05
DEV$V_TRM                     = 00000002
DEVDEP_SIZE                   = 000004A8
DEVDSC                          000000DC R       06
DEVNAM_LEN                      000001A8 R       06
DEV_NAME                        000000FB R       06
DIB                             0000010A R       06
DIB$K_LENGTH                  = 00000074
DIBBUF                          00000112 R       06
DOCLEAR_MSG                     000009C3 R       05
DOCLEAR_TIMEOUT               = FA0A1F00
DOSET_MSG                       00000978 R       05
DOSET_TIMEOUT                 = FA0A1F00
DOTDAT                          000002DF R       05
DO_FAO                          00001269 R       08
DO_FAO_EXIT                     000012A6 R       08
DO_FAO_INFO                     0000128A R       08
DO_FAO_SIGNAL                   00001286 R       08
DVI$_DEVNAM                   = 00000020
EFN2                          = 00000004
END_UPDATE                      000021FA R       08
ERROR_COUNT                     00000186 R       06
ERROR_EXIT                      000020A0 R       08
ERROR_SUMMARY                   00001C47 R       08
EXE$GL_SHBLIST                  ********   X     08
EXIT_DESC                       000001BE R       06
EXIT_FROM_RCV                   00001518 R       08
EXIT_FROM_TMO                   00001B5D R       08
EXIT_HANDLER                    00002125 R       08
FAB$B_BID                     = 00000000
FAB$B_FNS                     = 00000034
FAB$C_BID                     = 00000003
FAB$C_BLN                     = 00000050
FAB$C_FIX                     = 00000001
FAB$C_SEQ                     = 00000000
FAB$C_VAR                     = 00000002
FAB$K_BLN                     = 00000050
FAB$L_ALQ                     = 00000010
FAB$L_DEV                     = 00000040
FAB$L_FNA                     = 0000002C
FAB$L_FOP                     = 00000004
FAB$L_STS                     = 00000008
FAB$L_STV                     = 0000000C
FAB$M_UFO                     = 00020000
FAB$V_BIO                     = 00000005
FAB$V_CHAN_MODE               = 00000002
FAB$V_CR                      = 00000001
FAB$V_FILE_MODE               = 00000004
FAB$V_GET                     = 00000001
FAB$V_LNM_MODE                = 00000000
FAB$V_PUT                     = 00000000
FAB$V_SUP                     = 00000002
FAB$V_UFO                     = 00000011
FAB$V_UPD                     = 00000003
```

```
FAB$V_UPI                = 00000006              MAX_MSG_TYPE             = 0000000C
FAB$W_GBC                = 00000048              MAX_PROC_NAME            = 0000000F
FAB$W_IFI                = 00000002              MAX_UNIT_DESIG           = 00000005
FAO_BUF                    00000004 R     06     MA_B_MYPORT                000001AC
FILE                       000001D4 R     05     MA_K_CHKBUF                0000264C
FIND_IT                    00000215 R     08     MA_K_CHKFAB                00000590
FLAG                       00000002 R     06     MA_K_CHKRAB                000005E0
FOUND_IT                   000002FC R     08     MA_K_COMFAB              = 00000110
GETJPI_ITMLST              00000258 R     05     MA_K_COMRAB              = 00000160
GETSYI_ITMLST              00000230 R     05     MA_K_ERRLEN              = 0000002C
GS_K_ADAWI_PVT             0000003C              MA_K_SPRBUF                0000064C
GS_K_END                   00000EA8              MA_K_SPRFAB                000003FC
GS_K_PVT_AREAS             000006A8              MA_K_SPRRAB                0000044C
GS_K_Q_COUNT               0000004C              MA_L_DOCLEAR               000001E8
GS_K_Q_ENTRIES             00000068              MA_L_DOCLEARFAIL           000001C4
GS_L_ADAWI_COM             00000038              MA_L_DOSET                 000001E4
GS_L_AVAILABLE             00000024              MA_L_DOSETFAIL             000001C0
GS_L_CEF_FLAG              00000028              MA_L_ERRBLK                000001B0
GS_L_ECOLEVEL              0000001C              MA_L_HELLO                 000001DC
GS_L_FINISHED              00000030              MA_L_IADAWI                000001D4
GS_L_INTLK_FLAG            00000034              MA_L_ICU_UCME              000001E0
GS_L_OWNER                 00000020              MA_L_INTLKTMO              000001D0
GS_L_PARTICIPATING         0000002C              MA_L_IQUEUE                000001D8
GS_L_VERSION               00000010              MA_L_NOCAUSE               000001C8
GS_Q_BUSY                  00000060              MA_L_OLDSTUFF              000001CC
GS_Q_FREE                  00000058              MA_L_RBADCODE              000001B8
GS_Q_SYSGQVERSION          00000014              MA_L_RILOSTU               000001B4
GS_T_SECT_NAME             00000000              MA_L_SBADCODE              000001BC
HELLO_TIMEOUT            = FA0A1F00              MA_L_SILOSTU               000001B0
HEXUNIT                    00001231 R     08     MA_L_SPREFSTATE            000002CD
IADAWI_MSG                 00000B55 R     05     MA_Q_COMEFNAM              0000022F
ICU_UCME_TIMEOUT         = FA0A1F00              MA_Q_COMGSIAD              00000624
ILLEGAL_REC                00000130 R     05     MA_Q_COMGSNAM              00000208
INADDRESS                  00000196 R     06     MA_Q_COMGSRAD              0000062C
INCONSISTENCY              0000036C R     05     MA_Q_COMMBNAM              00000258
INIDEV_UPDERR              00000197 R     05     MA_Q_DAYTIME               000001A4
INI_FAB                    00001200 R     06     MA_Q_MEMNAM                000001F1
INI_RAB                    00001250 R     06     MA_Q_SPREFNAM              000002A6
INPUT_ITMLST               00000072 R     05     MA_Q_SPRGSIAD              00000634
INTERLOCKING_TEST          000010F7 R     08     MA_Q_SPRGSISB              00000644
INTERLOCK_LIMIT          = 0000000A              MA_Q_SPRGSNAM              0000027F
INTERLOCK_SAVED_R6         00001164 R     06     MA_Q_SPRGSRAD              0000063C
INTERPROC_EF               00000CA4 R     08     MA_Q_SPRMBNAM              000002D5
INTERPROC_GS               00000B94 R     08     MA_T_CHKFAB                00000490
INTLKTMO_MSG               00000AF1 R     05     MA_T_COMEFNAM              00000237
IO$M_CTRLCAST              ********   X   08     MA_T_COMFAB              = 00000014
IO$M_NOW                   ********   X   08     MA_T_COMGSNAM              00000210
IO$_READVBLK               ********   X   08     MA_T_COMMBNAM              00000260
IO$_SETMODE                ********   X   08     MA_T_MEMNAM                000001F9
IO$_WRITEVBLK              ********   X   08     MA_T_SPREFNAM              000002AE
IQUEUE_MSG                 00000BBE R     05     MA_T_SPRFAB                000002FC
ITERATION                  000001B2 R     06     MA_T_SPRGSNAM              00000287
JPI$_TQCNT               = 00000315              MA_T_SPRMBNAM              000002DD
LC_BITM                  = 00000020              MA_T_UNIT                  000001F0
LIB$SIGNAL                 ********   X   08     MA_W_COMMBCHN              00000256
MASK_2_ARGLST              0000124D R     08     MA_W_SPRMBCHN              000002D1
MAX_DEV_DESIG            = 0000000A              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     SYI$_VERSION            = 00001000
RMS_ERR_STRING               000001EE R     05     SYNCH_MBBUF               000010B0 R     06
SAFE_TO_UPDM               = 00000004              SYNCH_MBCHN               000010CF R     06
SAFE_TO_UPDV               = 00000002              SYNCH_MBDAY               00001102 R     06
SBADCODE_MSG                 00000905 R     05     SYNCH_MBISB               000010D3 R     06
SCSNODE                      000001F4 R     06     SYNCH_MBNAM               000010E3 R     06
SCSNODE_LENGTH               000001FA R     06     SYNCH_MBPTR               000010DB R     06
SEC$M_EXPREG                 ********   X   08     SYS$ASCEFC                ********  GX   08
SEC$M_GBL                    ********   X   08     SYS$ASSIGN                ********  GX   08
SEC$M_PERM                   ********   X   08     SYS$CANTIM                ********  GX   08
SEC$M_WRT                    ********   X   08     SYS$CLOSE                 ********  GX   08
SHB$B_NEXUS                = 00000014              SYS$CLRAST                ********   X   08
SHB$B_PORT                 = 00000015              SYS$CLREF                 ********  GX   08
SHB$L_ADP                  = 0000001C              SYS$CMKRNL                ********  GX   08
SHB$L_DATAPAGE             = 00000004              SYS$CONNECT               ********  GX   08
SHB$L_LINK                 = 00000000              SYS$CREATE                ********  GX   08
SHD$T_NAME                 = 00000020              SYS$CREMBX                ********  GX   08
SHM_INFO                     000012B9 R     08     SYS$CRMPSC                ********  GX   08
SHM_INFO_ARGS                000001E0 R     06     SYS$DACEFC                ********  GX   08
SHR$_ABENDD                = 000010E0              SYS$DASSGN                ********  GX   08
SHR$_BEGIND                = 00001038              SYS$DCLEXH                ********  GX   08
SHR$_ENDEDD                = 00001080              SYS$DELTVA                ********  GX   08
SHR$_OPENIN                = 00001098              SYS$DGBLSC                ********  GX   08
SHR$_TEXT                  = 00001130              SYS$ERASE                 ********  GX   08
SILOSTU_MSG                  00000836 R     05     SYS$EXIT                  ********  GX   08
SPREF_EFN                  = 00000040              SYS$EXPREG                ********  GX   08
SPRGSFAB                     00001380 R     06     SYS$FAO                   ********   X   08
SPRGSRAB                     000013D0 R     06     SYS$FAOL                  ********  GX   08
SPRGS_AST                    0000130F R     08     SYS$GET                   ********  GX   08
SPRGS_EFN                  = 0000003F              SYS$GETDVI                ********  GX   08
SPRGS_SIZE                 = 00000010              SYS$GETJPI                ********  GX   08
SS$_BADPARAM               = 00000014              SYS$GETMSG                ********  GX   08
SS$_CONTROLC               = 00000651              SYS$GETSYIW               ********  GX   08
SS$_CREATED                = 00000619              SYS$GETTIM                ********  GX   08
SS$_NORMAL                 = 00000001              SYS$HIBER                 ********  GX   08
SS$_NOSUCHSEC              = 00000978              SYS$INPUT                 00000061 R     05
SS$_SSFAIL                 = 0000045C              SYS$MGBLSC                ********  GX   08
SS$_WASSET                 = 00000009              SYS$OPEN                  ********  GX   08
SSERROR                      00001F17 R     08     SYS$PUTMSG                ********  GX   08
SS_SYNCH_EFN               = 00000003              SYS$QIO                   ********  GX   08
STATUS                       0000018A R     06     SYS$QIOW                  ********  GX   08
STR$UPCASE                   ********   X   08     SYS$READ                  ********  GX   08
STS$K_ERROR                = 00000002              SYS$READEF                ********  GX   08
STS$K_INFO                 = 00000003              SYS$SCHDWK                ********  GX   08
STS$K_SUCCESS              = 00000001              SYS$SETAST                ********  GX   08
STS$K_WARNING              = 00000000              SYS$SETEF                 ********  GX   08
STS$M_INHIB_MSG            = 10000000              SYS$SETIMR                ********  GX   08
STS$S_FAC_NO               = 0000000C              SYS$SETPRN                ********  GX   08
STS$S_SEVERITY             = 00000003              SYS$SETPRV                ********  GX   08
STS$V_FAC_NO               = 00000010              SYS$SETSFM                ********  GX   08
STS$V_SEVERITY             = 00000000              SYS$TEST                  000002CE R     05
SUC_EXIT                     00001190 R     08     SYS$TRNLOG                ********  GX   08
SUMMARY_HEADER               000007A0 R     05     SYS$UPDATE                ********  GX   08
SUPDEV_GBLSEC                00000020 R     05     SYS$UPDSEC                ********  GX   08
SUP_FAB                      0000129C R     06     SYS$WAITFR                ********  GX   08
SYI$_SCSNODE               = 00001067              SYS$WAKE                  ********  GX   08
SYI$_SID                   = 00001001              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

0427 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

UETINIT00
LIS

UETINIT01
LIS

UETLOCK00
LIS

UETMA7800
LIS

0428 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DISKMON
MAP

UETMEMY01
LIS

UETSSMM00
LIS

DCLDIET
MAP

DISKQ
MAP

UTIL32

CHECKSUM
MAP