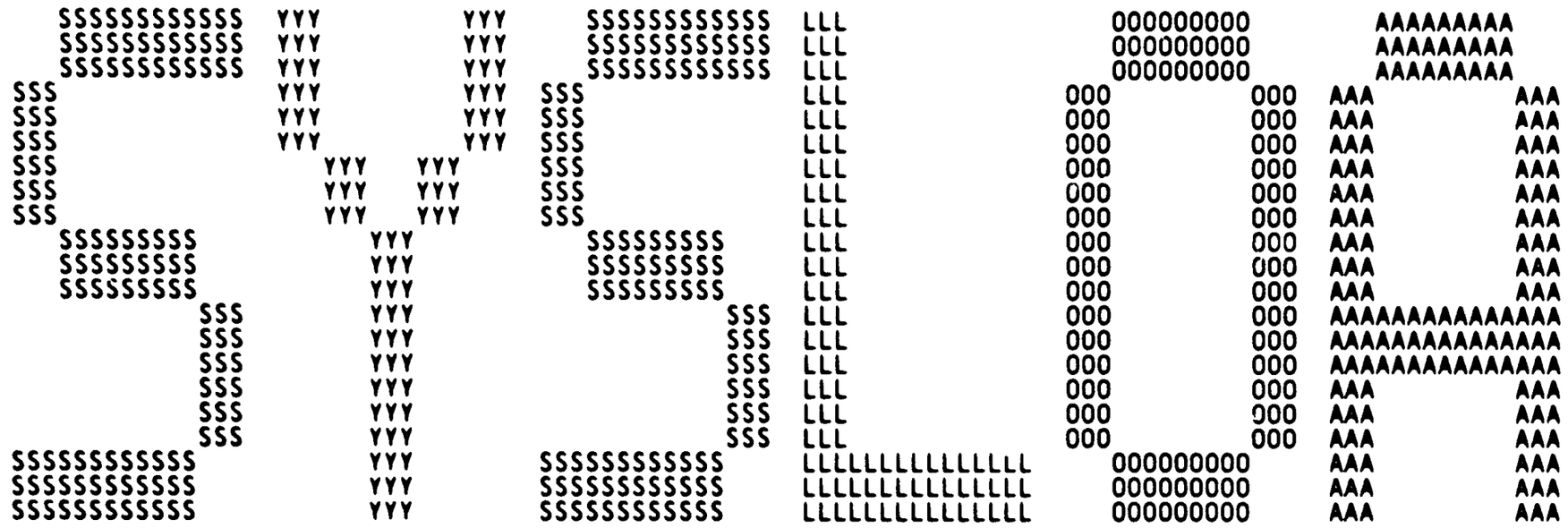
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA

```
MM      MM    CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK  77777777     888888     000000
MM      MM    CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK  77777777     888888     000000
MMMM  MMMM  CC          HH      HH  EE          CC          KK      KK          77  88      88  00      00
MMMM  MMMM  CC          HH      HH  EE          CC          KK      KK          77  88      88  00      00
MM  MM  MM  CC          HH      HH  EE          CC          KK    KK            77  88      88  00    0000
MM  MM  MM  CC          HH      HH  EE          CC          KK    KK            77  88      88  00    0000
MM      MM  CC          HHHHHHHHHH  EEEEEEEE    CC          KKKKKK            77    888888    00  00  00
MM      MM  CC          HHHHHHHHHH  EEEEEEEE    CC          KKKKKK            77    888888    00  00  00
MM      MM  CC          HH      HH  EE          CC          KK    KK        77      88      88  0000    00
MM      MM  CC          HH      HH  EE          CC          KK    KK        77      88      88  0000    00
MM      MM  CC          HH      HH  EE          CC          KK      KK    77        88      88  00      00
MM      MM  CC          HH      HH  EE          CC          KK      KK    77        88      88  00      00
MM      MM    CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK  77          888888     000000
MM      MM    CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK  77          888888     000000

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

Table of contents

```
0000     1          .NLIST CND
0000     3          .TITLE  MCHECK780 - MACHINE CHECK EXCEPTION HANDLER FOR 11/780
0000     7
0000     8          .IDENT  'V04-000'
0000     9 ;
0000    10 ;*****************************************************************************
0000    11 ;*                                                                           *
0000    12 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    13 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    14 ;*  ALL RIGHTS RESERVED.                                                     *
0000    15 ;*                                                                           *
0000    16 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    17 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    18 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    19 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    20 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    21 ;*  TRANSFERRED.                                                             *
0000    22 ;*                                                                           *
0000    23 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    24 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    25 ;*  CORPORATION.                                                             *
0000    26 ;*                                                                           *
0000    27 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    28 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    29 ;*                                                                           *
0000    30 ;*                                                                           *
0000    31 ;*****************************************************************************
0000    32
0000    33 ;++
0000    34 ; FACILITY:     EXECUTIVE, ERROR HANDLING
0000    35 ;
0000    36 ; ABSTRACT: IN A NUTSHELL, LOG IT AND TRY TO RECOVER.
0000    37 ;
0000    38 ; ENVIRONMENT:  RUNS ON INTERRUPT STACK AT IPL 31 UNTIL ERROR TYPE IS KNOWN
0000    39 ;               AND (IF POSSIBLE) CORRECTED, THEN RUNS AT SYNCH LEVEL
0000    40 ;               TO DO THE ERROR LOGGING.
0000    57
0000    58 ;--
0000    59          .SBTTL  HISTORY                 ; DETAILED
0000    60 ;
0000    61 ; AUTHOR: RICHARD LARY  , CREATION DATE: 6-NOV-77
0000    62 ;
0000    63 ; MODIFIED BY:
0000    64 ;
0000    65 ;       V03-013 WMC0002         Wayne Cardoza           25-Jul-1984
0000    66 ;               Add H memory to the tables.
0000    67 ;
0000    68 ;       V03-012 WMC0001         Wayne Cardoza           14-Jun-1984
0000    69 ;               Preserve cache state when handling machine check.
0000    70 ;               Properly clear group 1 cache parity errors.
0000    71 ;
0000    72 ;       V03-011 NPK3049         N. Kronenberg           10-Apr-1984
0000    73 ;               Tighten up check for BRRVR reference from unibus
0000    74 ;               interrupt service routine in CPTIMOUT.  Test for
0000    75 ;               PC as well as VA.
0000    76 ;
0000    77 ;       V03-010 RLRSBICONF      Robert L. Rappaport     22-Mar-1984
```

```
0000    78 ;               Test MMG$GL_SBICONF array elements for valid system
0000    79 ;               virtual address (high bit set) before using.  Also
0000    80 ;               correct error introduced by CONFREGL change.
0000    81 ;
0000    82 ;       V03-009 KPL0100         Peter Lieberwirth       10-Feb-1984
0000    83 ;               Change to use CONFREGL.
0000    84 ;
0000    85 ;       V03-008 KDM0053         Kathleen D. Morse       11-Jul-1983
0000    86 ;               Replace cpu-specific IPR references with the new
0000    87 ;               cpu-specific $PR780DEF symbols.
0000    88 ;
0000    89 ;       V03-007 TCM0011         Trudy C. Matthews       24-Jan-1983
0000    90 ;               Correct bug in MA780 logging routine that checked for
0000    91 ;               Multiple Interlock Accepted error bit in the wrong
0000    92 ;               MA780 register.
0000    93 ;
0000    94 ;       V03-006 KDM0040         Kathleen D. Morse       13-Jan-1983
0000    95 ;               Change PRMSW to MPSWITCH and integrate into multi-processing
0000    96 ;               code replacing [MP.SRC]MPMCHECK.MAR.  Fix bug that referenced
0000    97 ;               devices attached to primary (via CONFREG array) from the
0000    98 ;               secondary processor's machine-check code.
0000    99 ;
0000   100 ;       V03-005 RNG0001         Rod N. Gamache          15-Oct-1982
0000   101 ;               Fixed code that enabled the MS780-E memory CRD (corrected
0000   102 ;               read data) interrupts. Fixed code that re-enabled the MS780-C
0000   103 ;               CRD interrupts.
0000   104 ;
0000   105 ;
```

```
0000    107          .SBTTL  MEMORY_ROUTINES Macro
0000    108 ;++
0000    109 ; Macro MEMORY_ROUTINES
0000    110 ;       Build action routine vectors for different memory types.
0000    111 ;
0000    112 ; Inputs:
0000    113 ;       MEMTYPES        - A list of 'NDT$_' type codes for this controller.
0000    114 ;       LOGERR_RTN      - Action routine that determines if an error was
0000    115 ;                         reported for this controller; if so, it logs it.
0000    116 ;       LOGALL_RTN      - Action routine to unconditionally log this
0000    117 ;                         controller's registers.
0000    118 ;       ENAB_RTN        - Action routine to enable CRD interrupts for this
0000    119 ;                         memory controller.
0000    120 ;
0000    121 ; Outputs:
0000    122 ;       Additions to LOGERR_ROUTINES, LOGALL_ROUTINES, and ENAB_ROUTINES arrays.
0000    123 ;
0000    124 ; Note: Each invocation of this macro corresponds to one "general" memory type.
0000    125 ;       Each element in MEMTYPES list corresponds to one "specific" type.
0000    126 ;--
0000    127          .MACRO  MEMORY_ROUTINES         MEMTYPES,LOGERR_RTN,LOGALL_RTN,ENAB_RTN
0000    128          .SAVE
0000    129 ;
0000    130 ; Create arrays to map a set of specific type codes to one general memory type.
0000    131 ; Note: Psects MCHK$DATA0 and MCHK$DATA1 must be contiguous.
0000    132 ;
0000    133          .IRP    MEMTYP,MEMTYPES         ; Repeat for each memory type...
0000    134
0000    135          .IF     NDF,MPSWITCH            ;***** ONLY PRIMARY PROCESSOR...
0000    136          .PSECT  MCHK$DATA0,LONG,WRT     ; Add specific-type entry to MEMTYP
0000    137          .IFF                            ;***** ONLY SECONDARY PROCESSOR...
0000    138          .PSECT  Y$MPDATA0,LONG,WRT      ; Add specific-type entry to MEMTYP
0000    139          .ENDC                           ;***** PRIMARY and SECONDARY PROCESSORS
0000    140          .BYTE   MEMTYP                  ; array.
0000    141
0000    142          .IF     NDF,MPSWITCH            ;***** ONLY PRIMARY PROCESSOR...
0000    143          .PSECT  MCHK$DATA1              ; Add general-type entry to MEMTYP
0000    144          .IFF                            ;***** ONLY SECONDARY PROCESSOR...
0000    145          .PSECT  Y$MPDATA1               ; Add general-type entry to MEMTYP
0000    146          .ENDC                           ;***** PRIMARY and SECONDARY PROCESSORS
0000    147          .BYTE   GENERAL_MEMTYP
0000    148
0000    149 MEMTYPCNT = MEMTYPCNT + 1
0000    150          .ENDR
0000    151 GENERAL_MEMTYP = GENERAL_MEMTYP + 1
0000    152 ;
0000    153 ; Now create action routine vectors.
0000    154 ;
0000    155          .IF     NDF,MPSWITCH            ;***** ONLY PRIMARY PROCESSOR...
0000    156          .PSECT  MCHK$DATA2,LONG,WRT     ; LOGERR_ROUTINES array:
0000    157          .IFF                            ;***** ONLY SECONDARY PROCESSOR...
0000    158          .PSECT  Y$MPDATA2,LONG,WRT      ; LOGERR_ROUTINES array:
0000    159          .ENDC                           ;***** PRIMARY and SECONDARY PROCESSORS
0000    160          .LONG   <LOGERR_RTN-.>          ; Add self-relative offset to routine.
0000    161
0000    162          .IF     NDF,MPSWITCH            ;***** ONLY PRIMARY PROCESSOR...
0000    163          .PSECT  MCHK$DATA3,LONG,WRT     ; LOGALL_ROUTINES array:
```

```
0000   164          .IFF                                ;***** ONLY SECONDARY PROCESSOR...
0000   165          .PSECT  Y$MPDATA3,LONG,WRT          ; LOGALL_ROUTINES array:
0000   166          .ENDC                               ;***** PRIMARY and SECONDARY PROCESSORS
0000   167          .LONG   <LOGALL_RTN-.>              ; Add self-relative offset to routine.
0000   168
0000   169          .IF     NDF,MPSWITCH                ;***** ONLY PRIMARY PROCESSOR...
0000   170          .PSECT  MCHK$DATA4,LONG,WRT         ; ENAB_ROUTINES array:
0000   171          .IFF                                ;***** ONLY SECONDARY PROCESSOR...
0000   172          .PSECT  Y$MPDATA4,LONG,WRT          ; ENAB_ROUTINES array:
0000   173          .ENDC                               ;***** PRIMARY and SECONDARY PROCESSORS
0000   174          .LONG   <ENAB_RTN-.>                ; Add self-relative offset to routine.
0000   175
0000   176          .RESTORE
0000   177          .ENDM   MEMORY_ROUTINES
```

```
                        0000  179          .SBTTL  SYMBOL DEFINITIONS
                        0000  180
                        0000  181
0000000A                0000  182 CH_THRESHOLD     =       10.             ;3 ERRORS IN 100 MS TO DISABLE CACHE
00010000                0000  183 CH_MISSG0         =       ^X10000         ;"FORCE MISS GROUP 0" BIT
00008000                0000  184 CH_MISSG1         =       ^X8000          ;"FORCE MISS GROUP 1" BIT
00004000                0000  185 CH_REPLG0         =       ^X4000          ;"FORCE REPLACE GROUP 0" BIT
00002000                0000  186 CH_REPLG1         =       ^X2000          ;"FORCE REPLACE GROUP 1" BIT
0000000D                0000  187 CH$V_REPLG1       =       13
00000004                0000  188 CH$S_CONTROL      =       4               ;SIZE OF CACHE CONTROL FIELD
0021C000                0000  189 CH_REPAIR         =       ^X21C000        ;BITS TO SET IN SBIMT ON CACHE ERRORS
0021A000                0000  190 CH_REPAIR_1       =       ^X21A000        ;BITS CLEAR GROUP 1 CACHE ERRORS
00000003                0000  191 CH$V_GOERRS       =       3               ;START OF GROUP 0 ERRORS IN PARITY REG
00000007                0000  192 CH$S_GOERRS       =       7               ;LENGTH OF GROUP 0 ERROR BITS
00000001                0000  193 CHLOG_DISAB0      =       1               ;LOG BIT SAYING WE DISABLED GROUP 0
00000002                0000  194 CHLOG_DISAB1      =       2               ;LOG BIT SAYING WE DISABLED GROUP 1
                        0000  195
                        0000  196
00000019                0000  197 SBIFS$V_NEF      =       25              ;NESTED ERROR FLAG IN SBI FAULT/STATUS
                        0000  198
                        0000  199 ;THE FOLLOWING 5 DEFINITIONS ARE IN THE SBI ERROR REGISTER
00000040                0000  200 SBIER$M_IBTO      =       ^X40            ;IB TIMEOUT LATCH
00000080                0000  201 SBIER$M_IBRDS     =       ^X80            ;IB RDS LATCH
00001000                0000  202 SBIER$M_CPTO      =       ^X1000          ;CP TIMEOUT LATCH
00002000                0000  203 SBIER$M_RDS       =       ^X2000          ;RDS LATCH
00004000                0000  204 SBIER$M_CRD       =       ^X4000          ;CRD LATCH
                        0000  205
                        0000  206
                        0000  207 ;       MACHINE CHECK HARDWARE LOG OFFSETS
                        0000  208
00000000                0000  209 MCL_COUNT         =       0               ;BYTE LENGTH OF AREA (28 HEX)
00000004                0000  210 MCL_SUMMARY       =       4               ;SUMMARY WORD - BYTE 0=CODE, BYTE 1=
                        0000  211                                             ;TIMEOUT PENDING FLAG
00000008                0000  212 MCL_CES           =       8               ;CPU ERROR STATUS
0000000C                0000  213 MCL_UPC           =       12.             ;MICRO-PC AT FAULT TIME
00000010                0000  214 MCL_VA            =       16.             ;VIRTUAL ADDR AT FAULT TIME
00000014                0000  215 MCL_D             =       20.             ;CPU D REGISTER AT FAULT TIME
00000018                0000  216 MCL_TBER0         =       24.             ;TRANSLATION BUFFER STATUS REG 0
0000001C                0000  217 MCL_TBER1         =       28.             ;TBUF STATUS REG 1
00000020                0000  218 MCL_TIMOADDR      =       32.             ;PHYSICAL ADDRESS CAUSING SBI TIMEOUT
00000024                0000  219 MCL_PARITY        =       36.             ;CACHE STATUS REGISTER
00000028                0000  220 MCL_SBIERR        =       40.             ;SBI ERROR REGISTER
0000002C                0000  221 MCL_PC            =       44.             ;PC OF INSTRUCTION WHICH CAUSED CHECK
00000030                0000  222 MCL_PSL           =       48.             ;PSL OF MACHINE AT FAULT TIME
```

```
                     0000   224          .SBTTL MEMORY CONTROLLOR AND ERROR DEFINITIONS
                     0000   225
                     0000   226 ;
                     0000   227 ; Common error bit definitions.
                     0000   228 ;
0000001C             0000   229 MRC$V_ELSRF     =       28              ;ERROR LOG SERVICE REQUEST
10000000             0000   230 MRC$M_ELSRF     =       ^X10000000      ;WRITE 1 TO CLEAR FLAG
0000001D             0000   231 MRC$V_HERIMF    =       29              ;HIGH E   . RATE IN MEMORY
20000000             0000   232 MRC$M_HERIMF    =       ^X20000000      ;WRITE 1 TO CLEAR FLAG
0000001E             0000   233 MRC$V_INHBCRD   =       30              ;1 DISABLES CRD INTERRUPT
40000000             0000   234 MRC$M_INHBCRD   =       ^X40000000      ;0 CRD INTERRUPT ENABLE, 1 CRD DISABLE
                     0000   235 ;
                     0000   236 ; MA780-specific error bit definitions (in Array Error Register).
                     0000   237 ;
0000001F             0000   238 MRC$V_INVMAPPTY =       31              ;INVALID MAP PARITY ERROR
80000000             0000   239 MRC$M_INVMAPPTY =       ^X80000000      ;WRITE 1 TO CLEAR THE FLAG
                     0000   240 ;
                     0000   241 ; MS780E-specific error bit definitions.
                     0000   242 ;
00000014             0000   243 MRC$V_SUMMARY   =       20              ;ERROR SUMMARY BIT
00100000             0000   244 MRC$M_SUMMARY   =       ^X00100000      ;OR OF ALL ERROR BITS -- READ ONLY
00000013             0000   245 MRC$V_CTL1PTY   =       19              ;PARITY ERROR ON READ DATA FROM
00080000             0000   246 MRC$M_CTL1PTY   =       ^X00080000      ;CONTROLLER 1 TO SBI INTERFACE.
00000012             0000   247 MRC$V_CTL0PTY   =       18              ;PARITY ERROR ON READ DATA FROM
00040000             0000   248 MRC$M_CTL0PTY   =       ^X00040000      ;CONTROLLER 0 TO SBI INTERFACE.
                     0000   249                                         ;FOLLOWING BITS ARE IN REGISTERS C & D
00000007             0000   250 MRC$V_MSEQPTY   =       7               ;MICROSEQUENCER PARITY ERROR
00000080             0000   251 MRC$M_MSEQPTY   =       ^X00000080      ;
00000008             0000   252 MRC$V_IFPTY     =       8               ;PARITY ERROR ON WRITE DATA FROM
00000100             0000   253 MRC$M_IFPTY     =       ^X00000100      ;SBI INTERFACE TO CONTROLLER.
00000009             0000   254 MRC$V_CRDERR    =       9               ;CORRECTED READ DATA ERROR
00000200             0000   255 MRC$M_CRDERR    =       ^X00000200      ;
                     0000   256
00000384             0000   257 REENABTIME      = 60*15                 ;REENABLE INTERRUPT ERROR LOGGING
                     0000   258                                         ;EVERY 15 MINUTES
0000003C             0000   259 SOMETIME        = 60                    ;SCAN FOR NON-INTERRUPT ERRORS
                     0000   260                                         ;EVERY 60 SECONDS
00000003             0000   261 CRDINTMAX       = 3                     ;MAXIMUM NUMBER OF INTERRUPTS A CONT
                     0000   262                                         ;IS ALLOWED WITHIN REENABTIME
00000006             0000   263 CRDWATCHMAX     = 6                     ;MAXIMUM NUMBER OF ERRORS TO BE LOGGED
                     0000   264                                         ;WITHIN REENABTIME
                     0000   265
                     0000   266 ;       INCLUDED SYMBOL DEFINITIONS
                     0000   267
                     0000   268         $ADPDEF                         ;DEFINE ADAPTER CONTROL BLOCK SYMBOLS
                     0000   269         $EMBDEF <MC,SB,SE>              ;DEFINE EMB OFFSETS
                     0000   270         $IPLDEF                         ;PROCESSOR INTERRUPT LEVELS
                     0000   271         $MCHKDEF                        ;DEFINE RECOVERY BLOCK MASK BITS
                     0000   272         $NDTDEF                         ;DEFINE NEXUS DEVICE TYPES
                     0000   273         $PCBDEF                         ;PROCESS CTL BLOCK
                     0000   274         $PFNDEF                         ;PFN DATA BASE
                     0000   275         $PRDEF                          ;DEFINE PROCESSOR REGISTER NUMBERS
                     0000   276         $PR780DEF                       ;DEFINE 780-SPECIFIC PROCESSOR REGISTERS
                     0000   277         $PSLDEF                         ;DEFINE PSL
                     0000   278         $PTEDEF                         ;PTE SYMBOLS
                     0000   279         $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
                     0000   280         $VADEF                          ;DEF IN PFN PITS
```

```
                     0000   282          .SBTTL  MEMORY ACTION ROUTINE ARRAYS
                     0000   283
                00000000    285          .PSECT  MCHK$DATA0,LONG,WRT
                     0000   289 MEMTYP:                                ; Define base of array of memory type
                     0000   290                                        ; codes.
                00000000    292          .PSECT  MCHK$DATA2,LONG,WRT
                     0000   296 LOGERR_ROUTINES:                       ; Define base of array of routines to
                     0000   297                                        ; log memories with errors.
                00000000    299          .PSECT  MCHK$DATA3,LONG,WRT
                     0000   303 LOGALL_ROUTINES:                       ; Define base of array of routines to
                     0000   304                                        ; unconditionally log memories.
                00000000    306          .PSECT  MCHK$DATA4,LONG,WRT
                     0000   310 ENAB_ROUTINES:                        ; Define base of array of routines to
                     0000   311                                        ; enable CRD interrupts in memories.
                     0000   312 ;
                     0000   313 ; The following macro invocations add elements to the above arrays for each
                     0000   314 ; memory type.
                     0000   315 ;
                     0000   316
                00000000  0000   317 MEMTYPCNT = 0
                00000000  0000   318 GENERAL_MEMTYP = 0
                     0000   319
                     0000   320          MEMORY_ROUTINES -                 ; MS780C memory controller.
                     0000   321                  MEMTYPES<NDT$_MEM4NI,NDT$_MEM4I,NDT$_MEM16NI,NDT$_MEM16I>, -
                     0000   322                  LOGERR_RTN = LOG_MS780C, -
                     0000   323                  LOGALL_RTN = LOGC, -
                     0000   324                  ENAB_RTN = ENAB_MS780C
                     0000   325
                     0000   326          MEMORY_ROUTINES -                 ; MA780 memory controller.
                     0000   327                  MEMTYPES<NDT$_MPM0,NDT$_MPM1,NDT$_MPM2,NDT$_MPM3>, -
                     0000   328                  LOGERR_RTN = LOG_MA780, -
                     0000   329                  LOGALL_RTN = LOGMA, -
                     0000   330                  ENAB_RTN = ENAB_MA780
                     0000   331
                     0000   332          MEMORY_ROUTINES -                 ; MS780E memory controller.
                     0000   333                  MEMTYPES<NDT$_MEM64NIL,NDT$_MEM64EIL,NDT$_MEM64NIU, -
                     0000   334                           NDT$_MEM64EIU,NDT$_MEM64I, -
                     0000   335                           NDT$_MEM256NIL,NDT$_MEM256EIL,NDT$_MEM256NIU, -
                     0000   336                           NDT$_MEM256EIU,NDT$_MEM256I>, -
                     0000   337                  LOGERR_RTN = LOG_MS780E, -
                     0000   338                  LOGALL_RTN = LOGE, -
                     0000   339                  ENAB_RTN = ENAB_MS780E
```

```
                     0000   341          .SBTTL  LOCAL DATA STORAGE
                     0000   342
                     0000   343 ;
                     0000   344 ; Macro that will define a global name of the form MPS$ if
                     0000   345 ; MPSWITCH is defined, else EXE$.  It will also define a local name
                     0000   346 ; to be used within this module.
                     0000   347 ;
                     0000   348          .MACRO  GBLDEF  NAME
                     0000   349          .IF     DF,MPSWITCH                     ; For secondary processor only code...
                     0000   350 MPS$'NAME'::
                     0000   351          .IFF                                    ; For MCHECK780...
                     0000   352 EXE$'NAME'::
                     0000   353          .ENDC
                     0000   354 'NAME':                                         ; For local use...
                     0000   355          .ENDM   GBLDEF
                     0000   356
                     0000   358          .PSECT  MCHK$DATA,QUAD,WRT
                     0000   362 ; The following symbol is defined for a transfer vectror in SYSLOAVEC
                     0000   363 ; This location is NEVER JUMPED TO.  It is defined so these counters
                     0000   364 ; Can be located using a global symbol in the system map.
                     0000   365
                     0000   366 GBLDEF  MCHK_ERRCNT                             ;GLOBAL SYMBOL FOR SYSLOAVEC POINTER
                     0000   367 GBLDEF  GL_CSBITA                               ;USED TO HOLD COMPLEMENT OF SBITA
00000000             0000   368                  .LONG   0
                     0004   369 GBLDEF  GL_CH1OLD                              ;TIME OF LAST CACHE ERROR
00000000             0004   370                  .LONG   0
                     0008   371 GBLDEF  GL_CH2OLD                              ;TIME OF NEXT-TO-LAST CACHE ERROR
00000000             0008   372                  .LONG   0
                     000C   373 GBLDEF  GL_CPTIMOUT                           ;TIME OF LAST CP TIMEOUT/SBI ERROR
00000000             000C   374                  .LONG   0
                     0010   375 GBLDEF  AB_MEMERR                             ;ERROR COUNTERS FOR 16 ADAPTERS
00000020             0010   376                  .BLKB   16
                     0020   377 GBLDEF  GW_REENAB                             ;REENABLE TIMER
     0000            0020   378                  .WORD   0
                     0022   379 GBLDEF  GW_WATCH                              ;SCAN MEMORY CONTROLLER TIMER
     0000            0022   380                  .WORD   0
                     0024   381 GBLDEF  GL_CRDCNT                             ;COUNT OF CORRECTED MEMORY ERRORS
00000000             0024   382                  .LONG   0
                     0028   383 GBLDEF  GL_CHSTATE                            ;CURRENT STATE OF CACHE
00200000             0028   384                  .LONG   ^X200000
                     002C   385 GBLDEF  GL_BADTIMOUT                          ;TIME SINCE LAST BAD MCHK CODE
00000000             002C   386                  .LONG   0
                     0030   387
```

```
                          0030   389          .SBTTL  MACHINE CHECK ENTRY POINT
                    00000000   391          .PSECT  MCHK$CODE,LONG,WRT
                          0000   395
                          0000   396 ;        MACHINE CHECK ENTRY POINT - SCB VECTOR POINTS HERE.
                          0000   397 ;        IPL ^X1F = 31
                          0000   398
                          0000   399          .ALIGN  LONG                        ;A VECTOR MUST HAVE LONGWORD ALIGNMENT
                          0000   400 GBLDEF   MCHK                               ;EITHER EXE$MCHK:: OR MPS$MCHK::
                          0000   401
                       01 DD      0000   402          PUSHL   #MCHK$M_LOG             ;MASK WORD FOR PRTCTEST
                     30 AE DF     0002   403          PUSHAL  MCL_PC+4(SP)            ;PC,PSL POINTER FOR PRTCTEST
                  103F 8F BB      0005   404          PUSHR   #^M<R0,R1,R2,R3,R4,R5,AP>
              5C  5E 24 C1        0009   405          ADDL3   #<9*4>,SP,AP            ;POINT AP TO LOG FRAME ON STACK
                                  000D   406                                          ;ALL INTERRUPTS ARE LOCKED OUT!
                                  000D   407
                       50 33 DB   000D   408          MFPR    #PR780$_SBIMT,R0        ;GET CURRENT CACHE STATE
           50  50 F3 8F 78        0010   409          ASHL    #-CH$V_REPLG1,R0,R0     ;POSITION THE CACHE CONTROL BITS
                    0D 50 F0      0015   410          INSV    R0,#CH$V_REPLG1,-       ;SAVE THE CURRENT CACHE CONTROL BITS
                0028'CF 04        0018   411                  #CH$S_CONTROL,W^GL_CHSTATE
           33 0021C000 8F DA      001C   412          MTPR    #CH_REPAIR,#PR780$_SBIMT ;FORCE MISSES AND GROUP 0 REPLACE,
                                  0023   413                                          ; BUT ALLOW SBI TO INVALIDATE CACHE
        7E 04 AC F0 8F 8B         0023   414          BICB3   #^XF0,MCL_SUMMARY(AP),-(SP) ;GET LOW 4 BITS OF TYPE CODE
                                  0029   415          CASE    (SP)+,<-                ;BREAKOUT TYPE CODE
                                  0029   416                  CPTIMEOUT,-             ;CPU TIMEOUT/SBI ERROR CONFIRMATION
                                  0029   417                  CSPARITY,-              ;CONTROL STORE PARITY ERROR
                                  0029   418                  TBUFPARITY,-            ;TRANSLATION BUFFER PARITY ERROR
                                  0029   419                  CACHEPARITY,-           ;CACHE PARITY ERROR
                                  0029   420                  BADTYPE,-               ;THIS CODE DOESN'T EXIST
                                  0029   421                  READSUBST,-             ;READ DATA SUBSTITUTE (MEM READ ERROR)
                                  0029   422                  IBROMCHECK,-            ;"CAN'T GET HERE" ERROR FROM INST ROMS
                                  0029   423                  BADTYPE,-
                                  0029   424                  BADTYPE,-
                                  0029   425                  BADTYPE,-
                                  0029   426                  TBUFPARITY,-            ;IB-DETECTED TBUF ERROR
                                  0029   427                  BADTYPE,-
                                  0029   428                  READSUBST,-             ;IB-DETECTED MEMORY ERROR
                                  0029   429                  CPTIMEOUT,-             ;IB-DETECTED TIMEOUT OR SBI ERROR CONF
                                  0029   430                  BADTYPE,-
                                  0029   431                  CACHEPARITY>, TYPE=B    ;IB-DETECTED CACHE PROBLEM
                                  004D   432
                                  004D   433 BADTYPE:
     33   0028'CF    DA            004D   434          MTPR    W^GL_CHSTATE,#PR780$_SBIMT ;RE-ENABLE THE CACHE
       FC AC    02  C8            0052   435          BISL    #MCHK$M_MCK,-4(AP)      ;MASK FOR PRTCTEST
           002C'CF  DD            0056   436          PUSHL   W^GL_BADTIMOUT          ;TIME OF LAST BAD TYPE FAULT
    002C'CF    1B  DB            005A   437          MFPR    #PR780$_TODR,W^GL_BADTIMOUT ;TIME OF CURRENT FAULT
    002C'CF    8E  D1            005F   438          CMPL    (SP)+,W^GL_BADTIMOUT    ;COMING TO FAST?
               20  12            0064   439          BNEQ    DAMPUTATE               ;YES, ABORT
                                  0066   440 100$:
           103F 8F  BA            0066   441          POPR    #^M<R0,R1,R2,R3,R4,R5,AP>
 00000000'GF    16               006A   443          JSB     G^EXE$MCHK_BUGCHK       ;RECOVERY BLOCK ENABLED?
                                  0070   444          BUG_CHECK BADMCKCOD,FATAL       ;BAD MACHINE CHECK CODE
```

```
                                0074   449          .SBTTL  TRANSLATION BUFFER PARITY ERRORS
                                0074   450
                                0074   451 TBUFPARITY:
      33   0028'CF    DA       0074   452          MTPR    W^GL_CHSTATE,#PR780$_SBIMT ;RE-ENABLE CACHE
           39    00    DA       0079   453          MTPR    #0,#PR$_TBIA               ;CLEAR ENTIRE TBUF
        FC AC    02    C8       007C   454          BISL    #MCHK$M_MCK,-4(AP)         ;SET MACHINE CHECK CODE FOR PRTCTEST
                                0080   455
                                0080   456
                                0080   457 TRYRESUME:
   04 AC    01F0 8F    B3       0080   458          BITW    #^X1F0,MCL_SUMMARY(AP)     ;IS ERROR ABORT OR TIMEOUT PENDING
                                0086   459 DAMPUTATE:
                 2F    12       0086   460          BNEQ    AMPUTATE                   ;BRANCH IF YES, NO HOPE OF RESUMING
        04 AC    08    93       0088   461          BITB    #8,MCL_SUMMARY(AP)         ;SEE IF ERROR WAS IB ERROR
                 0A    12       008C   462          BNEQ    10$                        ;IF SO, WE CAN "DEFINITELY" RESUME
        7E    2C BC    9A       008E   463          MOVZBL  @MCL_PC(AP),-(SP)          ;GET OPCODE FOR RESTARTABILITY CHECK
1F 070D'CF    8E    E1       0092   464          BBC     (SP)+,W^RESUMABLE,AMPUTATE ;BRANCH IF INST NOT RESUMABLE,ABORT
                                0098   465 10$:    ;THERE IS A LOW PROBABILITY CASE HERE THAT MAY ALLOW THIS CODE TO
                                0098   466          ;CONTINUE WHEN WE CAN'T - IF A LOCATION IS READ FROM THE IO PAGE AND
                                0098   467          ;HAS A SIDE AFFECT WHICH MODIFIES THAT LOCATION, THE INSTRUCTION IS
                                0098   468          ;NOT RETRYABLE, A SOFTWARE SOLUTION IS TO IMPLEMENT A FLAG SET BY ANY
                                0098   469          ;POTENTIAL REFERENCE TO THE IO PAGE THAT MAY CAUSE A SIDE AFFECT.
                                0098   470          ;BBS    #IOSAFLAG,FLAG,AMPUTATE ;BRANCH IF INST MAY OF HAD SIDE AFFECT
                                0098   471 RESUME:
           53    02    B0       0098   472          MOVW    #EMB$K_MC,R3               ;SET TYPE OF LOG ENTRY
              022E    30       009B   473          BSBW    LOGGER                     ;WE'RE GOING TO MAKE IT - LOG ERROR
           103F 8F    BA       009E   474          POPR    #^M<R0,R1,R2,R3,R4,R5,AP> ;RESTORE REGISTERS
           5E    08    C0       00A2   475          ADDL    #8,SP                      ;REMOVE PRTCTEST STUFF FROM STACK
           5E    8E    C0       00A5   476          ADDL    (SP)+,SP                   ;POP HARDWARE LOG FROM STACK
                 02    02       00A8   477          REI                                ;AND TRY AGAIN
```

```
                                00A9   479          .SBTTL  ERRORS DETECTED IN INSTRUCTION DECODE ROMS
                                00A9   480          .SBTTL  CONTROL STORE PARITY ERRORS
                                00A9   481
                                00A9   482 IBROMCHECK:
                                00A9   483 CSPARITY:
            33   0028'CF    DA  00A9   484          MTPR    W^GL_CHSTATE,#PR780$_SBIMT ;CACHE PROBABLY OK - ENABLE IT
          06 AC    2C BC    90  00AE   485          MOVB    @MCL_PC(AP),MCL_SUMMARY+2(AP) ;SAVE OPCODE IN LOG
             FC AC    02    C8  00B3   486          BISL    #MCHK$M_MCK,-4(AP)      ;SET MASK CODE FOR PRTCTEST
                             00B7   487 AMPUTATE:
                 53   02    B0  00B7   488          MOVW    #EMB$K_MC,R3            ;SET TYPE OF LOG ENTRY
                    020F    30  00BA   489          BSBW    LOGGER                  ;LOG THE ERROR
          0E 30 AC    19    E0  00BD   490          BBS     #PSL$V_CURMOD+1,MCL_PSL(AP),REFLECTCHK ;BRANCH IF
                             00C2   491                                          ;FAILURE IN USER OR SUPERVISOR MODE
                             00C2   492
                 103F   8F    BA  00C2   493          POPR    #^M<R0,R1,R2,R3,R4,R5,AP> ;RESTORE REGS
           00000000'GF    16  00C6   495          JSB     G^EXE$MCHK_BUGCHK       ;RECOVERY BLOCK IN EFFECT?
                             00CC   496          BUG_CHECK MACHINECHK,FATAL      ;MACHINE CHECK IN KERNEL OR EXEC MODE
                             00D0   500
                             00D0   510 REFLECTCHK:
                 50   00    DB  00D0   511          MFPR    #PR$_KSP,R0             ;GET THE KERNEL MODE STACK POINTER
             70   2C AC    7D  00D3   512          MOVQ    MCL_PC(AP),-(R0)        ;INTERRUPT PC,PSL TO KERNEL STACK
                             00D7   513                                          ;IT IS NOT NECCESARY TO PROBE KERNEL
                             00D7   514                                          ;STACK FOR VALIDITY, THE FAILURE WILL
                             00D7   515                                          ;BE A KERNEL STACK NOT VALID BUGCHECK
                             00D7   516                                          ;FROM WITHIN MACHINE CHECK
                 00   50    DA  00D7   517          MTPR    R0,#PR$_KSP             ;REPLACE THE NEW KERNEL STACK POINTER
                 103F 8F    BA  00DA   518          POPR    #^M<R0,R1,R2,R3,R4,R5,AP>       ;RESTORE REGISTERS
                 5E   08    C0  00DE   519          ADDL    #8,SP                   ;CLEAR PRTCTST STUFF
                 5E   8E    C0  00E1   520          ADDL    (SP)+,SP                ;POP HARDWARE LOG FROM STACK
        6E    00000000'GF    9E  00E4   521          MOVAB   G^EXE$MCHECK,(SP)       ;SET UP A PC AND PSL FOR EXCEPTION
  04 AE   04 AE    02    18    EF  00FB   522          EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,4(SP),4(SP)
                             00F2   523                                          ;GET MODE WE WERE EXECUTING IN
     04 AE   04 AE    16    9C  00F2   524          ROTL    #PSL$V_PRVMOD,4(SP),4(SP) ;CREATE A PSL OF CURRENT TO BE
                             00F8   525                                          ;KERNEL WITH CORRECT PREVIOUS MODE
                             00F8   526                                          ;AS FROM A FAULT, 0'S IN REST OF PSL
                          02 00F8   528          REI                             ;GET TO EXCEPTION HANDLER
```

```
                                00F9   539          .SBTTL  CACHE PARITY ERROR
                                00F9   540
                                00F9   541 CACHEPARITY:                                ;ENTER WITH CACHE DISABLED REPLACING
                                00F9   542                                             ; GROUP 0
                     FC AC    02   C8  00F9   543          BISL    #MCHK$M_MCK,-4(AP)       ;SET MACHINE CHECK TYPE FOR PRTCTEST
                        10 BC     95  00FD   544          TSTB    @MCL_VA(AP)              ;FORCE DATA INTO GROUP 0 OF BAD CACHE
              33   0021A000 8F     DA  0100   545          MTPR    #CH_REPAIR_1,#PR780$_SBIMT ;NOW FORCE GROUP 1 REPLACEMENT AND
                        10 BC     95  0107   546          TSTB    @MCL_VA(AP)              ;FORCE GROUP 1 OF BAD LINE TO GOOD DATA
                     7E   1B     DB  010A   547          MFPR    #PR780$_TODR,-(SP)       ;GET TIME-OF-YEAR IN 10MS TICKS
              7E    6E   0008'CF   C3  010D   548          SUBL3   W^GL_CH2OLD,(SP),-(SP)   ;GET TIME SPAN OF LAST 2 ERRORS-NOW
                        0A   8E     D1  0113   549          CMPL    (SP)+,#CH_THRESHOLD      ;ARE THE ERRORS WIDELY SPACED
                             2E     1A  0116   550          BGTRU   20$                      ;BRANCH IF YES TO FORGIVE THE CACHE
0000007F 8F    24 AC    07   03  ED  0118   551          CMPZV   #CH$V_GOERRS,#CH$S_GOERRS,MCL_PARITY(AP),#^B1111111
                                    0122   552                                           ;IS GROUP 0 ALL GOOD?
                             11     12  0122   553          BNEQ    10$                      ;BRANCH IF GROUP 0 WAS BAD
                0029'CF    C0 8F     88  0124   554          BISB    #CH_MISSG1!CH_REPLG0@-8,W^GL_CHSTATE+1 ;DISABLE GROUP 1
                   0029'CF    20     8A  012A   555          BICB    #CH_REPLG1@-8,W^GL_CHSTATE+1 ;CANNOT FORCE REPLACE IN BOTH
                     07 AC    02     90  012F   556          MOVB    #CH[OG_DISAB1,MCL_SUMMARY+3(AP) ;LOG THAT WE DID IT
                             11     11  0133   557          BRB     20$
             0029'CF    0120 8F     A8  0135   558 10$:      BISW    #CH_MISSG0!CH_REPLG1@-8,W^GL_CHSTATE+1 ;DISABLE GROUP 0
               0029'CF    40 8F     8A  013C   559          BICB    #CH_REPLG0@-8,W^GL_CHSTATE+1 ;DON'T FORCE REPLACE IN BOTH!
                     07 AC    01     90  0142   560          MOVB    #CH[OG_DISAB0,MCL_SUMMARY+3(AP) ;LOG THAT WE DID IT
             0008'CF    0004'CF     D0  0146   561 20$:      MOVL    W^GL_CH1OLD,W^GL_CH2OLD ;MAINTAIN THE TIMING HISTORY
                0004'CF    8E     D0  014D   562          MOVL    (SP)+,W^GL_CH1OLD        ;UNTO THE THIRD GENERATION
                   33   0028'CF   DA  0152   563          MTPR    W^GL_CHSTATE,#PR780$_SBIMT ;RE-ENABLE THE CACHE - FINALLY!
                        FF26   31  0157   564 BRESUM:  BRW     TRYRESUME                ;SEE IF WE CAN CONTINUE FROM THE ERROR
```

```
                                015A   566          .SBTTL  CP TIMEOUT / SBI ERROR CONFIRMATION
                                015A   567
                                015A   568 CPTIMEOUT:
          33   0028'CF    DA   015A   569          MTPR    W^GL_CHSTATE,#PR780$_SBIMT ;ENABLE THE CACHE
            FC AC    06    C8   015F   570          BISL    #MCHK$M_MCK!MCHK$M_NEXM,-4(AP) ;SET TYPE FOR PRTCTEST
                                0163   571
                                0163   573 ; Add check for read of BRRVR register in UBA.  If this machine check
                                0163   574 ; is the result of a BRRVR read, then just REI.  Someone will either loose
                                0163   575 ; a character from a terminal, or a device timeout will result.  This is
                                0163   576 ; better than a system crash.
                                0163   577
     50   00000000'GF    D0   0163   578          MOVL    G^IOC$GL_ADPLIST,R0      ;POINT TO ADP LIST
                    29    13   016A   579 5$:      BEQL    25$                      ;DONE IF NOTHING LEFT ON LIST
      0000'8F    0E A0    B1   016C   580          CMPW    ADP$W_ADPTYPE(R0),#AT$_UBA ;IS THIS ADP FOR A UBA?
                    1B    12   0172   581          BNEQ    20$                      ;NO, LOOK AT REST OF LIST
                 51   03    D0   0174   582          MOVL    #3,R1                    ;LOOK AT VA'S OF ALL 4 BRRVR REGISTERS
          52   44 A041    09    C1   0177   583 10$:     ADDL3   #9,ADP$L_UBASCB(R0)[R1],R2 ;CALCULATE ADDRESS OF BRRVR FROM
                                017D   584                                           ;THE SCB VECTOR ADDRESS SAVED IN THE ADP
                                017D   585 ; **** NOTE ****  THE PREVIOUS INSTRUCTION ASSUMES THE CURRENTLY USED CODING
                                017D   586 ; SEQUENCE FOR DISPATCHING UBA INTERRUPTS IN THE MODULE INIADP.MAR.  ANY
                                017D   587 ; CHANGES TO THAT CODE MY AFFECT THIS ROUTINE.  THE ASSUMPTIONS ARE THAT THE
                                017D   588 ; VIRTUAL ADDRESS OF THE UBA BRRVR REGISTER IS AT AN OFFSET OF 10. BYTES PAST
                                017D   589 ; THE INTERRUPT VECTOR ADDRESS (9 IS ADDED TO THE SCB VECTOR VALUE BECAUSE
                                017D   590 ; THE VECTOR HAS BIT 0 SET TO INDICATE HANDLING THE INTERRUPT ON THE INTERRUPT
                                017D   591 ; STACK), THAT THE PC OF THE INSTRUCTION ACCESSING BRRVR IS 3 BYTES PAST THE
                                017D   592 ; INTERRUPT VECTOR ENTRY, AND THAT R4 AND R5 HAVE BEEN PUSHED ONTO THE STACK.
                                017D   593
           10 AC    62    D1   017D   594          CMPL    (R2),MCL_VA(AP)          ;SAME VA AS IN MACHINE CHECK STACK?
                    09    12   0181   595          BNEQ    15$                      ;BRANCH IF NOT BRRVR REFERENCE
                       52    06    C2   0183   596          SUBL    #<9-3>,R2                ;ELSE BACK UP TO PC OF INSTRUCTION
                                0186   597                                           ; IN UB INTERRUPT SERVICE THAT READS BRRVR
           2C AC    52    D1   0186   598          CMPL    R2,MCL_PC(AP)            ;DOES IT MATCH PC IN MCHECK STACK?
                    1C    13   018A   599          BEQL    30$                      ;BRANCH IF SO, DEFINITELY CAME FROM
                                018C   600                                           ; UB INTERRUPT SERVICE.
                 E8 51    F4   018C   601 15$:     SOBGEQ  R1,10$                   ;LOOP THROUGH ALL 4
           50    04 A0    D0   018F   602 20$:     MOVL    ADP$L_LINK(R0),R0        ;FOLLOW ADP LIST TO END
                    D5    11   0193   603          BRB     5$
                                0195   604 25$:
                                0195   605
        000C'CF    DD   0195   607          PUSHL   W^GL_CPTIMOUT            ;WE ONLY KEEP TRACK OF ONE TIMEOUT
000C'CF    1B    DB   0199   608          MFPR    #PR780$_TODR,W^GL_CPTIMOUT ;UPDATE THAT HISTORY
000C'CF    8E    D1   019E   609          CMPL    (SP)+,W^GL_CPTIMOUT      ;ARE TIMEOUTS LESS THAN 10 MS APART?
                    B2    12   01A3   610          BNEQ    BRESUM                   ;BRANCH IF NOT TO TRY AND CONTINUE
                  FF0F    31   01A5   611          BRW     AMPUTATE                 ;OTHERWISE SOMETHING IS VERY WRONG
                                01A8   612
                                01A8   614 ; This machine check was a CPU timeout caused by a read of the BRRVR register
                                01A8   615 ; in the UBA.  Log the error as a machine check, clean up the stack and REI.
                                01A8   616 ; This ignores the interrupt.
                                01A8   617
                                01A8   618 ; ********* WHAT IS THE "REAL" STATE OF THE UBA AT THIS POINT? *********
                                01A8   619
           53    02    B0   01A8   620 30$:     MOVW    #EMB$K_MC,R3             ;LOG THE ERROR AS A MACHINE CHECK
              011E    30   01AB   621          BSBW    LOGGER
           103F 8F    BA   01AE   622          POPR    #^M<R0,R1,R2,R3,R4,R5,AP> ;RESTORE SAVED REGISTERS
           5E    08    C0   01B2   623          ADDL    #8,SP                    ;CLEAR PRTCTEST STUFF FROM STACK
           5E    8E    C0   01B5   624          ADDL    (SP)+,SP                 ;CLEAR MACHINE CHECK FRAME FROM STACK
           5E    08    C0   01B8   625          ADDL    #8,SP                    ;CLEAR MACHINE CHECK PC,PSL FROM STACK
```

```
                                                I 10
54   8E   7D  01BB   626          MOVQ    (SP)+,R4                ;R4 AND R5 SAVED BY UBA INT DESPATCHER
          02  01BE   627          REI                             ;REI ON THE UBA INTERRUPT PC,PSL
              01BF   628
```

```
                                01BF    631          .SBTTL  READ DATA SUBSTITUTE ERROR
                                01BF    632
                                01BF    633
                                01BF    634          .ENABL  LSB
                                01BF    635 READSUBST:
            33    0028'CF     DA  01BF    636          MTPR    W^GL_CHSTATE,#PR780$_SBIMT ;REENABLE CACHE
              FC AC    02     C8  01C4    637          BISL    #MCHK$M_MCK,-4(AP)      ;SET MACHINE CHECK TYPE FOR PRTCTEST
              51   2C AC     DE  01C8    638          MOVAL   MCL_PC(AP),R1           ;SET POINTER TO PC,PSL
                 53    08     D0  01CC    639          MOVL    #EMB$K_HE,R3            ;SET HARD MEMORY ERROR TYPE
                     0290     30  01CF    640          BSBW    LOG_ERROR_MEM           ;LOG MEMORY ERROR
      7E    30 AC    05   10   EF  01D2    641          EXTZV   #PSL$V_IPL,#PSL$S_IPL,MCL_PSL(AP),-(SP) ;GET IPL WE WERE AT
                     02   8E   D1  01D8    642          CMPL    (SP)+,#IPL$_ASTDEL      ;ARE WE AT A NON-PAGEABLE PRIORITY?
                          51   14  01DB    643          BGTR    BAMPUTATE               ;ABORT - RECOVERY IS USELESS
                 52    10 AC     D0  01DD    644          MOVL    MCL_VA(AP),R2           ;GET VIRTUAL ADDRESS OF ERROR
            54    00000000'GF     D0  01E1    645          MOVL    G^SCH$GL_CURPCB,R4      ;CURRENT USER'S PCB ADDRESS
                  55    6C A4     D0  01E8    646          MOVL    PCB$L_PHD(R4),R5        ;CURRENT USERS PROCESS HEADER ADDRESS
                 00000000'GF     16  01EC    647          JSB     G^MMG$SVAPTECHK         ;TURN VA INTO VA OF PTE
                      50   63     D0  01F2    648          MOVL    (R3),R0                 ;GET THE PTE WHICH MAPS THE BAD PAGE
                           03     19  01F5    649          BLSS    5$                      ;BRANCH IF PAGE VALID
                         00C4     31  01F7    650          BRW     RDSNONRES               ;ELSE FATAL ERROR
                                  01FA    651 5$:
                    30 50   15    E0  01FA    652          BBS     #PTE$V_WINDOW,R0,BAMPUTATE ;BR IF PAGE IS PFN-MAPPED
                                  01FE    653          ASSUME  PTE$V_PFN EQ 0
               50   50   15   00   EF  01FE    654          EXTZV   #PTE$V_PFN,#PTE$S_PFN,R0,R0 ;ISOLATE PAGE FRAME NUMBER IN PTE
                00000000'GF    50   D1  0203    655          CMPL    R0,G^MMG$GL_MAXPFN      ;IS THERE PFN DATA BASE FOR PAG?
                               22   1A  020A    656          BGTRU   BAMPUTATE              ;BR IF NO PFN DATA BASE FOR PAGE
                7E   00000000'GF    D0  020C    657          MOVL    G^PFN$AB_TYPE,-(SP)     ;PFN TYPE ARRAY ADDRESS *********
                       9E40   20   88  0213    658          BISB    #PFN$M_BADPAG,@(SP)+[R0] ;MARK PAGE BAD          *******
                              51   D4  0217    659          CLRL    R1                      ;CLEAR MODIFY BIT PROPAGATOR
                    04 63     1A   E5  0219    660          BBCC    #PTE$V_MODIFY,(R3),10$  ;TEST (& CLEAR) MODIFY BIT IN PTE
                    51   80 8F     9A  021D    661          MOVZBL  #PFN$M_MODIFY,R1        ;SET MODIFY PROPAGATOR
                7E   00000000'GF    D0  0221    662 10$:     MOVL    G^PFN$AB_STATE,-(SP)    ;ADDRESS OF PFN STATE ARRAY ******
                       9E40   51   88  0228    663          BISB    R1,@(SP)+[R0]           ;PROPAGATE MODIFY BIT TO PFN DATABASE **
                                  022C    664
                                  022C    665          ASSUME  PFN$M_MODIFY EQ 128
                                  022C    666
                              03  14  022C    667          BGTR    15$                     ;PAGE NOT MODIFIED - HE'S OK
                                  022E    668 BAMPUTATE:
                            FE86  31  022E    669          BRW     AMPUTATE
                 04 AC    01F0  8F  B3  0231    670 15$:     BITW    #^X1F0,MCL_SUMMARY(AP) ;WAS ERROR FAULT OR ABORT?
                              F5  12  0237    671          BNEQ    BAMPUTATE               ;ABORT - DON'T TRY ANY REPAIRS
                       04 AC    08  93  0239    672          BITB    #8,MCL_SUMMARY(AP)      ;IS ERROR IB ERROR FAULT
                              0A  12  023D    673          BNEQ    20$                     ;BRANCH IF YES, IB ERRORS RESUME
                          7E   2C  BC  023F    674          MOVZBL  @MCL_PC(AP),-(SP)       ;GET OPCODE FOR RESTARTABILITY CHECK
                 E5 070D'CF       8E  0243    675          BBC     (SP)+,W^RESUMABLE,BAMPUTATE ;BRANCH IF INST NOT RESUMABLE
                7E   00000000'GF    D0  0249    676 20$:     MOVL    G^PFN$AW_REFCNT,-(SP)   ;ADDRESS OF PFN REFCNT ARRAY *****
                       01   9E40   B1  0250    677          CMPW    @(SP)+[R0],#1           ;CHECK FOR I/O IN PROGRESS, ETC. ***
                              D8  1A  0254    678          BGTRU   BAMPUTATE               ;IF SO, DON'T TRY ANYTHING FANCY
                7E   00000000'GF    D0  0256    679          MOVL    G^PFN$AB_TYPE,-(SP)     ;ADDRESS OF PFN TYPE ARRAY  *****
            01   9E40   03   00   EC  025D    680          CMPV    #PFN$V_PAGTYP,#PFN$S_PAGTYP,@(SP)+[R0],#PFN$C_SYSTEM ;****
                                  0263    681                                              ;CHECK FOR SYSTEM OR GLOBAL PAGE
                                  0263    682
                                  0263    683          ASSUME  PFN$C_SYSTEM EQ 1       ;CHECK TYPE OF PAGE
                                  0263    684          ASSUME  PFN$C_PROCESS EQ 0
                                  0263    685
                              C9  1A  0263    686          BGTRU   BAMPUTATE               ;BRANCH IF GLOBAL PAGE
                                  0265    687
                                  0265    688 ;        IN THE FUTURE WE MAY RECOVER FROM HARD ECC ERRORS ON GLOBAL PAGES
```

```
                                 0265    691 ;       AS WELL, BUT FOR NOW WE ABORT THE IMAGE.
                                 0265    692
                                 0265    693
                      52    D5   0265    694 30$:    TSTL    R2                      ;CHECK THE VIRTUAL ADDRESS
                      0B    14   0267    695         BGTR    40$                     ;BRANCH IF PAGE IS PROCESS PRIVATE
              54  00000000'GF    9E   0269    696         MOVAB   G^MMG$AL_SYSPCB,R4       ;SYSTEM PAGES ARE KEPT TRACK OF IN
               55    6C  A4     D0   0270    697         MOVL    PCB$L_PHD(R4),R5        ;A WORKING SET LIST IN THE SYSTEM PCB
               00 63    1F     E5   0274    698 40$:    BBCC    #PTE$V_VALID,(R3),50$   ;CLEAR VALID BIT FROM PTE
                                 0278    699 50$:    INVALID R2                      ;INVALIDATE TRANSLATION BUFFER OF VA
              7E  00000000'GF    D0   027B    700         MOVL    G^PFN$AW_REFCNT,-(SP)   ;ADDRESS OF PFN REFCNT ARRAY  *****
                        9E40     B7   0282    701         DECW    @(SP)+[R0]              ;REDUCE REFERENCE COUNT TO 0 *****
                          06     18   0285    702         BGEQ    60$                     ;
                  00000000'GF    16   0287    703         JSB     G^MMG$REFCNTNEG         ;
              7E  00000000'GF    D0   028D    704 60$:    MOVL    G^PFN$Ax_WSLX,-(SP)     ;ADDRESS OF PFN WSLX ARRAY  *****
                                 0294    705                 PFN_REFERENCE   -
                                 0294    706         MOVZWL  <@(SP)+[R0],R1>,-       ;GET WORKING-SET LIST INDEX FOR PAGE **
                                 0294    707                 LONG_OPCODE=MOVL,-
                                 0294    708                 IMAGE=SYSLOA780.EXE
                  00000000'GF    16   02A6    709         JSB     G^MMG$DELWSLEX          ;DELETE IT FROM THE WORKING SET ITS IN
                  00000000'GF    16   02AC    710         JSB     G^MMG$DELCONPFN         ;DELETE PAGE FROM PAGE TABLE
                        52    02   9A   02B2    711         MOVZBL  #PFN$C_BADPAGLST,R2     ;SET UP LIST TO PUT PAGE ON
                  00000000'GF    16   02B5    712         JSB     G^MMG$INSPFNT           ;PUT PAGE ONTO BAD PAGE LIST
                                 02BB    713
                                 02BB    714 ; AT THIS POINT, THE PTE FOR THE BAD PAGE CONTAINS ITS MASS STORAGE
                                 02BB    715 ; ADDRESS. THIS WILL CAUSE A FRESH COPY OF THE PAGE TO BE FETCHED WHEN
                                 02BB    716 ; THE PROCESS IS RESUMED.
                                 02BB    717
                      FDDA    31   02BB    718         BRW     RESUME                  ;LOG MACHINE CHECK AND RESUME PROCESS
                                 02BE    719
                                 02BE    720 RDSNONRES:                              ;
                                 02BE    721
                    103F  8F     BA   02BE    722         POPR    #^M<R0,R1,R2,R3,R4,R5,AP>
                  00000000'GF    16   02C2    723         JSB     G^EXE$MCHK_BUGCHK       ;RECOVERY BLOCK IN EFFECT?
                                 02C8    724         BUG_CHECK RDSNONRES,FATAL       ;READ DATA SUBSTITUTE PAGE NONRESIDENT
                                 02CC    725
                                 02CC    726         .DSABL  LSB
                                 02CC    727
```

```
                                02CC    730          .SBTTL  INTERFACE FROM MACHINE CHECK HANDLER TO ERROR LOGGER
                                02CC    731 ;++
                                02CC    732 ; LOGGER - Routine to log Machine Check interrupts and aborts
                                02CC    733 ;
                                02CC    734 ; INPUTS:
                                02CC    735 ;
                                02CC    736 ;       R3 - Error log type
                                02CC    737 ;       AP - Pointer to Machine Check error log frame
                                02CC    738 ;       -4(AP) - MASK FOR PRTCTEST
                                02CC    739 ;       -8(AP) - PC,PSL POINTER FOR PRTCTEST
                                02CC    740 ;
                                02CC    741 ; OUTPUTS:
                                02CC    742 ;
                                02CC    743 ;       Entry made in error log conditional on PRTCTEST
                                02CC    744 ;       R0-R5 destroyed
                                02CC    745 ;--
                                02CC    746
                                02CC    747 LOGGER:
      54   08   6C    C1       02CC    748          ADDL3   MCL_COUNT(AP),#<2*4>,R4 ;GET SIZE OF ENTRY IN BYTES
        55   04   AC    9E       02D0    749          MOVAB   MCL_SUMMARY(AP),R5      ;GET ADDRESS OF ENTRY
        51   F8   AC    7D       02D4    750          MOVQ    -8(AP),R1               ;GET MASK AND PC POINTER FOR PRTCTEST
 00000000'GF    16    02D8    752          JSB     G^EXE$MCHK_TEST         ;ARE WE TO LOG THIS ERROR?
         06   50    E8       02DE    753          BLBS    R0,10$                  ;NO, RECOVERY BLOCK IGNORES IT
 00000000'GF    D6    02E1    755          INCL    G^EXE$GL_MCHKERRS       ;KEEP COUNT OF MACHINE CHECKS
                                02E7    756 10$:                                ;FALL THROUGH TO "LOGIT"
                                02E7    757
                                02E7    758 ;++
                                02E7    759 ; LOGIT - INTERFACE TO SYSTEM ERROR LOG
                                02E7    760 ;
                                02E7    761 ; INPUTS:
                                02E7    762 ;
                                02E7    763 ;       R1 = PC,PSL POINTER FOR PRTCTEST
                                02E7    764 ;       R2 = MASK FOR PRTCTEST
                                02E7    765 ;       R3 = ERROR LOG TYPE
                                02E7    766 ;       R4 = SIZE OF LOG ENTRY IN BYTES
                                02E7    767 ;       R5 = ADDRESS OF LOG ENTRY
                                02E7    768 ;       (SP) = RETURN ADDRESS
                                02E7    769 ;--
                                02E7    770          .ENABL  LSB
                                02E7    771
                                02E7    772 LOGIT:
        50   30   DB    02E7    773          MFPR    #PR780$_SBIFS,R0        ;GET SBI FAULT/STATUS REGISTER
     00 50   19   E5    02EA    774          BBCC    #SBIFS$V_NEF,R0,10$     ;CLEAR NESTED ERROR FLAG
        30   50   DA    02EE    775 10$:     MTPR    R0,#PR780$_SBIFS        ;WRITE IT BACK TO CLEAR SILO LOCK
                                02F1    776                                  ;AND FAULT LATCH
        50   34   DB    02F1    777          MFPR    #PR780$_SBIER,R0        ;GET SBI ERROR REGISTER
50      70C0 8F    A8    02F4    778          BISW    #SBIER$M_IBTO!SBIER$M_IBRDS!SBIER$M_CPTO!SBIER$M_RDS!-
                                02F9    779                  SBIER$M_CRD,R0          ;SET BITS FOR ERRORS WE'RE HANDLING
        34   50   DA    02F9    780          MTPR    R0,#PR780$_SBIER        ;WRITE IT BACK TO CLEAR LATCHES
                                02FC    781
 00000000'GF    16    02FC    783          JSB     G^EXE$MCHK_TEST         ;ARE WE TO LOG THIS ERROR?
        21   50   E8    0302    784          BLBS    R0,20$                  ;NO, RECOVERY BLOCK IGNORES IT
                                0305    786
                                0305    787 MCHK$GL_LOG::
                                0305    788
51      54   10   C1    0305    789          ADDL3   #EMB$B_MC_SUMCOD,R4,R1  ;ADD SPACE FOR HEADER FOR BUFFER SIZE
                                0309    790
```

```
                00000000'GF   16  0309   792          JSB     G^ERL$ALLOCEMB          ;GET AN ERROR LOGGING BUFFER
                              030F   796
                      14 50   E9  030F   797          BLBC    R0,20$                  ;BRANCH IF DIDN'T GET IT
                         52   DD  0312   798          PUSHL   R2                      ;SAVE ADDRESS OF ERROR LOG BUFFER
                   04 A2  53   B0  0314   799          MOVW    R3,EMB$W_MC_ENTRY(R2)   ;SET ENTRY TYPE TO FAULT TYPE
               10 A2  65  54   28  0318   800          MOVC3   R4,(R5),EMB$B_MC_SUMCOD(R2) ;IN ONE SWELL FOOP.....
                      52  8E   D0  031D   801          MOVL    (SP)+,R2                ;GET POINTER TO BUFFER START IN R2
                              0320   802
                00000000'GF   16  0320   804          JSB     G^ERL$RELEASEMB         ;INDICATE BUFFER READY TO LOG
                              0326   808
                              05  0326   809 20$:      RSB                             ;EXIT WITH HARDWARE LOG STILL ON STACK
                                  0327   810
                                  0327   811          .DSABL  LSB
```

```
                     0327   813
                     0327   814          .SBTTL  SBI ERROR INTERRUPTS
                     0327   815 ;++
                     0327   816 ; Handle SBI Faults and Asynchronous Write Timeouts on the SBI.
                     0327   817 ;
                     0327   818 ; SBI Fault:
                     0327   819 ;       Log the error; try to resume normal execution.
                     0327   820 ;
                     0327   821 ; Asynchronous Write Timeouts:
                     0327   822 ;       Log the error.
                     0327   823 ;       Set up a "fake" machine check log on the stack.  This is so we
                     0327   824 ;       can share the exception exit path (REFLECTCHK) that machine checks
                     0327   825 ;       take if the current process is executing in USER or SUPER mode.
                     0327   826 ;       If the current process is in EXEC or KERNEL mode, bugcheck.
                     0327   827 ;--
                     0327   828          .ALIGN  LONG                           ;THIS IS VECTORED TO
                     0328   829 GBLDEF   INT5C                                  ;SBI FAULT VECTOR
                     0328   830 GBLDEF   LOGSBF
                     0328   831          SETIPL  #^X1F                          ;DISABLE ALL INTERRUPTS
00FF 8F    BB        032B   832          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7>   ;SAVE SOME WORK REGS
53   04    9A        032F   833          MOVZBL  #EMB$K_BE,R3                   ;ERROR LOG TYPE
52   03    D0        0332   834          MOVL    #MCHK$M_MCK!MCHK$M_LOG,R2 ;MASK FOR PRTCTEST
     43    10        0335   835          BSBB    LOGSBI                         ;USE SAME CODE AS ASYNC WRITE FAILURE
00FF 8F    BA        0337   836          POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7>   ;RESTORE R0-R7
     02              033B   837          REI                                    ;TRY TO CONTINUE
                     033C   838
                     033C   839          .ALIGN  LONG                           ;THIS IS VECTORED TO
                     033C   840 GBLDEF   INT60                                  ;ASYNCHRONOUS WRITE TIMEOUT
                     033C   841 GBLDEF   LOGAWE
                     033C   842          SETIPL  #^X1F                          ;DISABLE ALL INTERRUPTS
     00FF 8F    BB   033F   843          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7>   ;SAVE SOME WORK REGS
     53   07    9A   0343   844          MOVZBL  #EMB$K_AW,R3                   ;ERROR LOG TYPE
     52   07    D0   0346   845          MOVL    #MCHK$M_LOG!MCHK$M_MCK!MCHK$M_NEXM,R2 ;PRTCTEST MASK
          2F    10   0349   846          BSBB    LOGSBI                         ;USE SAME CODE AS SBI FAULT ERROR
     00FF 8F    BA   034B   847          POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7>   ;RESTORE R0-R7
     5E   28    C2   034F   848          SUBL    #40,SP                         ;ALLOCATE FAKE MACHINE CHECK FRAME
          28    DD   0352   849          PUSHL   #40                            ;SIZE OF FRAME
          07    DD   0354   850          PUSHL   #MCHK$M_MCK!MCHK$M_LOG!MCHK$M_NEXM
       30 AE    DF   0356   851          PUSHAL  MCL_PC+4(SP)                   ;MASK AND PC,PSL FOR PRTCTEST
   103F 8F    BB   0359   852          PUSHR   #^M<R0,R1,R2,R3,R4,R5,AP> ;SAVE REGISTERS FOR COMMON CODE
5C 5E    24    C1   035D   853          ADDL3   #<9*4>,SP,AP                   ;POINT AP TO FAKE MACHINE CHECK FRAME
0003'CF  A0 8F    93   0361   854          BITB    #^B10100000,W^GL_CSBITA+3 ;WAS WRITE IN USER OR SUPERVISOR
                     0367   855                                                 ;MODE AND NOT UPDATING A PAGE TABLE
          03    12   0367   856          BNEQ    10$                            ;IF NOT, MUST BUGCHECK
        FD64    31   0369   857          BRW     REFLECTCHK                     ;BRANCH IF OK TO CONTINUE
                     036C   858 10$:
   103F 8F    BA   036C   859          POPR    #^M<R0,R1,R2,R3,R4,R5,AP>
00000000'GF   16   0370   861          JSB     G^EXE$MCHK_BUGCHK              ;RECOVERY BLOCK IN EFFECT?
                     0376   862          BUG_CHECK ASYNCWRTER,FATAL             ;WRITE ERROR IN KERNAL OR EXEC MODE
                     037A   863                                                 ;OR WHILE UPDATING PAGE TABLE
```

```
                                037A   868 ;++
                                037A   869 ; LOGSBI -- Subroutine to log SBI errors.
                                037A   870 ;
                                037A   871 ; Implicit Inputs:
                                037A   872 ;           +------------------------+
                                037A   873 ;           |   return address       | :(SP)
                                037A   874 ;           +------------------------+
                                037A   875 ;           |       saved            |
                                037A   876 ;           |       R0 - R7          |
                                037A   877 ;           +------------------------+
                                037A   878 ;           |   interrupt PC         |
                                037A   879 ;           +------------------------+
                                037A   880 ;           |   interrupt PSL        |
                                037A   881 ;           +------------------------+
                                037A   882 ;
                                037A   883 ; Create an SBI error log buffer that contains:
                                037A   885 ;           The contents of the configuration register of every
                                037A   886 ;           SBI adapter on the bus or 0 (16 longwords).
                                037A   891 ;           A copy of the SBI silo (16 longwords).
                                037A   892 ;           SBI processor registers SBITA, SBIER, SBIMT, SBISC, and SBIFS.
                                037A   893 ;--
                                037A   894 LOGSBI:                                      ;LOG SBI ERROR
        00000000'GF    16       037A   896             JSB     G^EXE$MCHK_TEST         ;ARE WE TO LOG THIS ERROR?
              06 50    E8       0380   897             BLBS    R0,5$                   ;NO, RECOVERY BLOCK IGNORES IT
        00000000'GF    D6       0383   899             INCL    G^EXE$GL_MCHKERRS       ;KEEP COUNT OF MACHINE CHECKS
                                0389   900 5$:
         7E   24 AE    7D       0389   901             MOVQ    <9*4>(SP),-(SP)         ;MAKE A SECOND COPY OF PC,PSL
     57  00000000'GF    D0       038D   902             MOVL    G^EXE$GL_CONFREGL,R7    ;ARRAY OF NEXUS DEVICE TYPE CODES
     55  00000000'GF    D0       0394   903             MOVL    G^MMG$GL_SBICONF,R5     ;ARRAY OF ADAPTER VA'S
           50    0F    D0       039B   904             MOVL    #15,R0                  ;INDEX OF LAST POSSIBLE ITEM ON SBI
              7E       D4       039E   905 10$:        CLRL    -(SP)                   ;ASSUME NO ADAPTOR HERE
        51    6540     D0       03A0   906             MOVL    (R5)[R0],R1             ;GET VA OF CONTROLLER/ADAPTER
              08       18       03A4   907             BGEQ    20$                     ;GEQ IMPLIES NO VALID SYSTEM VA.
           6740        D5       03A6   908             TSTL    (R7)[R0]                ;TEST ADAPTER TYPE (ONLY WORKS FOR SBI)
              03       13       03A9   909             BEQL    20$                     ;IF EQL, NO ADAPTOR HERE
           6E   61     D0       03AB   919             MOVL    (R1),(SP)               ;STORE ADAPTOR CSR0 ON STACK
             ED 50     F4       03AE   920 20$:        SOBGEQ  R0,10$                  ;LOOP THRU ALL POSSIBLE 16
           50   0F     D0       03B1   921             MOVL    #15,R0                  ;SET UP COUNT OF NUMBER OF TIMES TO
                                03B4   922                                             ;READ SILO
           7E   31     DB       03B4   923 30$:        MFPR    #PR780$_SBIS,-(SP)      ;SAVE INFORMATION FOR ERROR LOGGER
             FA 50     F4       03B7   924             SOBGEQ  R0,30$                  ;LOOP THRU ALL 16
           7E   35     DB       03BA   925             MFPR    #PR780$_SBITA,-(SP)     ;SAVE SBI TIMEOUT REGISTER
0000'CF    7E   6E     D2       03BD   926             MCOML   (SP),W^GL_CSBITA         ;SAVE COMPLEMENT SBITA FOR LATER CHECK
           7E   34     DB       03C2   927             MFPR    #PR780$_SBIER,-(SP)     ;SAVE SBI ERROR REGISTER
           7E   33     DB       03C5   928             MFPR    #PR780$_SBIMT,-(SP)     ;SAVE SBI MAINTENANCE REGISTER
           7E   32     DB       03C8   929             MFPR    #PR780$_SBISC,-(SP)     ;SAVE SBI SILO COMPARATOR
           7E   30     DB       03CB   930             MFPR    #PR780$_SBIFS,-(SP)     ;SAVE SBI FAULT/STATUS REGISTER
7E    009C 8F         3C       03CE   931             MOVZWL  #<16*4>+<16*4>+<7*4>,-(SP) ;SAVE NUMBER OF BYTES OF ENTRY
                                03D3   932
51    0098 CE         DE       03D3   933             MOVAL   <<16*4>+<16*4>+<6*4>>(SP),R1 ;ADDRESS OF PC,PSL FOR PRTCTEST
      54    6E        D0       03D8   934             MOVL    (SP),R4                 ;# OF BYTES TO LOG
  55    04 AE         9E       03DB   935             MOVAB   4(SP),R5                ;ADDRESS OF LOG ENTRY
           FF05       30       03DF   936             BSBW    LOGIT                   ;CALL ERROR LOGGER
           5E   8E    C0       03E2   937             ADDL    (SP)+,SP                ;CLEAN STACK OF LOG AND FAKE PC,PSL
                05    05       03E5   938             RSB                             ;RETURN
                                03E6   939
```

```
                                03E6   941          .SBTTL  MEMORY TIMER SCAN
                                03E6   942
                                03E6   944 ECC$REENABLE::                          ; Define timer scan entry point.
                                03E6   948
                  0022'CF   B7  03E6   949          DECW    W^GW_WATCH             ; Count seconds down.
                        19  14  03EA   950          BGTR    REENABLE_INTS          ; Br if time hasn't elapsed yet.
                                03EC   951 ;
                                03EC   952 ; Scan all memory controllers for unreported errors.  This will yield a
                                03EC   953 ; representative sample of memory errors in the error log, even if CRD
                                03EC   954 ; interrupts are disabled.
                                03EC   955 ;
              0022'CF   3C  B0  03EC   956          MOVW    #SOMETIME,W^GW_WATCH   ; Reset time interval.
                        0A  BB  03F1   957          PUSHR   #^M<R1,R3>             ; Save working registers.
                        7E  DC  03F3   958          MOVPSL  -(SP)                  ; Set up interrupt PSL.
                        00  10  03F5   959          BSBB    10$                    ; Fake interrupt PC on stack.
                    51  6E  DE  03F7   960 10$:     MOVAL   (SP),R1                ; Point to exception PC,PSL.
                    53  06  D0  03FA   961          MOVL    #EMB$K_SE,R3           ; Log soft memory errors.
                       0062  30  03FD   962          BSBW    LOG_ERROR_MEM          ; Scan all memory controllers, and
                                0400   963                                         ; log any that report errors.
                    5E  08  C0  0400   964          ADDL    #8,SP                  ; Pop PC,PSL pair from stack.
                        0A  BA  0403   965          POPR    #^M<R1,R3>             ; Restore registers.
                                0405   966
                                0405   967 REENABLE_INTS:
                  0020'CF   B7  0405   968          DECW    W^GW_REENAB            ; Count seconds down.
                        2B  14  0409   969          BGTR    10$                    ; Branch if reenable time not elapsed.
          0020'CF   0384  8F  B0  040B   970          MOVW    #REENABTIME,W^GW_REENAB ; Reset the time interval.
                        3F  BB  0412   971          PUSHR   #^M<R0,R1,R2,R3,R4,R5> ; Save working registers.
0010'CF   10   00   00 8F   00  2C  0414   972          MOVC5   #0,#0,#0,#16,W^AB_MEMERR; Reset 16 bytes of error count to 0.
         00000000'GF   00'  E1  041D   973          BBC     S^#EXE$V_CRDENABL, -   ; Branch if SYSGEN parameter specifies
                        0F         0424   974                  G^EXE$GL_FLAGS,5$      ; no CRD interrupts.
           55   00000000'GF   D0  0425   975          MOVL    G^MMG$GL_SBICONF,R5    ; Get addr of SBICONF for action routines.
               53   0000'CF   DE  042C   976          MOVAL   W^ENAB_ROUTINES,R3     ; Array of action routine vectors.
                       008C   30  0431   977          BSBW    LOCATE_MEM             ; Locate mem and call action routines
                                0434   978                                         ; to re-enable CRD interrupts.
                         3F   BA  0434   979 5$:      POPR    #^M<R0,R1,R2,R3,R4,R5> ; Restore registers.
                              05  0436   980 10$:     RSB
```

```
                                                D 11
                     0437   982          .SBTTL  Memory Error Interrupts
                     0437   983 ;
                     0437   984 ; SBI Alert interrupts are vectored here.
                     0437   985 ;
                     0437   986          .ALIGN  LONG
                     0438   987 GBLDEF   INT58                           ; EXE$INT58:: or MPS$INT58::
                     0438   988 GBLDEF   LOGSBA                          ; EXE$LOGSBA:: or MPS$LOGSBA::
                     0438   989
          0A   BB    0438   990          PUSHR   #^M<R1,R3>               ; Save some registers.
                     043A   991          SETIPL  #^X1F                    ; Disable all interrupts.
    51  08 AE   DE  043D   992          MOVAL   8(SP),R1                 ; Set pointer to interrupt PC,PSL.
        53   05   D0  0441   993          MOVL    #EMB$K_SA,R3             ; Set SBI Alert error log type.
        001B 30   0444   994          BSBW    LOG_ERROR_MEM            ; Log memory controller registers.
          0A   BA  0447   995          POPR    #^M<R1,R3>               ; Restore registers.
               02  0449   996          REI
                     044A   997
                     044A   998 ;
                     044A   999 ; CRD (Soft, or Corrected) memory error interrupts are vectored here.
                     044A  1000 ;
                     044A  1001          .ALIGN  LONG
                     044C  1002 GBLDEF   INT54                           ; EXE$INT54:: or MPS$INT54::
                     044C  1003 GBLDEF   LOGCRD                          ; EXE$LOGCRD:: or MPS$LOGCRD::
                     044C  1004
          0A   BB    044C  1005          PUSHR   #^M<R1,R3>               ; Save some registers.
                     044E  1006          SETIPL  #^X1F                    ; Disable all interrupts.
  0024'CF   D6  0451  1007          INCL    W^GL_CRDCNT              ; Keep count of these errors.
    51  08 AE   DE  0455  1008          MOVAL   8(SP),R1                 ; Set pointer to interrupt PC,PSL.
        53   06   D0  0459  1009          MOVL    #EMB$K_SE,R3             ; Set soft memory error type.
        0003 30   045C  1010          BSBW    LOG_ERROR_MEM            ; Log memory controller registers.
          0A   BA  045F  1011          POPR    #^M<R1,R3>
               02  0461  1012          REI
```

```
                     0462  1014          .SBTTL  LOGMEM Master Routine
                     0462  1015 ;++
                     0462  1016 ;
                     0462  1017 ; FUNCTIONAL DESCRIPTION:
                     0462  1018 ;       This routine is called to build an errorlog containing the device
                     0462  1019 ;       registers of the memory controllers on an 11/780 system.  If called
                     0462  1020 ;       at the LOG_ERROR_MEM entry point, it will scan the memory controller
                     0462  1021 ;       status registers, and only log those controllers which report errors.
                     0462  1022 ;       If called at the LOG_ALL_MEM entry point, it will unconditionally log
                     0462  1023 ;       all memory controllers on the system.
                     0462  1024 ;
                     0462  1025 ; INPUTS:
                     0462  1026 ;       R1      - pointer to exception PC,PSL
                     0462  1027 ;       R3      - Error log type code (e.g. EMB$K_type)
                     0462  1028 ;
                     0462  1029 ; OUTPUTS:
                     0462  1030 ;       Format of error log:
                     0462  1031 ;               # of memory controllers logged
                     0462  1032 ;               memory type-specific log #1
                     0462  1033 ;               memory type-specific log #2
                     0462  1034 ;                       .
                     0462  1035 ;                       .
                     0462  1036 ;               PC of instruction at fault time
                     0462  1037 ;               PSL at fault time
                     0462  1038 ;
                     0462  1039 ;       All registers are preserved.
                     0462  1040 ;
                     0462  1041 ;--
                     0462  1042
                     0462  1043 LOG_ERROR_MEM:                          ; Log controllers with errors.
          1FFF 8F    BB   0462  1044          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP>
     53   0000'CF    DE   0466  1045          MOVAL   W^LOGERR_ROUTINES,R3    ; Array of action routine vectors.
            09       11   046B  1046          BRB     LOGMEM                  ; Join common code.
                     046D  1047
                     046D  1048 LOG_ALL_MEM:                            ; Unconditionally log all controllers.
          1FFF 8F    BB   046D  1049          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP>
     53   0000'CF    DE   0471  1050          MOVAL   W^LOGALL_ROUTINES,R3    ; Array of action routine vectors.
                     0476  1051
                     0476  1052 LOGMEM:                                  ; Log memory controller registers.
            55       7C   0476  1053          CLRQ    R5                      ; Zero error log byte count and number
                     0478  1054                                           ; of controllers logged.
57   00000000'GF    D0   0478  1055          MOVL    G^MMG$GL_SBICONF,R7     ; For use by action routines.
          5C   01    D0   047F  1056          MOVL    #SS$_NORMAL,AP          ; Assume no fatal memory errors.
                     0482  1057 ;
                     0482  1058 ; Locate all memory controllers on the SBI.  When a memory controller is
                     0482  1059 ; found, call the appropriate action routine to create that controller's
                     0482  1060 ; portion of the common error log buffer on the stack.
                     0482  1061 ;
              003B   30   0482  1062          BSBW    LOCATE_MEM
                     0485  1063 ;
                     0485  1064 ; The error log buffer has been built on the stack; SP points to the beginning.
                     0485  1065 ; Add the number of memory controllers logged, then log the errors.
                     0485  1066 ; Current register usage:
                     0485  1067 ;       R5      - Number of bytes in the error log.
                     0485  1068 ;       R6      - Number of memory controllers logged.
                     0485  1069 ;       SP      - Points to the beginning of the error log buffer.
                     0485  1070 ;       AP      - LBS if no fatal memory errors were discovered, else LBC.
```

```
                        0485  1071 ;
          51   6E45  9E  0485  1072          MOVAB   (SP)[R5],R1             ; Get address of saved R0 on stack.
        53   0C A1  D0  0489  1073          MOVL    12(R1),R3               ; Restore input value of R3.
        51   04 A1  D0  048D  1074          MOVL    4(R1),R1                ; Restore input value of R1.
                56  DD  0491  1075          PUSHL   R6                      ; Add # of controllers to log buffer.
      7E    55   04  C1  0493  1076          ADDL3   #<1*4>,R5,-(SP)         ; Total # bytes in error log buffer.
                55  D5  0497  1077          TSTL    R5                      ; Were any memory registers logged?
                12  13  0499  1078          BEQL    10$                     ; No.  Skip call to error logger.
       00000000'GF  D6  049B  1079          INCL    G^EXE$GL_MEMERRS        ; Keep count of memory errors
            54   6E  D0  04A1  1080          MOVL    (SP),R4                 ; Use # bytes as input to LOGIT.
        55   04 AE  DE  04A4  1081          MOVAL   4(SP),R5                ; Address of error log buffer.
                52  D4  04A8  1082          CLRL    R2                      ; Always log memory errors.
              FE3A  30  04AA  1083          BSBW    LOGIT                   ; Log the error.
            5E   8E  C0  04AD  1084 10$:     ADDL    (SP)+,SP                ; Remove error log buffer from stack.
              08 5C  E8  04B0  1085          BLBS    AP,20$                  ; Br if fatal error not signalled.
            1FFF 8F  BA  04B3  1086          POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP>
                        04B7  1088          BUG_CHECK ASYNCWRTER,FATAL      ; Unrecoverable memory controller error.
                        04BB  1092 20$:
            1FFF 8F  BA  04BB  1093          POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP>
                        05  04BF  1094          RSB
```

```
                        04C0  1096          .SBTTL  LOCATE_MEM Dispatching Routine
                        04C0  1097 ;++
                        04C0  1098 ;       Routine to locate memory controllers on 11/780 SBI.
                        04C0  1099 ;
                        04C0  1100 ; FUNCTIONAL DESCRIPTION:
                        04C0  1101 ;       This routine scans an array of adapter type codes that tell which
                        04C0  1102 ;       adapters are attached to the SBI.  When it finds a memory controller
                        04C0  1103 ;       adapter, it dispatches to an action routine for that memory controller
                        04C0  1104 ;       type.
                        04C0  1105 ;
                        04C0  1106 ; INPUTS:
                        04C0  1107 ;       R3 - address of action routine table; 1 action routine/memory controller
                        04C0  1108 ;       Current format of action routine tables (the tables are created by the
                        04C0  1109 ;       MEMORY_ROUTINES macro):
                        04C0  1110 ;               (R3):  self-relative offset to MS780C action routine
                        04C0  1111 ;             4(R3):  self-relative offset to MA780 action routine
                        04C0  1112 ;             8(R3):  self-relative offset to MS780E action routine
                        04C0  1113 ;
                        04C0  1114 ; On entry to memory action routine:
                        04C0  1115 ;       R0,R1   - local registers, not preserved across calls to action routines
                        04C0  1116 ;       R2      - TR# of this memory controller
                        04C0  1117 ;       R3      - not available to be used by action routines
                        04C0  1118 ;       R4      - address of CONFREGL array (If the 780 ever gets a BI, code
                        04C0  1119 ;                 must change, because TSTL assumes no high-order bits set.)
                        04C0  1120 ;       R5-AP   - available; contents are preserved across calls to multiple
                        04C0  1121 ;                 action routines (i.e. can be used for global storage)
                        04C0  1122 ;
                        04C0  1123 ;       Note: an action routine may deposit a -1 in R2 to cause LOCATE_MEM
                        04C0  1124 ;       to prematurely exit the memory scan loop (and not call any other
                        04C0  1125 ;       memory action routines).
                        04C0  1126 ;
                        04C0  1127 ; OUTPUTS:
                        04C0  1128 ;       R0-R4 destroyed.  (Other registers may be destroyed by action routines.)
                        04C0  1129 ;--
                        04C0  1130
                        04C0  1131 LOCATE_MEM:
      54   00000000'GF   D0  04C0  1132          MOVL    G^EXE$GL_CONFREGL,R4     ; Get address of CONFREGL.
52    00000000'GF   01   C3  04C7  1133          SUBL3   #1,G^EXE$GL_NUMNEXUS,R2 ; Get index into nexus arrays.
                        04CF  1134 ;
                        04CF  1135 ; Loop through all nexuses.  If a memory controller is found at any of the
                        04CF  1136 ; nexus slots, then call the action routine associated with that memory.
                        04CF  1137 ;
            51   6442   D0  04CF  1138 10$:     MOVL    (R4)[R2],R1              ; Get nexus device type from CONFREGL.
                   14   13  04D3  1139          BEQL    20$                      ; Not a memory; go to next nexus.
    0000'CF   12   51   3A  04D5  1140          LOCC    R1,#MEMTYPCNT,W^MEMTYP   ; Find type in memory type array.
                        04DB  1141                                               ; R1 <- addr of type code (if found).
                   0C   13  04DB  1142          BEQL    20$                      ; Not a memory; go to next nexus.
          51   12 A1   9A  04DD  1143          MOVZBL  MEMTYPCNT(R1),R1         ; Use offset to get general memory type.
           51   6341   DE  04E1  1144          MOVAL   (R3)[R1],R1              ; Get self-relative address of action
             00 B141   16  04E5  1145          JSB     @(R1)[R1]                ; routine, and call it.
                E3 52   F4  04E9  1146 20$:     SOBGEQ  R2,10$                   ; Loop through all nexuses.
                        05  04EC  1147          RSB                              ; Return.
```

```
                                                M 11
                     04ED  1149          .SBTTL  ENAB Action Routines
                     04ED  1150 ;++
                     04ED  1151 ;
                     04ED  1152 ; FUNCTIONAL DESCRIPTION:
                     04ED  1153 ;       These action routines re-enable CRD interrupts for each 11/780 memory
                     04ED  1154 ;       controller.  Memory types currently supported:
                     04ED  1155 ;
                     04ED  1156 ;               MS780C (local memory - 4k and 16k chips)
                     04ED  1157 ;               MS780E (local memory - 64k chips)
                     04ED  1158 ;               MA780  (multiport memory)
                     04ED  1159 ;
                     04ED  1160 ; INPUTS:
                     04ED  1161 ;       R2      - TR# of this memory
                     04ED  1162 ;       R4      - address of EXE$GL_CONFREGL array
                     04ED  1163 ;       R5      - address of MMG$GL_SBICONF array
                     04ED  1164 ;
                     04ED  1165 ; OUTPUTS:
                     04ED  1166 ;       R0,R1 destroyed; all other registers preserved.
                     04ED  1167 ;
                     04ED  1168 ;--
                     04ED  1169 ENAB_MS780C:
           51   6542 D0  04ED  1171          MOVL    (R5)[R2],R1             ; Get address of controller registers.
08 A1 30000000 8F   D0  04F1  1172          MOVL    #<MRC$M_ELSRF!MRC$M_HERIMF>,8(R1) ; Enable interrupts
                     04F9  1173                                              ; and clear error flags.
                  05 04F9  1175          RSB                             ; That's it.
                     04FA  1176
                     04FA  1177 ENAB_MS780E:
           51   6542 D0  04FA  1179          MOVL    (R5)[R2],R1             ; Get address of controller registers.
08 A1 30000000 8F   D0  04FE  1180          MOVL    #<MRC$M_ELSRF!MRC$M_HERIMF>,8(R1) ; Enable interrupts
                     0506  1181                                              ; and clear error flags in 1st contr.
0C A1 30000000 8F   D0  0506  1182          MOVL    #<MRC$M_ELSRF!MRC$M_HERIMF>,12(R1) ; Enable interrupts
                     050E  1183                                              ; and clear error flags in 2nd contr.
                  05 050E  1185          RSB                             ; That's it.
                     050F  1186
                     050F  1187 ENAB_MA780:
           51   6542 D0  050F  1188          MOVL    (R5)[R2],R1             ; Get address of controller registers.
                     0513  1190          $PRTCTINI       B^10$, -        ; Protect against non-existent memory
                     0513  1191                          #<MCHK$M_LOG!MCHK$M_NEXM> ; machine checks.
10 A1 30000000 8F   C8  051F  1193          BISL    #<MRC$M_ELSRF!MRC$M_HERIMF>,16(R1) ; Enable interrupts
                     0527  1194                                              ; and clear error flags.
                     0527  1196          $PRTCTEND       10$             ; End of protected code.
                  05 0528  1198          RSB                             ; That's it.
```

```
0529  1200          .SBTTL  LOGMEM Action Routines
0529  1201 ;++
0529  1202 ;  FUNCTIONAL DESCRIPTION:
0529  1203 ;       One action routine per memory controller type follows.  These
0529  1204 ;       routines create an 11/780 memory error log entry.  Currently, the
0529  1205 ;       following memory controllers are supported:
0529  1206 ;
0529  1207 ;               MS780C (local memory - 4K and 16K chips)
0529  1208 ;               MS780E (local memory - 64K chips)
0529  1209 ;               MA780  (multiport memory)
0529  1210 ;
0529  1211 ;       Each action routine contributes to the common error log buffer being
0529  1212 ;       built on the stack.  Because different routines are being used to build
0529  1213 ;       a common error log buffer on the stack, the contents of the stack is
0529  1214 ;       significant at all times.
0529  1215 ;
0529  1216 ; INPUTS:
0529  1217 ;       R2 - nexus index for this memory (TR #)
0529  1218 ;       R3 - not available for use by action routines
0529  1219 ;       R4 - address of SBI configuration array (CONFREGL)
0529  1220 ;       R5 - current errorlog byte count
0529  1221 ;       R6 - current number of controllers logged
0529  1222 ;       R7 - address of array of SBI virtual addresses (SBICONF)
0529  1223 ;       R8-R11 - scratch
0529  1224 ;       AP - memory controller status: LBC = fatal controller error discovered
0529  1225 ;
0529  1226 ; IMPLICIT INPUTS:
0529  1227 ;                       +-----------------------------------------+
0529  1228 ;       (SP):           |       caller's return address           |
0529  1229 ;                       +-----------------------------------------+
0529  1230 ;                       |       return to caller's caller         |
0529  1231 ;                       +-----------------------------------------+
0529  1232 ;                       |         previous error log              |
0529  1233 ;                       |                 .                       |
0529  1234 ;                       |                 .                       |
0529  1235 ;
0529  1236 ; OUTPUTS:
0529  1237 ;       R2-R4 preserved
0529  1238 ;       R5 and R6 updated
0529  1239 ;
0529  1240 ; IMPLICIT OUTPUTS:
0529  1241 ;                       +-----------------------------------------+
0529  1242 ;       (SP):           |       return to caller's caller         |
0529  1243 ;                       +-----------------------------------------+
0529  1244 ;                       |  error log entry for this controller    |
0529  1245 ;                       |  (null if no error for this memory)     |
0529  1246 ;                       +-----------------------------------------+
0529  1247 ;                       |       previous error log                |
0529  1248 ;                       |                 .                       |
0529  1249 ;                       |                 .                       |
0529  1250 ;
0529  1251 ;--
```

```
                        0529  1253          .SBTTL  LOG_MS780C
                        0529  1254 ;++
                        0529  1255 ;       LOG_MS780C - Build error log for MS780C memory controller
                        0529  1256 ;
                        0529  1257 ;       The portion of the error log built for the MS780C memory controller
                        0529  1258 ;       has the following format:
                        0529  1259 ;                       +-------------------------------+
                        0529  1260 ;                       |       adapter TR#             |
                        0529  1261 ;                       +-------------------------------+
                        0529  1262 ;                       |       memory register A       |
                        0529  1263 ;                       +-------------------------------+
                        0529  1264 ;                       |       memory register B       |
                        0529  1265 ;                       +-------------------------------+
                        0529  1266 ;                       |       memory register C       |
                        0529  1267 ;                       +-------------------------------+
                        0529  1268 ;       Register A contains the type code in the low-order byte.  For MS780C
                        0529  1269 ;       memories, this type code is in the range of 8 - 11 (hex).
                        0529  1270 ;--
                        0529  1271
                        0529  1272 LOG_MS780C:
                        0529  1273 ;
                        0529  1274 ; Determine whether to log this controller.
                        0529  1275 ;
 58    6742    D0       0529  1277          MOVL    (R7)[R2],R8             ; Get VA of controller registers.
5A    08 A8    D0       052D  1278          MOVL    8(R8),R10               ; Read memory controller register C.
01 5A    1C    E0       0531  1279          BBS     #MRC$V_ELSRF,R10,LOGC   ; Branch if error log requested.
              05       0535  1283          RSB                             ; Else return.
                        0536  1285 ;
                        0536  1286 ; Build error log on stack.
                        0536  1287 ; This is the entry point used when unconditionally logging all memories.
                        0536  1288 ;
                        0536  1289 LOGC:
          03    BA      0536  1290          POPR    #^M<R0,R1>              ; Get return addr in R0, caller's
                        0538  1291                                          ; return in R1.
                        0538  1292          DSBINT  DST=R9                  ; Raise IPL while reading registers.
          5A    DD      053E  1293          PUSHL   R10                     ; Save memory register C in log.
       04 A8    DD      0540  1294          PUSHL   4(R8)                   ; Save memory register B in log.
          68    DD      0543  1295          PUSHL   (R8)                    ; Save memory register A in log.
                        0545  1296          ENBINT  SRC=R9                  ; Drop back to previous IPL.
          52    DD      0548  1297          PUSHL   R2                      ; Save TR# in log.
                        054A  1298 ;
                        054A  1299 ; Check for CRD error.  If the number of recent CRD errors > CRDINTMAX, then
                        054A  1300 ; disable CRD interrupts.  If the number of recent CRD errors > CRDWATCHMAX,
                        054A  1301 ; then stop logging CRD errors.
                        054A  1302 ;
    1E 5A    1C    E1   054A  1303          BBC     #MRC$V_ELSRF,R10,20$    ; Branch if no error log requested.
     0010'CF42    96   054E  1304          INCB    W^AB_MEMERR[R2]         ; Count memory errors for this contr.
03   0010'CF42    91   0553  1305          CMPB    W^AB_MEMERR[R2],#CRDINTMAX      ; Too many CRD interrupts?
          04    15      0559  1306          BLEQ    10$                     ; No, skip CRD interrupt disable.
    00 5A    1E    E2   055B  1307          BBSS    #MRC$V_INHBCRD,R10,10$  ; Set bit to inhibit CRD interrupts.
                        055F  1308 10$:
06   0010'CF42    91   055F  1309          CMPB    W^AB_MEMERR[R2],#CRDWATCHMAX    ; Too many CRD error logs?
          05    15      0565  1310          BLEQ    20$                     ; No, go log this one.
       5E    10    C0   0567  1311          ADDL2   #<4*4>,SP               ; Pop memory CSRs from stack.
          05    11      056A  1312          BRB     30$                     ; Skip logging CRD for this controller.
                        056C  1313 20$:
       55    10    C0   056C  1314          ADDL2   #<4*4>,R5               ; Add # of bytes in this log to total.
```

```
                56   D6  056F  1315          INCL    R6                      ; Count number of controllers logged.
                                 0571  1316 30$:
          08 A8 5A   D0  0571  1317          MOVL    R10,8(R8)               ; Clear errors from register C.
                51   DD  0575  1318          PUSHL   R1                      ; Restore caller's caller to stack.
                60   17  0577  1319          JMP     (R0)                    ; Return to caller.
```

```
                                                L 11
                     0579  1322          .SBTTL  LOG_MS780E
                     0579  1323 ;++
                     0579  1324 ;       LOG_MS780E - Build error log for MS780E memory controller
                     0579  1325 ;
                     0579  1326 ;       The portion of the error log built for the MS780E memory controller
                     0579  1327 ;       has the following format:
                     0579  1328 ;               +--------------------------------+
                     0579  1329 ;               |       adapter TR#              |
                     0579  1330 ;               +--------------------------------+
                     0579  1331 ;               |       memory register A        |
                     0579  1332 ;               +--------------------------------+
                     0579  1333 ;               |       memory register B        |
                     0579  1334 ;               +--------------------------------+
                     0579  1335 ;               |       memory register C        |
                     0579  1336 ;               +--------------------------------+
                     0579  1337 ;               |       memory register D        |
                     0579  1338 ;               +--------------------------------+
                     0579  1339 ;
                     0579  1340 ;       Register A contains the type code in the low-order byte.  For MS780E
                     0579  1341 ;       memories, this type code is in the range of 68 - 6C (hex).
                     0579  1342 ;
                     0579  1343 ;--
                     0579  1344
                     0579  1345 LOG_MS780E:
                     0579  1346 ;
                     0579  1347 ; Determine whether to log any errors for this controller.
                     0579  1348 ;
          58   6742  D0 0579  1350          MOVL    (R7)[R2],R8             ; Get VA of controller registers.
      68  00100000 8F  D3 057D  1351          BITL    #MRC$M_SUMMARY,(R8)     ; Test error summary bit in Register A.
            01  12 0584  1352          BNEQ    LOGE                    ; Branch if there are any errors.
               05 0586  1356          RSB                             ; Else return.
                     0587  1358                                          .
                     0587  1359 ;
                     0587  1360 ; This is the entry point used when unconditionally logging all memories.
                     0587  1361 ;
                     0587  1362 LOGE:
            03   BA  0587  1363          POPR    #^M<R0,R1>              ; Get return address in R0, caller's
                     0589  1364                                          ; return in R1.
      59    03   D0  0589  1365          MOVL    #3,R9                   ; Initialize loop counter.
                     058C  1366          DSBINT  DST=R10                 ; Raise IPL while reading registers.
        6849   DD  0592  1367 10$:     PUSHL   (R8)[R9]                ; Push registers in reverse order.
         FA 59   F4 0595  1368          SOBGEQ  R9,10$                  ; Push 4 registers for error log.
            52   DD  0598  1369          PUSHL   R2                      ; Save TR# in error log.
                     059A  1370          ENBINT  SRC=R10                 ; Drop back to previous IPL.
                     059D  1371 ;
                     059D  1372 ; The MS780E memory subsystem consists of 1 SBI Interface module and 2 Memory
                     059D  1373 ; Controller modules.  Each controller can control up to 8 memory array cards.
                     059D  1374 ; The MS780E subsystem can operate in any one of 3 interleave modes: when both
                     059D  1375 ; Control Cards are present, the subsystem will usually operate in internally
                     059D  1376 ; interleaved mode; if only one Control Card is present, the sub-system may
                     059D  1377 ; operate in non-interleaved mode, or may be externally interleaved with
                     059D  1378 ; another memory subsystem on a different SBI slot.
                     059D  1379 ;
                     059D  1380 ; MS780E controller registers A and B reside on the SBI Interface module and
                     059D  1381 ; are always accessible and valid when the subsystem is present.  Register C
                     059D  1382 ; gives information about the lower controller and array cards while Register
                     059D  1383 ; D gives information about the upper set.  If either controller is not
```

```
                                059D  1384 ; present, its status register (C or D) will be accessible but the contents
                                059D  1385 ; are UNDEFINED.
                                059D  1386 ;
    68    04 AE    D0          059D  1387          MOVL    4(SP),(R8)              ; Clear error bits set in Register A.
04 A8     08 AE    D0          05A1  1388          MOVL    8(SP),4(R8)             ; Clear error bits set in Register B.
                                05A6  1389 ;
                                05A6  1390 ; Check Register A for interleave mode.
                                05A6  1391 ;
          59       7C          05A6  1392          CLRQ    R9                      ; Will hold copies of Register C and D.
05 04 AE    02     E0          05A8  1393          BBS     #2,4(SP),LOWER          ; If internally interleaved, check both
                                05AD  1394                                          ; upper and lower controllers.
11 04 AE    01     E0          05AD  1395          BBS     #1,4(SP),UPPER          ; Branch if upper controller is enabled.
                                05B2  1396 ;
                                05B2  1397 ; Check lower controller for fatal parity errors.
                                05B2  1398 ;
                                05B2  1399 LOWER:
    59    0C AE    D0          05B2  1400          MOVL    12(SP),R9               ; Get copy of Register C from stack.
    3F 59    07     E0          05B6  1401          BBS     #MRC$V_MSEQPTY,R9,-     ; Branch if microsequencer parity
                                05BA  1402                  FATAL_MEM               ; error.
    3B 59    08     E0          05BA  1403          BBS     #MRC$V_IFPTY,R9,-       ; Branch if SBI Interface write data
                                05BE  1404                  FATAL_MEM               ; parity error.
10 04 AE    02     E1          05BE  1405          BBC     #2,4(SP),CHK_CRD_LOW    ; Branch if not internally interleaved.
                                05C3  1406 ;
                                05C3  1407 ; Check upper controller for fatal parity errors.
                                05C3  1408 ;
                                05C3  1409 UPPER:
    5A   10 AE    D0          05C3  1410          MOVL    16(SP),R10              ; Get copy of Register D from stack.
    2E 5A    07     E0          05C7  1411          BBS     #MRC$V_MSEQPTY,R10,-    ; Branch if microsequencer parity
                                05CB  1412                  FATAL_MEM               ; error.
    2A 5A    08     E0          05CB  1413          BBS     #MRC$V_IFPTY,R10,-      ; Branch if SBI interface write data
                                05CF  1414                  FATAL_MEM               ; parity error.
                                05CF  1415 ;
                                05CF  1416 ; Determine if this error was a CRD in either controller.
                                05CF  1417 ;
    04 5A    09     E0          05CF  1418          BBS     #MRC$V_CRDERR,R10,CRD_MS780E ; Branch if CRD error in upper contr.
                                05D3  1419 CHK_CRD_LOW:
    24 59    09     E1          05D3  1420          BBC     #MRC$V_CRDERR,R9,LOG_E  ; Branch if not a CRD error in lower.
                                05D7  1421 ;
                                05D7  1422 ; This was a CRD error.  If the number of recent CRD errors > CRDINTMAX, then
                                05D7  1423 ; disable CRD interrupts for this subsystem.  If the number of recent CRD
                                05D7  1424 ; errors > CRDWATCHMAX, then don't log another CRD error.
                                05D7  1425 ;
                                05D7  1426 ; NOTE: it is always safe to write to both register C and D even if one of the
                                05D7  1427 ; controllers is disabled.
                                05D7  1428 ;
                                05D7  1429 CRD_MS780E:
     0010'CF42      96          05D7  1430          INCB    W^AB_MEMERR[R2]         ; Count memory errors for this contr.
03   0010'CF42      91          05DC  1431          CMPB    W^AB_MEMERR[R2],#CRDINTMAX      ; Too many CRD interrupts?
           08       15          05E2  1432          BLEQ    11$                     ; No, skip CRD interrupt disable.
    00 59    1E     E2          05E4  1433          BBSS    #MRC$V_INHBCRD,R9,10$   ; Set bit to inhibit CRD interrupts.
    00 5A    1E     E2          05E8  1434 10$:     BBSS    #MRC$V_INHBCRD,R10,11$  ; Set bit to inhibit in upper contr.
                                05EC  1435 11$:
06   0010'CF42      91          05EC  1436          CMPB    W^AB_MEMERR[R2],#CRDWATCHMAX    ; Too many CRD error logs?
           07       15          05F2  1437          BLEQ    LOG_E                   ; No, go ahead and log this one.
      5E   14       C0          05F4  1438          ADDL2   #<5*4>,SP               ; Pop memory CSRs from stack.
           07       11          05F7  1439          BRB     CLEAR_ERRS_E            ; Skip logging CRDs for this controller.
                                05F9  1440 ;
```

```
                        05F9  1441 ; Found a fatal controller error.  Report it, try to log the error and return.
                        05F9  1442 ;
                        05F9  1443 FATAL_MEM:
                  5C    D4  05F9  1444          CLRL    AP                      ; Signal fatal memory error.
                        05FB  1445 ;
                        05FB  1446 ; Increment log counts.
                        05FB  1447 ;
                        05FB  1448 LOG_E:
                  55    14  C0  05FB  1449          ADDL2   #<5*4>,R5               ; Add # of bytes in this log to total.
                        56  D6  05FE  1450          INCL    R6                      ; Count number of controllers logged.
                            0600  1451 CLEAR_ERRS_E:
            08 A8       59  D0  0600  1452          MOVL    R9,8(R8)                ; Clear errors from register C.
            0C A8       5A  D0  0604  1453          MOVL    R10,12(R8)              ; Clear errors from register D.
                        51  DD  0608  1454          PUSHL   R1                      ; Restore caller's caller to stack.
                        60  17  060A  1455          JMP     (R0)                    ; Return to caller.
```

```
                                   060C             .SBTTL  LOG_MA780
                                   060C     ;++
                                   060C     ;       LOG_MA780 - Build error log for MA780 memory controller
                                   060C     ;
                                   060C     ;       The portion of the error log built for the MA780 memory controller
                                   060C     ;       has the following format:
                                   060C     ;               +--------------------------------+
                                   060C     ;               !         adapter TR#            !
                                   060C     ;               +--------------------------------+
                                   060C     ;               ! Port Configuration Register    !
                                   060C     ;               +--------------------------------+
                                   060C     ;               ! Port Interface Control Reg     !
                                   060C     ;               +--------------------------------+
                                   060C     ;               ! Port Controller Status Reg     !
                                   060C     ;               +--------------------------------+
                                   060C     ;               ! Port Invalidation Control Reg  !
                                   060C     ;               +--------------------------------+
                                   060C     ;               ! Array Error Register           !
                                   060C     ;               +--------------------------------+
                                   060C     ;               ! Configuration Status Reg 0     !
                                   060C     ;               +--------------------------------+
                                   060C     ;               ! Configuration Status Reg 1     !
                                   060C     ;               +--------------------------------+
                                   060C     ;               ! Maintenance Control Register   !
                                   060C     ;               +--------------------------------+
                                   060C     ;
                                   060C     ;       The Port Configuration Register contains the type code in the
                                   060C     ;       low-order byte.  For MA780 memories, this type code is in the
                                   060C     ;       range of 40 - 43 (hex).
                                   060C     ;
                                   060C     ;--
                                   060C
                                   060C     LOG_MA780:
                                   060C     ;
                                   060C     ; Determine whether to log any errors for this controller.
                                   060C     ;
                                   060C
        58    6742        D0       060C             MOVL    (R7)[R2],R8             ; Get VA of controller register.
                5A        D4       0610             CLRL    R10                     ; Use R10 as memory error flag; assume
                                   0612                                             ; there is an error condition.
                                   0612             $PRTCTINI       B^10$, -        ; Protect following code from
                                   0612                     #<MCHK$M_LOG!MCHK$M_MCK>; all machine checks.
    68     00400000 8F    D3       061E             BITL    #^X00400000,(R8)        ; Check for power-up interrupt.
                2B        12       0625             BNEQ    5$                      ; Branch if found.
 04 A8     FF000000 8F    D3       0627             BITL    #^XFF000000,4(R8)       ; Check Port Interface Control Reg.
                21        12       062F             BNEQ    5$                      ; Branch if found error.
 10 A8     10000000 8F    D3       0631             BITL    #MRC$M_ELSRF,16(R8)     ; Check Array Error register.
                17        12       0639             BNEQ    5$                      ; Branch if found error.
 18 A8     00000400 8F    D3       063B             BITL    #^X00000400,24(R8)      ; Check Multiple Interlock Accepted err.
                0D        12       0643             BNEQ    5$                      ; Branch if found error.
 08 A8     D000C000 8F    D3       0645             BITL    #^XD000C000,8(R8)       ; Lastly, check Port Contr. Status Reg.
                03        12       064D             BNEQ    5$                      ; Branch if found error.
         5A     01        D0       064F             MOVL    #1,R10                  ; Signal no errors found.
                                   0652     5$:
                                   0652             $PRTCTEND       10$             ; End of protected code.
            03 50         E9       0653             BLBC    R0,20$                  ; If MA780 has disappeared, just return.
            01 5A         E9       0656             BLBC    R10,LOGMA               ; If any errors were found, log them.
```

```
                 05  0659  1519 20$:    RSB                             ; Else return.
                         065A  1520 ;
                         065A  1521 ; This is the entry point used when unconditionally logging all memories.
                         065A  1522 ;
                         065A  1523 ; Build error log on stack.  First set SP to where the top of the buffer
                         065A  1524 ; will be, and use R9 as a temporary stack pointer while the log is being
                         065A  1525 ; built.  This is so the machine check protection routines can freely use the
                         065A  1526 ; stack above where the error log is being built.
                         065A  1527 ;
                         065A  1528 LOGMA:
          0802 8F    BA  065A  1529         POPR    #^M<R1,R11>             ; Get return address in R1, caller's
                         065E  1530                                         ; return in R11.
          59   5E    D0  065E  1531         MOVL    SP,R9                   ; Use R9 as temporary stack pointer.
          5E   24    C2  0661  1532         SUBL    #<9*4>,SP               ; Point SP to where stack top will be.
                         0664  1534         $PRTCTINI       W^50$   -       ; Protect following code from
                         0664  1535                 #<MCHK$M_LOG!MCHK$M_NEXM>       ;non-existent memory errors.
                         0671  1537         DSBINT  DST=R10                 ; Raise IPL while logging registers.
     79   1C A8    D0  0677  1538         MOVL    28(R8),-(R9)            ; Maintenance Control Register
     79   18 A8    D0  067B  1539         MOVL    24(R8),-(R9)            ; Configuration Status Register 1
     79   14 A8    D0  067F  1540         MOVL    20(R8),-(R9)            ; Configuration Status Register 0
     79   10 A8    D0  0683  1541         MOVL    16(R8),-(R9)            ; Array Error Register
     79   0C A8    D0  0687  1542         MOVL    12(R8),-(R9)            ; Port Invalidation Control Register
                         068B  1544         $PRTCTINI       B^10$,-         ; Protect this register access against
                         068B  1545                 #<MCHK$M_LOG!MCHK$M_MCK>; all machine checks.
     79   08 A8    D0  0697  1547         MOVL    8(R8),-(R9)             ; Read Port Controller Status Register.
                         069B  1549         $PRTCTEND       10$             ; End of protected code.
          03 50    E8  069C  1550         BLBS    R0,15$                  ; Branch if no machine check occurred.
        79   00    D0  069F  1552         MOVL    #0,-(R9)                ; Else put fake copy of register in log.
                         06A2  1553 15$:
     79   04 A8    D0  06A2  1554         MOVL    4(R8),-(R9)             ; Port Interface Control Register
        79   68    D0  06A6  1555         MOVL    (R8),-(R9)              ; Port Configuration Register
                         06A9  1556         ENBINT  SRC=R10                 ; Restore IPL to previous level.
        79   52    D0  06AC  1557         MOVL    R2,-(R9)                ; Save TR# in error log.
                         06AF  1558 ;
                         06AF  1559 ; Clear errors from registers.
                         06AF  1560 ;
          68   04 A9    C8  06AF  1561         BISL    4(R9),(R8)              ; Clear errs in Port Config Reg (pwr-up)
     04 A8     08 A9    C8  06B3  1562         BISL    8(R9),4(R8)             ; Clear errors in Port Interface
                         06B8  1563                                         ; Control Register.
                         06B8  1565         $PRTCTINI       B^20$,-         ; Protect this register access against
                         06B8  1566                 #<MCHK$M_LOG!MCHK$M_MCK>; all machine checks.
     08 A8     0C A9    C8  06C4  1568         BISL    12(R9),8(R8)            ; Clear errors in Port Controller
                         06C9  1569                                         ; Status register.
                         06C9  1571         $PRTCTEND       20$             ; End of protected code.
     14 A8     18 A9    C8  06CA  1573         BISL    24(R9),20(R8)           ; Clear errors in Port Configuration
                         06CF  1574                                         ; Status Register (Mult Interlock Accpt)
               14 A9    DD  06CF  1575         PUSHL   20(R9)                  ; Get copy of Array Error Register
                         06D2  1576                                         ; on top of stack.
                         06D2  1577 ;
                         06D2  1578 ; Check for CRD error.  If the # of recent CRD errors > CRDINTMAX, then disable
                         06D2  1579 ; CRD interrupts for this controller.  If the # of recent CRD errors >
                         06D2  1580 ; CRDWATCHMAX, then don't log another CRD error for this controller.
                         06D2  1581 ;
        1E 6E     1C    E1  06D2  1582         BBC     #MRC$V_ELSRF,(SP),40$   ; Branch if this wasn't a data error.
         0010'CF42    96  06D6  1583         INCB    W^AB_MEMERR[R2]         ; Count data errors for this contr.
03       0010'CF42    91  06DB  1584         CMPB    W^AB_MEMERR[R2],#CRDINTMAX      ; Too many CRD interrupts?
                   04    15  06E1  1585         BLEQ    30$                     ; No, skip CRD interrupt disable.
```

```
                                                D 12
                00 6E    1E    E2  06E3  1586          BBSS    #MRC$V_INHBCRD,(SP),30$ ; Set bit to inhibit CRD interrupts.
                                06E7  1587 30$:
                   5A    01    D0  06E7  1588          MOVL    #1,R10                  ; Assume error will be logged.
              06    0010'CF42   91  06EA  1589          CMPB    W^AB_MEMERR[R2],#CRDWATCHMAX    ; Too many CRD error logs?
                         02    15  06F0  1590          BLEQ    40$                     ; No, go ahead and log this one.
                         5A    D4  06F2  1591          CLRL    R10                     ; Signal "don't log this error".
                                06F4  1592 40$:
                10 A8    8E    D0  06F4  1593          MOVL    (SP)+,16(R8)            ; Clear errors from Array Error Reg.
                                06F8  1594
                                06F8  1596          $PRTCTEND       50$             ; End of protected code.
                                06F9  1598 ;
                                06F9  1599 ; Note: If no machine check occurred, R9 and SP are now identical.  We can
                                06F9  1600 ; resume using SP.
                                06F9  1601 ;
                   03 50    E9  06F9  1603          BLBC    R0,NOLOG_MA             ; MA780 disappeared, nothing to log
                   05 5A    E8  06FC  1605          BLBS    R10,LOG_MA              ; Branch to log the error.
                                06FF  1607 NOLOG_MA:
                   5E    24  C0  06FF  1608          ADDL    #<9*4>,SP               ; Clean error log off the stack.
                         05    11  0702  1609          BRB     EXIT_MA                 ; And return.
                                0704  1611 LOG_MA:
                   55    24  C0  0704  1612          ADDL    #<9*4>,R5               ; Add # of bytes in this log to total.
                         56    D6  0707  1613          INCL    R6                      ; Increment count of memories logged.
                                0709  1614 EXIT_MA:
                         5B    DD  0709  1615          PUSHL   R11                     ; Restore caller's caller to stack.
                         61    17  070B  1616          JMP     (R1)                    ; Return to caller.
```

```
                070D  1618          .SBTTL  TABLE OF RESUMABLE INSTRUCTIONS.
                070D  1619 ;        EACH BIT IN THE TABLE IS A 1 IF THE INSTRUCTION IS RESUMABLE,
                070D  1620 ;        AND A 0 IF IT IS NOT.
                070D  1621
                070D  1622 RESUMABLE:
3C3B            070D  1623          .WORD   ^B0011110000111011      ;REI, LDPCTX, SVPCTX, INSQUE, REMQUE
                070F  1624                                          ;CVTPS, CVTSP
FFFF            070F  1625          .WORD   ^B1111111111111111
FF00            0711  1626          .WORD   ^B1111111100000000      ;PACKED DECIMAL INSTRUCTIONS
FEFF            0713  1627          .WORD   ^B1111111011111111      ;EDITPC
FFFF            0715  1628          .WORD   ^B1111111111111111
002F            0717  1629          .WORD   ^B0000000000101111      ;EMODF,CVTFD,INTERLOCKED INSTRUCTIONS
0F00            0719  1630          .WORD   ^B0000111100000000      ;DOUBLE PRECISION FLOATING POINT
C14A            071B  1631          .WORD   ^B1100000101001010      ;MORE DOUBLE PREC/QUAD, EMUL, EDIV
FFFF            071D  1632          .WORD   ^B1111111111111111
FFFF            071F  1633          .WORD   ^B1111111111111111
FFFF            0721  1634          .WORD   ^B1111111111111111
F3FF            0723  1635          .WORD   ^B1111001111111111      ;PUSHR, POPR
FFFF            0725  1636          .WORD   ^B1111111111111111
F4FF            0727  1637          .WORD   ^B1111010011111111      ;ADWC, SBWC, MFPR
FF3F            0729  1638          .WORD   ^B1111111100111111      ;BBSSI, BBCCI
00FF            072B  1639          .WORD   ^B0000000011111111      ;ASHP, CVTLP, CALLG, CALLS, XFC, EXPANSION
                072D  1640
                072D  1641          .end
```

```
AB_MEMERR                   00000010 R      07
ADP$L_LINK                = 00000004
ADP$L_UBASCB              = 00000044
ADP$W_ADPTYPE             = 0000000E
AMPUTATE                    000000B7 R      08
AT$_UBA                     ********   X    08
BADTYPE                     0000004D R      08
BAMPUTATE                   0000022E R      08
BRESUM                      00000157 R      08
BUG$_ASYNCWRTER             ********   X    08
BUG$_BADMCKCOD              ********   X    08
BUG$_MACHINECHK             ********   X    08
BUG$_RDSNONRES              ********   X    08
CACHEPARITY                 000000F9 R      08
CH$S_CONTROL              = 00000004
CH$S_GOERRS               = 00000007
CH$V_GOERRS               = 00000003
CH$V_REPLG1               = 0000000D
CHK_CRD_LOW                 000005D3 R      08
CHLOG_DISAB0              = 00000001
CHLOG_DISAB1              = 00000002
CH_MISSG0                 = 00010000
CH_MISSG1                 = 00008000
CH_REPAIR                 = 0021C000
CH_REPAIR_1               = 0021A000
CH_REPLG0                 = 00004000
CH_REPLG1                 = 00002000
CH_THRESHOLD              = 0000000A
CLEAR_ERRS_E                00000600 R      08
CPTIMEOUT                   0000015A R      08
CRDINTMAX                 = 00000003
CRDWATCHMAX               = 00000006
CRD_MS780E                  000005D7 R      08
CSPARITY                    000000A9 R      08
DAMPUTATE                   00000086 R      08
ECC$REENABLE                000003E6 RG     08
EMB$B_MC_SUMCOD           = 00000010
EMB$K_AW                  = 00000007
EMB$K_BE                  = 00000004
EMB$K_HE                  = 00000008
EMB$K_MC                  = 00000002
EMB$K_SA                  = 00000005
EMB$K_SE                  = 00000006
EMB$W_MC_ENTRY            = 00000004
ENAB_MA780                  0000050F R      08
ENAB_MS780C                 00C004ED R      08
ENAB_MS780E                 000004FA R      08
ENAB_ROUTINES               00000000 R      05
ERL$ALLOCEMB                ********   X    08
ERL$RELEASEMB               ********   X    08
EXE$AB_MEMERR               00000010 RG     07
EXE$GL_BADTIMOUT            0000002C RG     07
EXE$GL_CH1OLD               00000004 RG     07
EXE$GL_CH2OLD               00000008 RG     07
EXE$GL_CHSTATE              00000028 RG     07
EXE$GL_CONFREGL             ********   X    08
EXE$GL_CPTIMOUT             0000000C RG     07
EXE$GL_CRDCNT               00000024 RG     07
EXE$GL_CSBITA               00000000 RG     07
EXE$GL_FLAGS                ********   X    08
EXE$GL_MCHKERRS             ********   X    08
EXE$GL_MEMERRS              ********   X    08
EXE$GL_NUMNEXUS             ********   X    08
EXE$GW_REENAB               00000020 RG     07
EXE$GW_WATCH                00000022 RG     07
EXE$INT54                   0000044C RG     08
EXE$INT58                   00000438 RG     08
EXE$INT5C                   00000328 RG     08
EXE$INT60                   0000033C RG     08
EXE$LOGAWE                  0000033C RG     08
EXE$LOGCRD                  0000044C RG     08
EXE$LOGSBA                  00000438 RG     08
EXE$LOGSBF                  00000328 RG     08
EXE$MCHECK                  ********   X    08
EXE$MCHK                    00000000 RG     08
EXE$MCHK_BUGCHK             ********   X    08
EXE$MCHK_ERRCNT             00000000 RG     07
EXE$MCHK_PRTCT              ********   X    08
EXE$MCHK_TEST               ********   X    08
EXE$V_CRDENABL              ********   X    08
EXIT_MA                     00000709 R      08
FATAL_MEM                   000005F9 R      08
GENERAL_MEMTYP            = 00000003
GL_BADTIMOUT                0000002C R      07
GL_CH1OLD                   00000004 R      07
GL_CH2OLD                   00000008 R      07
GL_CHSTATE                  00000028 R      07
GL_CPTIMOUT                 0000000C R      07
GL_CRDCNT                   00000024 R      07
GL_CSBITA                   00000000 R      07
GW_REENAB                   00000020 R      07
GW_WATCH                    00000022 R      07
IBROMCHECK                  000000A9 R      08
INT54                       0000044C R      08
INT58                       00000438 R      08
INT5C                       00000328 R      08
INT60                       0000033C R      08
IOC$GL_ADPLIST              ********   X    08
IPL$_ASTDEL               = 00000002
LOCATE_MEM                  000004C0 R      08
LOGALL_ROUTINES             00000000 R      04
LOGAWE                      0000033C R      08
LOGC                        00000536 R      08
LOGCRD                      0000044C R      08
LOGE                        00000587 R      08
LOGERR_ROUTINES             00000000 R      03
LOGGER                      000002CC R      08
LOGIT                       000002E7 R      08
LOGMA                       0000065A R      08
LOGMEM                      00000476 R      08
LOGSBA                      00000438 R      08
LOGSBF                      00000328 R      08
LOGSBI                      0000037A R      08
LOG_ALL_MEM                 0000046D R      08
```

```
LOG_E                   000005FB R      08
LOG_ERROR_MEM           00000462 R      08
LOG_MA                  00000704 R      08
LOG_MA780               0000060C R      08
LOG_MS780C              00000529 R      08
LOG_MS780E              00000579 R      08
LOWER                   000005B2 R      08
MCHK                    00000000 R      08
MCHK$GL_LOG             00000305 RG     08
MCHK$M_LOG            = 00000001
MCHK$M_MCK            = 00000002
MCHK$M_NEXM           = 00000004
MCHK_ERRCNT             00000000 R      07
MCL_CES               = 00000008
MCL_COUNT             = 00000000
MCL_D                 = 00000014
MCL_PARITY            = 00000024
MCL_PC                = 0000002C
MCL_PSL               = 00000030
MCL_SBIERR            = 00000028
MCL_SUMMARY           = 00000004
MCL_TBER0             = 00000018
MCL_TBER1             = 0000001C
MCL_TIMOADDR          = 00000020
MCL_UPC               = 0000000C
MCL_VA                = 00000010
MEMTYP                  00000000 R      02
MEMTYPCNT             = 00000012
MMG$AL_SYSPCB           ******** X      08
MMG$DECONPFN            ******** X      08
MMG$DELWSLEX            ******** X      08
MMG$GL_MAXPFN           ******** X      08
MMG$GL_SBICONF          ******** X      08
MMG$GW_BIGPFN           ******** X      08
MMG$INSPFNT             ******** X      08
MMG$REFCNTNEG           ******** X      08
MMG$SVAPTECHK           ******** X      08
MRC$M_CRDERR          = 00000200
MRC$M_CTLOPTY         = 00040000
MRC$M_CTL1PTY         = 00080000
MRC$M_ELSRF           = 10000000
MRC$M_HERIMF          = 20000000
MRC$M_IFPTY           = 00000100
MRC$M_INHBCRD         = 40000000
MRC$M_INVMAPPTY       = 80000000
MRC$M_MSEQPTY         = 00000080
MRC$M_SUMMARY         = 00100000
MRC$V_CRDERR          = 00000009
MRC$V_CTLOPTY         = 00000012
MRC$V_CTL1PTY         = 00000013
MRC$V_ELSRF           = 0000001C
MRC$V_HERIMF          = 0000001D
MRC$V_IFPTY           = 00000008
MRC$V_INHBCRD         = 0000001E
MRC$V_INVMAPPTY       = 0000001F
MRC$V_MSEQPTY         = 00000007
MRC$V_SUMMARY         = 00000014
MSCP_ERRLOG             00000000 R      07
NDT$_MEM16I           = 00000011
NDT$_MEM16NI          = 00000010
NDT$_MEM256EIL        = 00000071
NDT$_MEM256EIU        = 00000073
NDT$_MEM256I          = 00000074
NDT$_MEM256NIL        = 00000070
NDT$_MEM256NIU        = 00000072
NDT$_MEM4I            = 00000009
NDT$_MEM4NI           = 00000008
NDT$_MEM64EIL         = 00000069
NDT$_MEM64EIU         = 0000006B
NDT$_MEM64I           = 0000006C
NDT$_MEM64NIL         = 00000068
NDT$_MEM64NIU         = 0000006A
NDT$_MPM0             = 00000040
NDT$_MPM1             = 00000041
NDT$_MPM2             = 00000042
NDT$_MPM3             = 00000043
NOLOG_MA                000006FF R      08
PCB$L_PHD             = 0000006C
PFN$AB_STATE            ******** X    08
PFN$AB_TYPE             ******** X    08
PFN$AW_REFCNT           ******** X    08
PFN$AX_WSLX             ******** X    08
PFN$C_BADPAGLST       = 00000002
PFN$C_PROCESS         = 00000000
PFN$C_SYSTEM          = 00000001
PFN$M_BADPAG          = 00000020
PFN$M_MODIFY          = 00000080
PFN$S_PAGTYP          = 00000003
PFN$V_PAGTYP          = 00000000
PR$_IPL               = 00000012
PR$_KSP               = 00000000
PR$_TBIA              = 00000039
PR$_TBIS              = 0000003A
PR780$_SBIER          = 00000034
PR780$_SBIFS          = 00000030
PR780$_SBIMT          = 00000033
PR780$_SBIS           = 00000031
PR780$_SBISC          = 00000032
PR780$_SBITA          = 00000035
PR780$_TODR           = 0000001B
PSL$S_CURMOD          = 00000002
PSL$S_IPL             = 00000005
PSL$V_CURMOD          = 00000018
PSL$V_IPL             = 00000010
PSL$V_PRVMOD          = 00000016
PTE$S_PFN             = 00000015
PTE$V_MODIFY          = 0000001A
PTE$V_PFN             = 00000000
PTE$V_VALID           = 0000001F
PTE$V_WINDOW          = 00000015
RDSNORRES               000002BE R      08
READSUBST               000001BF R      08
REENABLE_INTS           00000405 R      08
REENABTIME            = 00000384
REFLECTCHK              000000D0 R      08
```

```
RESUMABLE                       0000070D R      08
RESUME                          00000098 R      08
SBIER$M_CPTO                  = 00001000
SBIER$M_CRD                   = 00004000
SBIER$M_IBRDS                 = 00000080
SBIER$M_IBTO                  = 00000040
SBIER$M_RDS                   = 00002000
SBIFS$V_NEF                   = 00000019
SCH$GL_CURPCB                   ********    X   08
SOMETIME                      = 0000003C
SS$_NORMAL                    = 00000001
TBUFPARITY                      00000074 R      08
TRYRESUME                       00000080 R      08
UPPER                           000005C3 R      08
```

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
MCHK$DATA0	00000012	(18.)	02	(2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC LONG
MCHK$DATA2	0000000C	(12.)	03	(3.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC LONG
MCHK$DATA3	0000000C	(12.)	04	(4.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC LONG
MCHK$DATA4	0000000C	(12.)	05	(5.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC LONG
MCHK$DATA1	00000012	(18.)	06	(6.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
MCHK$DATA	00000030	(48.)	07	(7.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC QUAD
MCHK$CODE	0000072D	(1837.)	08	(8.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.05	00:00:02.68
Command processing	107	00:00:00.39	00:00:04.01
Pass 1	416	00:00:09.91	00:00:35.99
Symbol table sort	6	00:00:01.26	00:00:05.80
Pass 2	282	00:00:02.90	00:00:11.71
Symbol table output	30	00:00:00.13	00:00:00.14
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	876	00:00:14.68	00:01:00.36

The working set limit was 1800 pages.
95397 bytes (187 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1210 non-local and 53 local symbols.
1641 source lines were read in Pass 1, producing 33 object records in Pass 2.
42 pages of virtual memory were used to define 36 macros.

```
                                        +----------------------------+
                                        ! Macro library statistics !
                                        +----------------------------+

Macro library name                       Macros defined
------------------                       --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                       23
_$255$DUA28:[SYSLIB]STARLET.MLB;2                     8
TOTALS (all libraries)                               31

1265 GETS were required to define 31 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:MCHECK780/OBJ=OBJ$:MCHECK780 MSRC$:MCHECK780/UPDATE=(ENH$:MCHECK780)+EXECML$/LIB
```

LIOSUB750
LIS

MCHECKUV1
LIS

MCHECK730
LIS

MCHECK750
LIS

LIOSUB730
LIS

LIOSUB790
LIS

MCHECK780
LIS

MCF790
LIS

MCHECK790
LIS

LIOSUB780
LIS

LIOSUBUV1
LIS